INTERNATIONAL TELECOMMUNICATION UNION

ITU-T

TELECOMMUNICATION
STANDARDIZATION SECTOR
OF ITU

H.235

(08/2003)

SERIES H: AUDIOVISUAL AND MULTIMEDIA SYSTEMS

Infrastructure of audiovisual services – Systems aspects

Security and encryption for H-series (H.323 and other H.245-based) multimedia terminals

ITU-T Recommendation H.235

ITU-T H-SERIES RECOMMENDATIONS

AUDIOVISUAL AND MULTIMEDIA SYSTEMS

CHARACTERISTICS OF VISUAL TELEPHONE SYSTEMS	H.100–H.199
INFRASTRUCTURE OF AUDIOVISUAL SERVICES	
General	H.200–H.219
Transmission multiplexing and synchronization	H.220–H.229
Systems aspects	**H.230–H.239**
Communication procedures	H.240–H.259
Coding of moving video	H.260–H.279
Related systems aspects	H.280–H.299
SYSTEMS AND TERMINAL EQUIPMENT FOR AUDIOVISUAL SERVICES	H.300–H.399
SUPPLEMENTARY SERVICES FOR MULTIMEDIA	H.450–H.499
MOBILITY AND COLLABORATION PROCEDURES	
Overview of Mobility and Collaboration, definitions, protocols and procedures	H.500–H.509
Mobility for H-Series multimedia systems and services	H.510–H.519
Mobile multimedia collaboration applications and services	H.520–H.529
Security for mobile multimedia systems and services	H.530–H.539
Security for mobile multimedia collaboration applications and services	H.540–H.549
Mobility interworking procedures	H.550–H.559
Mobile multimedia collaboration inter-working procedures	H.560–H.569
BROADBAND AND TRIPLE-PLAY MULTIMEDIA SERVICES	
Broadband multimedia services over VDSL	H.610–H.619

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

ITU-T Recommendation H.235

Security and encryption for H-series (H.323 and other H.245-based) multimedia terminals

Summary

This Recommendation describes enhancements within the framework of the H.3xx-series Recommendations to incorporate security services such as *Authentication* and *Privacy* (data encryption). The proposed scheme is applicable to both simple point-to-point and multipoint conferences for any terminals which utilize ITU-T Rec. H.245 as a control protocol.

For example, H.323 systems operate over packet-based networks which do not provide a guaranteed quality of service. For the same technical reasons that the base network does not provide QOS, the network does not provide a secure service. Secure real-time communication over insecure networks generally involves two major areas of concern – *authentication* and *privacy*.

This Recommendation describes the security infrastructure and specific privacy techniques to be employed by the H.3xx-series of multimedia terminals. This Recommendation will cover areas of concern for interactive conferencing. These areas include, but are not strictly limited to, authentication and privacy of all real-time media streams that are exchanged in the conference. This Recommendation provides the protocol and algorithms needed between the H.323 entities.

This Recommendation utilizes the general facilities supported in ITU-T Rec. H.245 and as such, any standard which operates in conjunction with this control protocol may use this security framework. It is expected that, wherever possible, other H-series terminals may interoperate and directly utilize the methods described in this Recommendation. This Recommendation will not initially provide for complete implementation in all areas, and will specifically highlight endpoint authentication and media privacy.

This Recommendation includes the ability to negotiate services and functionality in a generic manner, and to be selective concerning cryptographic techniques and capabilities utilized. The specific manner in which they are used relates to systems capabilities, application requirements and specific security policy constraints. This Recommendation supports varied cryptographic algorithms, with varied options appropriate for different purposes; e.g., key lengths. Certain cryptographic algorithms may be allocated to specific security services (e.g., one for fast media stream encryption and another for signalling encryption).

It should also be noted that some of the available cryptographic algorithms or mechanisms may be reserved for export or other national issues (e.g., with restricted key lengths). This Recommendation supports signalling of well-known algorithms in addition to signalling non-standardized or proprietary cryptographic algorithms. There are no specifically mandated algorithms; however, it is strongly suggested that endpoints support as many of the applicable algorithms as possible in order to achieve interoperability. This parallels the concept that the support of ITU-T Rec. H.245 does not guarantee the interoperability between two entities' codecs.

Version 2 of ITU-T Rec. H.235 supersedes ITU-T Rec. H.235 version 1 featuring several improvements such as elliptic curve cryptography, security profiles (simple password-based and sophisticated digital signature), new security countermeasures (media anti-spamming), support for the Advanced Encryption Algorithm (AES), support for backend service, object identifiers defined and changes incorporated from the H.323 implementors guide.

Version 3 of ITU-T Rec. H.235 supersedes ITU-T Rec. H.235 version 2 featuring a procedure for encrypted DTMF signals, object identifiers for the AES encryption algorithm for media payload encryption, the enhanced OFB (EOFB) stream-cipher encryption mode for encryption of media streams, an authentication-only option in Annex D for smooth NAT/firewall traversal, a key distribution procedure on the RAS channel, procedures for more secure session key transport and more robust session key distribution and updating, procedures for securing multiple payload streams, better security support for direct-routed calls in a new Annex I, signalling means for more flexible error reporting, clarifications and efficiency improvements for fast start security and for Diffie-Hellman signalling along with longer Diffie-Hellman parameters and changes incorporated from the H.323 implementors guide.

Source

ITU-T Recommendation H.235 was approved by ITU-T Study Group 16 (2001-2004) under the ITU-T Recommendation A.8 procedure on 6 August 2003.

Keywords

Authentication, certificate, digital signature, encryption, integrity, key management, multimedia security, security profile.

History

Version	Approval
H.235v1	1998-02-06
H.235v2	2000-11-17
H.235v3	2003-08-06

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 Assembly (WTSA), 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 WTSA 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.

Compliance with this Recommendation is voluntary. However, the Recommendation may contain certain mandatory provisions (to ensure e.g., interoperability or applicability) and compliance with the Recommendation is achieved when all of these mandatory provisions are met. The words "shall" or some other obligatory language such as "must" and the negative equivalents are used to express requirements. The use of such words does not suggest that compliance with the Recommendation is required of any party.

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 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

ITU-T Recommendation H.235

Security and encryption for H-series (H.323 and other H.245-based) multimedia terminals

1 Scope

The primary purpose of this Recommendation is to provide for authentication, privacy, and integrity within the current H-series protocol framework. The current text of this Recommendation (2003) provides details on implementation with ITU-T Rec. H.323. This framework is expected to operate in conjunction with other H-series protocols that utilize ITU-T Rec. H.245 as their control protocol.

Additional goals in this Recommendation include:

1) Security architecture should be developed as an extensible and flexible framework for implementing a security system for H-series terminals. This should be provided through flexible and independent services and the functionality that they supply. This includes the ability to negotiate and to be selective concerning cryptographic techniques utilized, and the manner in which they are used.
2) Provide security for all communications occurring as a result of H.3xx protocol usage. This includes aspects of connection establishment, call control, and media exchange between all entities. This requirement includes the use of confidential communication (privacy), and may exploit functions for peer authentication as well as protection of the user's environment from attacks.
3) This Recommendation should not preclude integration of other security functions in H.3xx entities which may protect them against attacks from the network.
4) This Recommendation should not limit the ability for any H.3xx-series Recommendation to scale as appropriate. This may include both the number of secured users and the levels of security provided.
5) Where appropriate, all mechanisms and facilities should be provided independent of any underlying transport or topologies. Other means that are outside the scope of this Recommendation may be required to counter such threats.
6) Provisions are made for operation in a mixed environment (secured and unsecured entities).
7) This Recommendation should provide facilities for distributing session keys associated with the cryptography utilized. (This does not imply that public-key-based certificate management must be part of this Recommendation.)
8) This Recommendation provides two security profiles that facilitate interoperability. Annex D describes a simple, yet secure password-based security profile while Annex E is a signature security profile deploying digital signatures, certificates and a public-key infrastructure that overcomes the limitations of Annex D.

The security architecture, described in this Recommendation, does not assume that the participants are familiar with each other. It does, however, assume that appropriate precautions have been taken to physically secure the H-series endpoints. The principal security threat to communications, therefore, is assumed to be eavesdropping on the network, or some other method of diverting media streams.

ITU-T Rec. H.323 provides the means to conduct an audio, video and data conference between two or more parties, but does not provide the mechanism to allow each participant to authenticate the identity of the other participants, nor provide the means to make the communications private (i.e., encrypt the streams).

ITU-T Recs H.323, H.324 and H.310 make use of the logical channel signalling procedures of ITU-T Rec. H.245, in which the content of each logical channel is described when the channel is opened. Procedures are provided for expression of receiver and transmitter capabilities, transmissions are limited to what receivers can decode, and receivers may request a particular desired mode from transmitters. The security capabilities of each endpoint are communicated in the same manner as any other communication capability.

Some H-series (H.323) terminals may be used in multipoint configurations. The security mechanism described in this Recommendation will allow for secure operation in these environments, including both centralized and decentralized MCU operation.

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; 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. The reference to a document within this Recommendation does not give it, as a stand-alone document, the status of a Recommendation.

- ITU-T Recommendation H.225.0 (2003), *Call signalling protocols and media stream packetization for packet-based multimedia communication systems*.
- ITU-T Recommendation H.235 (1998), *Security and encryption for H-series (H.323 and other H.245-based) multimedia terminals*.
- ITU-T Recommendation H.235 (2000), *Security and encryption for H-series (H.323 and other H.245-based) multimedia terminals*.
- ITU-T Recommendation H.530 (2002), *Symmetric security procedures for H.323 mobility in H.510*.
- ITU-T Recommendation H.530 Cor.1 (2003), *Symmetric security procedures for H.323 mobility in H.510*.
- ITU-T Recommendation H.245 (2003), *Control protocol for multimedia communication*.
- ITU-T Recommendation H.323 (2003), *Packet-based multimedia communications systems*.
- ITU-T Recommendation Q.931 (1998), *ISDN user-network interface layer 3 specification for basic call control*.
- ITU-T Recommendation X.509 (2000) | ISO/IEC 9594-8:2001, *Information technology – Open Systems Interconnection – The Directory: Public-key and attribute certificate frameworks*.
- ITU-T Recommendation X.800 (1991), *Security architecture for Open Systems Interconnection for CCITT applications*.

  ISO 7498-2:1989, *Information processing systems – Open Systems Interconnection – Basic Reference Model – Part 2: Security Architecture*.
- ITU-T Recommendation X.803 (1994) | ISO/IEC 10745:1995, *Information technology – Open Systems Interconnection – Upper layers security model*.
- ITU-T Recommendation X.810 (1995) | ISO/IEC 10181-1:1996, *Information technology – Open Systems Interconnection – Security frameworks for open systems: Overview*.

– ITU-T Recommendation X.811 (1995) | ISO/IEC 10181-2:1996, *Information technology – Open Systems Interconnection – Security frameworks for open systems: Authentication framework*.

– ISO/IEC 9797:1994, *Information technology – Security techniques – Data integrity mechanism using a cryptographic check function employing a block cipher algorithm*.

– ISO/IEC 9798-2:1999, *Information technology – Security techniques – Entity authentication – Part 2: Mechanisms using symmetric encipherment algorithms*.

– ISO/IEC 9798-3:1998, *Information technology – Security techniques – Entity authentication – Part 3: Mechanism using digital signature techniques*.

– ISO/IEC 9798-4:1999, *Information technology – Security techniques – Entity authentication – Part 4: Mechanisms using a cryptographic check function*.

– ISO/IEC 10116:1997, *Information technology – Security techniques – Modes of operation for an n-bit block cipher*.

– ISO/IEC 15946-1:2002, *Information technology – Security techniques – Cryptographic techniques based on elliptic curves – Part 1: General*.

– ISO/IEC 15946-2:2002, *Information technology – Security techniques – Cryptographic techniques based on elliptic curves – Part 2: Digital signatures*.

– ATM Forum: af-sec-0100.002 (2001), *ATM Security Specification Version 1.1*.

– IETF RFC 1321 (1992), *The MD5 Message-Digest Algorithm*.

– IETF RFC 2104 (1997), *HMAC: Keyed-Hashing for Message Authentication*.

– IETF RFC 2865 (2000), *Remote Authentication Dial In User Service (RADIUS)*.

– IETF RFC 2198 (1997), *RTP Payload for Redundant Audio Data*.

– IETF RFC 2246 (1999), *The TLS Protocol Version 1.0*.

– IETF RFC 2401 (1998), *Security Architecture for the Internet Protocol*.

– IETF RFC 2402 (1998), *IP Authentication Header*.

– IETF RFC 2407 (1998), *The Internet IP Security Domain of Interpretation for ISAKMP*.

– IETF RFC 2412 (1998), *The OAKLEY Key Determination Protocol*.

– IETF RFC 2437 (1998), *PKCS #1: RSA Cryptography Specifications Version 2.0*.

– IETF RFC 2833 (2000), *RTP Payload for DTMF Digits, Telephony Tones and Telephony Signals*.

– IETF RFC 3280 (2002), *Internet X.509 Public Key Infrastructure Certificate and Certificate Revocation List (CRL) Profile*.

3 Terms and definitions

For the purposes of this Recommendation, the definitions given in clauses 3/H.323, 3/H.225.0 and 3/H.245 apply along with those in this clause. Some of the following terms are used as defined in ITU-T Rec. X.800 | ISO 7498-2 and in ITU-T Recs X.803, X.810 and X.811.

3.1 access control: The prevention of unauthorized use of a resource, including the prevention of use of a resource in an unauthorized manner (ITU-T Rec. X.800).

3.2 authentication: The provision of assurance of the claimed identity of an entity (ITU-T Rec. X.811).

3.3 authorization: The granting of permission on the basis of authenticated identification.

3.4 attack: The activities undertaken to bypass or exploit deficiencies in a system's security mechanisms. By a direct attack on a system they exploit deficiencies in the underlying algorithms, principles, or properties of a security mechanism. Indirect attacks are performed when they bypass the mechanism, or when they make the system use the mechanism incorrectly.

3.5 certificate: A set of security-relevant data issued by a security authority or trusted third party, together with security information which is used to provide the integrity and data origin authentication services for the data (ITU-T Rec. X.810). In this Recommendation, the term refers to "public key" certificates which are values that represent an owner's public key (and other optional information) as verified and signed by a trusted authority in an unforgeable format.

3.6 cipher: A cryptographic algorithm, a mathematical transform.

3.7 confidentiality: The property that prevents disclosure of information to unauthorized individuals, entities, or processes.

3.8 cryptographic algorithm: Mathematical function that computes a result from one or several input values.

3.9 encipherment: Encipherment (encryption) is the process of making data unreadable to unauthorized entities by applying a cryptographic algorithm (an encryption algorithm). Decipherment (decryption) is the reverse operation by which ciphertext is transformed to plaintext.

3.10 integrity: The property that data has not been altered in an unauthorized manner.

3.11 key management: The generation, storage, distribution, deletion, archiving and application of keys in accordance with a security policy (ITU-T Rec. X.800).

3.12 media stream: A media stream can be of type audio, video or data or a combination of any of them. Media stream data conveys user or application data (payload) but no control data.

3.13 non-repudiation: Protection from denial by one of the entities involved in a communication of having participated in all or part of the communication.

3.14 privacy: A mode of communication in which only the explicitly enabled parties can interpret the communication. This is typically achieved by encryption and shared key(s) for the cipher.

3.15 private channel: For this Recommendation, a private channel is one that is a result of prior negotiation on a secure channel. In this context, it may be used to handle media streams.

3.16 public key cryptography: An encryption system utilizing asymmetric keys (for encryption/decryption) in which the keys have a mathematical relationship to each other which cannot be reasonably calculated.

3.17 security profile: A (sub)set of consistent, interoperable procedures and features out of ITU-T Rec. H.235 useful for securing H.323 multimedia communication among the involved entities in a specific scenario.

3.18 spamming: A denial-of-service attack when sending unauthorized data in excess to a system. A special case is media spamming when sending RTP packets on UDP ports. Usually the system is flooded with packets; the processing consumes precious system resources.

3.19 symmetric (secret-key based) cryptographic algorithm: An algorithm for performing encipherment or the corresponding algorithm for performing decipherment in which the same key is required for both encipherment and decipherment (ITU-T Rec. X.810).

3.20 threat: A potential violation of security (ITU-T Rec. X.800).

4 Symbols and abbreviations

This Recommendation uses the following abbreviations:

X || Y	Concatenation of X and Y
3DES	Triple DES
AES	Advanced Encryption Algorithm
ASN.1	Abstract Syntax Notation No. 1
BES	Back-end Server
CA	Certificate Authority
CBC	Cipher Block Chaining
CFB	Cipher Feedback mode
CRL	Certificate Revocation List
DES	Data Encryption Standard
DH	Diffie-Hellman
DNS	Domain Name System
DSS	Digital Signature Standard
DTMF	Dual Tone Multi-Frequency
ECB	Electronic Code Book
ECC and EC	Elliptic Curve Cryptosystem (see section 8.7 of *ATM Forum Security Specification Version 1.1*). A public-key cryptosystem.
EC-GDSA	Elliptic curve digital signature with appendix analog of the NIST Digital Signature Algorithm (DSA) (see also ISO/IEC 15946-2, chapter 5)
ECKAS-DH	Elliptic Curve Key Agreement Scheme – Diffie-Hellman. The Diffie-Hellman key agreement scheme using elliptic curve cryptography
EOFB	Enhanced OFB mode
EP	Endpoint
GK	Gatekeeper
GW	Gateway
ICV	Integrity Check Value
ID	Identifier
IPSEC	Internet Protocol Security
ISAKMP	Internet Security Association Key Management Protocol
IV	Initialization Vector
LDAP	Lightweight Directory Access Protocol
MAC	Message Authentication Code
MCU	Multipoint Control Unit
MD5	Message Digest 5
MPS	Multiple Payload Stream

NAT	Network Address Translation
OCSP	Online Certificate Status Protocol
OFB	Output Feedback Mode
OID	Object Identifier
PDU	Protocol Data Unit
PKCS	Public-Key Crypto System
PKI	Public Key Infrastructure
PRF	Pseudo-Random Function
QOS	Quality of Service
RSA	Rivest, Shamir and Adleman (public key algorithm)
RTCP	Real-time Transport Control Protocol
RTP	Real-time Transport Protocol
SDU	Service Data Unit
SHA1	Secure Hash Algorithm 1
SRTP	Secure Real-Time Transport Protocol
SSL	Secure Socket Layer
TLS	Transport Level Security
TSAP	Transport Service Access Point
XOR, ⊕	Exclusive OR

5 Conventions

In this Recommendation the following conventions are used:

– "shall" indicates a mandatory requirement.

– "should" indicates a suggested but optional course of action.

– "may" indicates an optional course of action rather than a recommendation that something take place.

References to clauses, subclauses, annexes and appendices refer to those items within this Recommendation unless another Recommendation is explicitly listed. For example, "1.4" refers to clause 1.4 of this Recommendation; "6.4/H.245" refers to clause 6.4 in ITU-T Rec. H.245.

This Recommendation describes the use of "n" different message types: H.245, RAS, Q.931, etc. To distinguish between the different message types, the following convention is followed. H.245 message and parameter names consist of multiple concatenated words highlighted in bold typeface (**maximumDelayJitter**). RAS message names are represented by three-letter abbreviations (**ARQ**). Q.931 message names consist of one or two words with the first letters capitalized (**Call Proceeding**).

This Recommendation defines various object identifiers (OIDs) for signalling security capabilities, procedures or security algorithms. These OIDs relate to a hierarchical tree of assigned values that may origin from external sources or are part of the ITU-T maintained OID tree. Those OIDs that are specifically related to ITU-T Rec. H.235 have the following appearance in the text:

"OID" = {itu-t (0) recommendation (0) h (8) 235 version (0) **V N**} where **V** symbolically represents a single decimal digit denoting the corresponding version of ITU-T Rec. H.235; e.g., 1, 2 or 3.

N symbolically represents a decimal number uniquely identifying the instance of the OID and thus, the procedure, algorithm or security capability.

Thus, the ASN.1 encoded OID consists of a sequence of numbers. For convenience, a textual mnemonic shorthand string notation for each OID is used in the text such as "OID". A mapping is given that relates each OID string with the ASN.1 sequence of numbers. Implementations conforming to ITU-T Rec. H.235 shall use only the ASN.1 encoded numbers.

When deploying media encryption in conjunction with payload padding, the text sometimes says "the value of the pad should be determined by the normal convention of the cipher algorithm"; see e.g., 8.6.1, B.2.4 and Figure I.5. This means that some cipher algorithms (e.g., DES) provide further implementation advice as to how the sender may choose the value of the padding byte(s). Examples could be random fill-in values, static values or other generated patterns. Whatever method is deployed does not impact interoperability, yet the security quality may well be different. This is considered as an implementation matter and is not specified any further in this Recommendation.

6 System introduction

6.1 Summary

1) The call signalling channel may be secured using TLS [TLS] or IPSEC [IPSEC] on a secure well-known port (ITU-T Rec. H.225.0).
2) Users may be authenticated either during the initial call connection, in the process of securing the H.245 channel and/or by exchanging certificates on the H.245 channel.
3) The encryption capabilities of a media channel are determined by extensions to the existing capability negotiation mechanism.
4) Initial distribution of key material from the master is via H.245 **OpenLogicalChannel** or **OpenLogicalChannelAck** messages.
5) Re-keying may be accomplished by H.245 commands: **EncryptionUpdateCommand**, **EncryptionUpdateRequest, EncryptionUpdate and EncryptionUpdateAck**.
6) Key material distribution is protected either by operating the H.245 channel as a private channel or by specifically protecting the key material using the selected exchanged certificates.
7) The security protocols presented conform either to ISO published standards or to IETF proposed standards.

6.2 Authentication

The process of authentication verifies that the respondents are, in fact, who they say they are. Authentication may be accomplished in conjunction with the exchange of public key-based certificates. Authentication may also be accomplished by an exchange which utilizes a shared secret between the entities involved. This may be a static password or some other *a priori* piece of information.

This Recommendation describes the protocol for exchanging the certificates, but does not specify the criteria by which they are mutually verified and accepted. In general, certificates give some assurance to the verifier that the presenter of the certificate is who he says he is. The intent behind the certificate exchange is to authenticate the *user* of the endpoint, not simply the physical endpoint. Using digital certificates, an authentication protocol proves that the respondents possess the private keys corresponding to the public keys contained in the certificates. This authentication protects against man-in-the-middle attacks, but does not automatically prove who the respondents are. To do this normally requires that there be some policy regarding the other contents of the certificates. For

authorization certificates, for example, the certificate would normally contain the service-provider's identification along with some form of user account identification prescribed by the service provider.

The authentication framework in this Recommendation does not prescribe the contents of certificates (i.e., does not specify a certificate policy) beyond that required by the authentication protocol. However, an application using this framework may impose high-level policy requirements such as presenting the certificate to the user for approval. This higher level policy may either be automated within the application or require human interaction.

For authentication which does not utilize digital certificates, this Recommendation provides the signalling to complete various challenge/response scenarios. This method of authentication requires prior coordination by the communicating entities so that a shared secret may be obtained. An example of this method would be a customer of a subscription-based service.

As a third option, the authentication may be completed within the context of a separate security protocol such as TLS [TLS] or IPSEC [IPSEC].

Both bidirectional and unidirectional authentication may be supported by peer entities. This authentication may occur on some or all of the communication channels.

All of the specific authentication mechanisms described in this Recommendation are identical to, or derived from, ISO-developed algorithms as specified in Parts 2 to 3 of ISO/IEC 9798, or based on IETF protocols.

6.2.1 Certificates

The standardization of certificates, including their generation, administration and distribution is outside the scope of this Recommendation. The certificates used to establish secure channels (call signalling and/or call control) shall conform to those prescribed by whichever protocol has been negotiated to secure the channel.

It should be noted that for authentication utilizing public key certificates, the endpoints are required to provide digital signatures using the associated private key value. The exchange of public key certificates alone does not protect against man-in-the-middle attacks. The H.235 protocols conform to this requirement.

6.3 Call establishment security

There are at least two reasons to motivate securing the call establishment channel (e.g., H.323 using Q.931). The first is for simple authentication, before accepting the call. The second reason is to allow for call authorization. If this functionality is desired in the H-series terminal, a secure mode of communication should be used (such as TLS/IPSEC for H.323) before the exchange of call connection messages. Alternatively, the authorization may be provided based upon a service-specific authentication. The constraints of a service-specific authorization policy are outside the scope of this Recommendation.

6.4 Call control (H.245) security

The call control channel (H.245) should also be secured in some manner to provide for subsequent media privacy. The H.245 channel shall be secured using any negotiated privacy mechanism (this includes the option of "none"). H.245 messages are utilized to signal encryption algorithms and encryption keys used in the shared, private, media channels. The ability to do this, on a logical channel by logical channel basis, allows different media channels to be encrypted by different mechanisms. For example, in centralized multipoint conferences, different keys may be used for streams to each endpoint. This may allow media streams to be made private for each endpoint in the conference. In order to utilize the H.245 messages in a secure manner, the entire H.245 channel (logical channel 0) should be opened in a negotiated secure manner.

The mechanism by which H.245 is made secure is dependent on the H-series terminals involved. The only requirement on all systems that utilize this security structure is that each shall have some manner in which to negotiate and/or signal that the H.245 channel is to be operated in a particular secured manner before it is actually initiated. For example, H.323 will utilize the H.225.0 connection signalling messages to accomplish this.

6.5 Media stream privacy

This Recommendation describes media privacy for media streams carried on packet-based transports. These channels may be unidirectional with respect to H.245 logical channel characterizations. The channels are not required to be unidirectional on a physical or transport level.

A first step in attaining media privacy should be the provision of a private control channel on which to establish cryptographic keying material and/or set up the logical channels which will carry the encrypted media streams. For this purpose, when operating in a secure conference, any participating endpoints may utilize an encrypted H.245 channel. In this manner, cryptographic algorithm selection and encryption keys as passed in the H.245 **OpenLogicalChannel** command are protected.

The H.245 secure channel may be operated with characteristics different from those in the private media channel(s) as long as it provides a mutually acceptable level of privacy. This allows for the security mechanisms protecting media streams and any control channels to operate in a completely independent manner, providing completely different levels of strength and complexity.

If it is required that the H.245 channel be operated in a non-encrypted manner, the specific media encryption keys may be encrypted separately in the manner signalled and agreed to by the participating parties. A logical channel of type **h235Control** may be utilized to provide the material to protect the media encryption keys. This logical channel may be operated in any appropriately negotiated mode.

The privacy (encryption) of data carried in logical channels shall be in the form specified by the **OpenLogicalChannel**. Transport-specific header information shall not be encrypted. The privacy of data is to be based upon end-to-end encryption.

6.6 Trusted elements

The basis for authentication (trust) and privacy is defined by the terminals of the communications channel. For a connection establishment channel, this may be between the caller and a hosting network component. For example, a telephone "trusts" that the network switch will connect it with the telephone whose number has been dialled. For this reason, any entity which terminates an encrypted H.245 control channel or any **encryptedData** type logical channels shall be considered a trusted element of the connection; this may include MC(U)s and gateways. The result of trusting an element is the confidence to reveal the privacy mechanism (algorithm and key) to that element.

Given the above, it is incumbent upon participants in the communications path to authenticate any and all "trusted" elements. This will normally be done by certificate exchange as would occur for the "standard" end-to-end authentication. This Recommendation will not require any specific level of authentication, other than to suggest that it be acceptable to all entities using the trusted element. Details of a trust model and certificate policy are for further study.

Privacy can be assured between the two endpoints only if connections between trusted elements are proven to be protected against man-in-the-middle attacks.

6.6.1 Key escrow

Although not specifically required for operation, this Recommendation contains provision for entities utilizing the H.235 protocol to support the facility known as trusted third party (TTP) within the signalling elements.

The ability to recover lost media encryption keys should be supported in installations where this functionality is desired or required.

Key escrow is a facility which is often referred to as a Trusted Third Party (TTP). This facility is for further study.

6.7 Non-repudiation

For further study.

6.8 Mobility security

H.323-based systems may be deployed in a mobility environment according to ITU-T Rec. H.510. Security procedures and protocols for such systems are described in ITU-T Rec. H.530. ITU-T Rec. H.530 deploys protocols and procedures from this Recommendation.

6.9 Security profiles

This Recommendation includes a couple of annexes (i.e., Annexes D, E and F) that each hold security profiles of H.235. A security profile specifies specific usage of H.235 or a subset of H.235 functionality for well-defined environments with scoped applicability.

Depending on the environment and application, security profiles may be implemented either selectively or altogether. Typically, H.235-enabled systems indicate within object identifiers as part of signalling messages which security profiles they deploy. H.235-enabled systems should choose the security profile according to their needs.

Optionally, endpoints may initially offer multiple security profiles simultaneously, in RRQ/GRQ messages, and let the gatekeeper select the most adequate one by answering it in the RCF/GCF message. LRQ/LCF transactions between gatekeepers may also carry several security profiles. When calculating digital signatures or hash values to provide message integrity, first the hash values and digital signatures which do not provide message integrity should be calculated over the field subset and set in the message, all the digital signatures and hash values that provide message integrity should be set to zeroes in the message buffer, then all the digital signatures and hash values should be calculated using this buffer, and then set in the message.

7 Connection establishment procedures

7.1 Introduction

As stated in the system introduction clause, both the call connection channel (H.225.0 for H.323-series) and call control (H.245) channel shall operate in the negotiated secured or unsecured mode starting with the first exchange. For the call connection channel, this is done *a priori* (for H.323, a TLS secured TSAP (port 1300) shall be utilized for the Q.931 messages). For the call control channel, security mode is determined by information passed in the initial connection setup protocol in use by the H-series terminal.

In the cases in which there are no overlapping security capabilities, the called terminal may refuse the connection. The error returned should convey no information about any security mismatch; the calling terminal will have to determine the problem by some other means. In cases where the calling terminal receives a message without sufficient security capabilities, it should terminate the call.

If the calling and called terminals have compatible security capabilities, it shall be assumed by both sides that the H.245 channel shall operate in the secure mode negotiated. Failure to set up the H.245 channel in the secure mode determined here should be considered a protocol error and the connection terminated.

8 H.245 signalling and procedures

In general, the privacy aspects of media channels are controlled in the same manner as any other encoding parameter; each terminal indicates its capabilities, the source of the data selects a format to use, and the receiver acknowledges or denies the mode. All transport-independent aspects of the mechanism such as algorithm selection are indicated in generic logical channel elements. Transport specifics such as key/encryption algorithm synchronization are passed in transport-specific structures.

8.1 Secure H.245 channel operation

Assuming that the connection procedures in the previous clause (Connection establishment procedures) indicate a secure mode of operation, the negotiated handshake and authentication shall occur for the H.245 control channel before any other H.245 messages are exchanged. If negotiated, any exchange of certificates shall occur using any mechanism appropriate for the H-series terminal(s). After completing the securing of the H.245 channel, the terminals use the H.245 protocol in the same manner that they would in an insecure mode.

8.2 Unsecured H.245 channel operation

Alternatively, the H.245 channel may operate in an unsecured manner and the two entities open a secure logical channel with which to perform authentication and/or shared-secret derivation. For example, TLS or IPSEC may be utilized by opening a logical channel with the **dataType** containing a value for **h235Control**. This channel could then be used to derive a shared secret which protects any media session keys or to transport the **EncryptionSync**.

8.3 Capability exchange

Following the procedures in 5.2/H.245 (Capability exchange procedures) and the appropriate H-series system Recommendation, endpoints exchange capabilities using H.245 messages. These capability sets may now contain definitions which indicate security and encryption parameters. For example, an endpoint might provide capabilities to send and receive H.261 video. It may also signal the ability to send and receive encrypted H.261 video.

Each encryption algorithm that is utilized in conjunction with a particular media codec implies a new capability definition. As with any other capability, endpoints may supply both independent and dependent encrypted codecs in their exchange. This will allow endpoints to scale their security capabilities based upon overheads and resources available.

After capability exchange has been completed, endpoints may open secure logical channels for media in the same manner that they would in an insecure manner.

8.4 Master role

The H.245 master-slave is used to establish the master entity for the purpose of bidirectional channel operation and other conflict resolution. This role of master is also utilized in the security methods. Although the security mode(s) of a media stream is set by the source (in deference to the capabilities of the receiver), the master is the endpoint which generates the encryption key. This generation of the encryption key is done, regardless of whether the master is the receiver or the source of the encrypted media. In order to allow for multicast channel operation with shared keys, the MC (also the master) should generate the keys.

8.5 Logical channel signalling

Endpoints open secure media logical channels in the same manner that they open unsecured media logical channels. Each channel may operate in a completely independent manner from other channels – in particular where this pertains to security. The particular mode shall be defined in the

OpenLogicalChannel dataType field. The initial encryption key shall be passed in either the **OpenLogicalChannel** or **OpenLogicalChannelAck** depending on the master/slave relationship of the originator of the **OpenLogicalChannel**.

The **OpenLogicalChannelAck** shall act as confirmation of the encryption mode. If the **openLogicalChannel** is unacceptable to the recipient, either **dataTypeNotSupported** or **dataTypeNotAvailable** (transient condition) shall be returned in the cause field of the **OpenLogicalChannelReject**.

During the protocol exchange that establishes the logical channel, the encryption key shall be passed from the master to the slave (regardless of who initiated the **OpenLogicalChannel**). For media channels opened by an endpoint (other than the master), the master shall return the initial encryption key and the initial synchronization point in the **OpenLogicalChannelAck** (in the **encryptionSync** field). For media channels opened by the master, the **OpenLogicalChannel** shall include the initial encryption key and the synchronization point in the **encryptionSync** field.

8.6 Fast connect security

Endpoints may deploy the fast connect procedure (see 8.1.7 and 8.1.7.1/H.323) using the fast start element for securely exchanging key material (master key and session encryption keys). The procedures given in 8.6.1 describe "plain" fast start that does not use multiple offered encryption algorithms whereas 8.6.1.1 describes the particular case of fast start with multiple offered encryption algorithms that enables more compact message encoding.

8.6.1 Unidirectional fast start security

This procedure describes how to establish a (half-duplex) unidirectional security logical channel from the caller to the callee.

Procedures of the caller

The caller (source of the **Setup**) presents both its DH token, and the supported FastStart structures. The DH token shall be conveyed within an embedded ClearToken as part of a CryptoToken, or as a separate ClearToken, see also 8.8. During the **Setup**-to-**Connect** sequence, a Diffie-Hellman (DH) exchange shall be performed: this seeds both endpoints with a shared secret. The **ClearToken** field of the **CryptoToken** fields shall contain a **dhkey**, used to pass the parameters as specified in this Recommendation. **halfkey** contains the random public key of one party, **modsize** contains the DH-prime and **generator** contains the DH-group. The DH parameters to be used are indicated in Table D.4. For more details, please refer to [RFC 2412, Appendix E2].

NOTE – Since the H.225.0 messages are authenticated (as described earlier by procedure I), the DH exchange is an authenticated one.

In either direction with a H.225.0 call signalling message carrying a Diffie-Hellman half-key, when identification information is available, the caller or callee, when being registered, shall also include a separate end-to-end **ClearToken** with **sendersID** set to the endpoint identifier of the sender and **tokenOID** set to "E". Any intermediate H.323 signalling entity shall forward that particular end-to-end token unmodified.

The FastStart structures hold the offered open logical channels with the proposed security capabilities. Both H235Cap and nonH235Cap channels should be offered. During the H.245 Cap exchange, endpoints present **H235SecurityCapability** entries for the codecs that they support. Each codec is associated with a separate H.235 security capability. According to Annex D, these capabilities should indicate support for 128-bit AES-CBC (OID – "Z3"), 56-bit RC2-compatible-CBC (OID – "X"), should indicate support for 56-bit DES-CBC (OID – "Y") and may indicate support for 168-bit triple-DES-CBC (OID – "Z"), or 168-bit triple-DES-EOFB (OID – "Z1"), RC2-compatible-EOFB (OID – "X1"), DES-EOFB (OID – "Y1") or AES-EOFB (OID – "Z2"), see also Table D.6.

OpenLogicalChannel conveys both **forwardLogicalChannelParameters** and **reverseLogicalChannelParameters** with **dataType** providing **h235Media** with **encryptionAuthenticationAndIntegrity** where in the **encryptionCapability** at most one **MediaEncryptionAlgorithm** shall be present.

For the security relationship's purpose, the callee is the *a priori* master, see also 8.4.

The caller should set **mediaWaitForConnect** to true, to ascertain that session key material is available and received encrypted media can be decrypted. In scenarios, where "early media" is desired, such that the callee transmits encrypted or non-encrypted media simultaneously with sending the response message and encryption key material, the caller should be prepared not to be able to decrypt the contents unless key material is available.

NOTE – In this case, if the callee sends encrypted media to the caller (which theoretically it may do, because it has the caller's RTP/RTCP addresses), the caller will not be able to decipher it without the shared secret provided in the (Alerting, Call Proceeding) Connect message.

Procedures of the callee

During FastStart, the callee presents its DH token (see also 8.8) and the accepted FastStart structures. In case the Diffie-Hellman procedure is applied, it is recommended that the callee returns its DH token as part of the response message at the earliest opportunity; i.e., in the response message immediately following the SETUP. This allows the caller to compute the master key from the DH shared secret and to be prepared for receiving the session key and encrypted media.

NOTE 1 – In case there is no encryption algorithm available at both sides, the media stream may be left unencrypted or the connection may be aborted, depending on the security policy.

Each entity shall take appropriate least significant bits from the common shared Diffie-Hellman secret for the key encryption key (master key); i.e., the 56 least significant bits of the Diffie-Hellman secret for OID "X", OID "X1", OID "Y1" or OID "Y" and the 168 least significant bits of the Diffie-Hellman secret for OID "Z", OID "Z1" or OID "Z2" and the 128 least significant bits of the Diffie-Hellman secret for OID "Z3" or OID "Z2", see also Table D.6.

OpenLogicalChannel(Ack) responses are issued with the (master) created session key included in the **encryptionSync** field. This **encryptionSync** holds the session key for the directed logical channel from caller to callee. Key transport shall proceed according to the procedure described in B.2.4, using either **KeySyncMaterial** or **V3KeySyncMaterial** (see B.2.4.1). The session key shall be encrypted with the DH shared secret in a manner described below.

NOTE 2 – There is no prescribed method for generating the session keys, which are utilized to encrypt the media. The generation of these values is an implementation matter affected by local resources, policy, and the encryption algorithm to be used. Care should be taken to avoid generation of weak keys.

Using the procedure of B.2.4, the encrypted session key shall be carried in the **H.235Key/sharedSecret** within the **encryptionSync** field. The session key shall be carried in the **keyMaterial** field of the **KeySyncMaterial** – if not a multiple of the block size – shall be padded to a multiple of blocks before encryption. The value of the pad should be determined by the normal convention of the cipher algorithm. The (padded) **KeySyncMaterial** shall be encrypted using:

– 56 bits of the shared secret, starting with the least significant bits from the Diffie-Hellman secret for OID "X", OID "X1" , OID "Y1" or OID "Y";

– all the bits of the shared secret for OID "Z2", OID "Z" or OID "Z1" starting with the least significant bits from the DH secret.

Alternatively and preferably, the improved key transport according to B.2.4.1 should be used when possible due to the outcome of the version 3 indicating procedure (see B.2.3).

In case a full duplex secured media channel out of two unidirectional channels is to be established using fast start, the callee shall open a second logical channel towards the caller. This logical channel shall be signalled in a separate fastStart element. Using the available DH shared secret as

master key, the callee includes a different session key for that logical channel within **encryptionSync**.

8.6.1.1 Using multiple encryption algorithms in fast connect

The negotiation of media encryption as part of fast connect procedures leads to an inefficient expansion of the number of **OpenLogicalChannel** elements in the **fastConnect** element of a SETUP message. This occurs because a separate **OLC** is required for each combination of codec (**dataType**) and encryption algorithm (including "none").

The encryption algorithm to be applied to a media stream is specified through inclusion of the **dataType.h235Media.encryptionAuthenticationAndIntegrity.encryptionCapability dataType** in the **OLC**. H.235v2 practice is to include only a single **MediaEncryptionAlgorithm** in the **encryptionCapability**, although the latter element is defined as a sequence of the former elements. This procedure permits the inclusion of a preference-ordered sequence of encryption capabilities in each offered **OLC**. The receiver of the **OLC** shall then select a single algorithm from among those offered, and shall return the **OLC** with only the selected algorithm present (along with the appropriate transport addresses and encryption key information).

In order to provide the maximum efficiency, the Object ID "NULL-ENCR" (see Table 1) represents the "null" encryption algorithm which means that no encryption operation is to take place. Using this particular method requires only one **OLC** per offered codec per direction.

Procedure for the caller (see 8.1.7.1/H.323)

If an offered **dataType** element specifies encryption via the **h235Media** choice, the included **encryptionAuthenticationAndIntegrity** element may include an **encryptionCapability** element containing multiple encryption algorithms (including the NULL algorithm). This construct shall be taken to offer a choice of any one of the specified algorithms for encryption of the associated media capability.

Procedure for the callee (see 8.1.7.1/H.323)

If multiple encryption algorithms are offered for a channel, the called endpoint must select one and modify the **OpenLogicalChannel** to remove the others.

Table 1/H.235 – Object identifier for NULL encryption

Object identifier reference	Object identifier value	Description
"NULL-ENCR"	{itu-t (0) recommendation (0) h (8) 235 version (0) 3 26}	Indicates the "NULL encryption algorithm"

8.6.2 Bidirectional fast start security

Security for bidirectional T.120 data channels is for further study.

8.7 Encrypted H.245 DTMF

Endpoints may choose to send encrypted DTMF signals to achieve confidentiality. Using the session encryption key, endpoints may encrypt the DTMF signals in **UserInputIndication** as:

- Encrypted basic string: **encryptedAlphanumeric**;
- Encrypted iA5 string: **encryptedSignalType** within **signal**;
- Encrypted general string: **encryptedAlphanumeric** within **extendedAlphanumeric**.

NOTE 1 – The additional parameters for RTP in the iA5 string with timestamps and logical channel numbers or the signal update with the tone duration are not encrypted, as they are considered not to convey sensitive information.

The negotiated capability **secureDTMF** relates to an encrypted iA5 string.

The key management as specified by Annex D.7 should be applied to yield a session encryption key. That session encryption key shall be used to encrypt the H.245 DTMF signals.

NOTE 2 – This does not necessarily imply that the session key should be applied for RTP payload encryption as well.

However, when also using the DTMF via RTP by setting the **rtpPayloadIndication** flag, it is highly recommended that the RTP payload be secured using the voice encryption profile of Annex D.7.

Table 2 provides the available encryption algorithms (DES, 3DES or AES) which should deploy EOFB (incl. OFB as a special case, see B.2.5). To avoid potential padding of the DTMF characters, CBC, CFB or other block chaining modes that may make padding necessary, are not recommended for encryption of DTMF signals.

8.7.1 Encrypted basic string

If **encryptedBasicString** in **UserInputCapability** has been selected, then **encryptedAlphanumeric** shall indicate the applied encryption algorithm within **algorithmOID**, **paramS** holds the initial value for the encryption operation. The encrypted alphanumeric string shall be placed in **encrypted**.

8.7.2 Encrypted iA5 string

If **encryptedIA5String** in **UserInputCapability** has been selected, then **encryptedSignalType** shall hold the encrypted **ClearSignalType** where **sig** carries the plaintext **signalType** character. **signalType** shall hold a dummy "!" which shall be discarded by the recipient.

algorithmOID shall indicate the applied encryption algorithm, **paramS** holds the initial value for the encryption operation.

8.7.3 Encrypted general string

If **encryptedGeneralString** in **UserInputCapability** has been selected, then **encryptedAlphanumeric** within **extendedAlphanumeric** shall indicate the applied encryption algorithm within **algorithmOID**, while **alphanumeric** shall hold an empty string and **paramS** holds the initial value for the encryption operation.

8.7.4 List of object identifiers

Table 2/H.235 – Object identifiers for H.245 DTMF encryption

Object identifier reference	Object identifier value	Description
"DES-EOFB-DTMF"	{itu-t (0) recommendation (0) h (8) 235 version (0) 3 12}	H.245 DTMF encryption with DES-56 in EOFB mode
"3DES-EOFB-DTMF"	{itu-t (0) recommendation (0) h (8) 235 version (0) 3 13}	H.245 DTMF encryption with 3DES-168 in EOFB mode
"AES-EOFB-DTMF"	{itu-t (0) recommendation (0) h (8) 235 version (0) 3 14}	H.245 DTMF encryption with AES-128 in EOFB mode

8.8 Diffie-Hellman operation

This Recommendation supports the Diffie-Hellman protocol for end-to-end key agreement. Depending on the situation, the negotiated Diffie-Hellman key may act as master key (Annex D.7) or as a dynamic session key (Annex F and ITU-T Rec. H.530).

The Diffie-Hellman system is characterized by the system parameters g and p where p shall be a large prime and g denotes the generator of the multiplicative group modulo p or of a strong subgroup modulo p. g^x mod p denotes the (public) Diffie-Hellman half-key of the caller while g^y mod p denotes the (public) Diffie-Hellman half-key of the callee. RFC 2412 provides further background information and advice how to choose secure Diffie-Hellman parameters.

ITU-T Rec. H.235 conveys a Diffie-Hellman instance (g, p, g^x) encoded within a **ClearToken** where **dhkey** holds the **halfkey** g^x mod p (resp. g^y mod p) for some secret random x *(*resp. *y)*, the prime p in **modsize** and the **generator** g. A special case is the triplet (0, 0, 0) or an empty **dhkey** that does not represent any DH-instance but shall be used in signalling that the voice encryption profile is not being used.

Often, the DH-system parameters p and g are fixed for a set of applications with well-defined values, yet end systems may also choose their own set of parameters. The callee should be aware of the fact that non-standard DH-parameters may provide less security than the parameters look alike at first sight; e.g., the caller might have chosen a non-prime, or g generates just a smaller subgroup. While extensive parameter testing is unfeasible in practice, it is up to the security policy of the callee whether to accept or reject such offers.

For the fixed DH system parameters, a shorthand characterization through an object identifier may yield more compact encoded messages than including literal values. A **ClearToken** that carries a DH-instance with fixed, standardized DH parameters, may reference the DH instance with a DH-OID in the **tokenOID** field; unless the **tokenOID** is used for other purposes (such as in D.6.3.2 for a distinguished **CryptoToken**). The sender may additionally include the literal DH values but need not do so.

In case several DH-instances are to be indicated each through a DH-OID, the DH-parameters in a distinguished **CryptoToken** (which is being occupied by Annex D) shall be omitted by leaving **dhkey** absent, and all DH-instances shall then be carried within separate **ClearTokens** where the **tokenOID** holds the DH-OID, and **dhkey** may be left absent; any other fields within that **ClearToken** shall not be used.

NOTE 1 – This does not rule out the possibility to convey a DH instance in a distinguished **CryptoToken** or other available **ClearTokens** by literally including the DH parameter values.

In case a non-standard DH-instance is to be indicated, the DH-OID "DHdummy" shall be used and the non-standard DH-group parameters shall be explicitly provided in the **ClearToken**.

The caller may submit one or several **ClearTokens** each conveying a different Diffie-Hellman instance. The caller is encouraged to provide as many DH instances as possible as his/her security policy permits. This allows the callee to choose an appropriate instance for the response, thereby increasing the likelihood of finding a successful common parameter set.

The callee shall select and accept a single DH instance (if at all) that it chooses from the unordered set of DH instances provided by the caller in the SETUP message. In case the callee is able to select a DH instance that matches his/her own security needs, the callee shall not modify a proposed DH instance or return one that was not sent by the caller. The strength of the encryption algorithms available to both EPs during the call should correspond to the strength provided by the chosen DH instance that is returned by the callee; see Table D.4. The callee shall indicate the chosen DH instance in the response message.

In case the callee rejects any of the proposals for security reasons or due to lack of processing capabilities, the callee shall leave **dhkey** absent in the response message.

The callee shall include its DH token in the **Setup**-to-**Connect** response. The callee may include its DH token in the immediate response message following SETUP, or may include the DH token at some later stage, but at latest in the CONNECT message.

NOTE 2 – There are several aspects to be taken into account as to when the callee may include the DH token(s) during the **Setup**-to-**Connect** responses: the response time, the processing load upon the callee, capability of early media and other aspects. These issues are considered implementation dependent.

For some reasons, however, certain routing GKs may not deliver all **Setup**-to-**Connect** responses to the caller. Thus, one or more H.225.0 call signalling response messages, including a possible DH token, may be dropped and would not arrive at the caller. The caller would then be unable to compute the DH master key and media session key(s). To prevent such cases, the callee should always include the same DH token in each **Setup**-to-**Connect** response message.

In cases where the DH-OID indicates a different DH-instance than is actually being conveyed within **modsize** and **generator**, the literal values conveyed within **modsize** and **generator** shall take precedence over the DH-OID in the token. For the response, the callee should replace the conflicting DH-OID with the static DH-OID, e.g., "DH1024," that corresponds to the **modsize** and **generator** or "DHdummy" if there is no corresponding DH-OID.

9 Multipoint procedures

9.1 Authentication

Authentication shall occur between an endpoint and the MC(U) in the same manner that it would in a point-to-point conference. The MC(U) shall set the policy concerning level and stringency of authentication. As stated in 6.6, the MC(U) is trusted; existing endpoints in a conference may be limited by the authentication level employed by the MC(U). New **ConferenceRequest**/**ConferenceResponse** commands allow endpoints to obtain the certificates of other participants in the conference from the MC(U). As outlined in H.245 procedures, endpoints in a multipoint conference may request other endpoint certificates via the MC, but may not be able to perform direct cryptographic authentication within the H.245 channel.

9.2 Privacy

MC(U) shall win all master/slave exchanges and, as such, shall supply encryption key(s) to participants in a multipoint conference. Privacy for individual sources within a common session (assuming multicast) may be achieved with individual or common keys. These two modes may be arbitrarily chosen by the MC(U) and shall not be controllable from any particular endpoint except in modes allowed by MC(U) policy. In other words, a common key may be used across multiple logical channels as opened from different sources.

10 Authentication signalling and procedures

10.1 Introduction

Authentication is, in general, based either on using a shared secret (you are authenticated properly if you know the secret) or on public key-based methods with certifications (you prove your identity by possessing the correct private key). A shared secret and the subsequent use of symmetric cryptography requires a prior contact between the communicating entities. A prior face-to-face or secure contact can be replaced by generating or exchanging the shared secret key with methods based on public key cryptography, e.g., by Diffie-Hellman key exchange. The communication parties in the key generation and exchange have to be authenticated, for example, by using digitally signed messages; otherwise the communication parties cannot be sure with whom they share the secret.

This Recommendation presents authentication methods based on subscription, i.e., there must be a prior contact for sharing a secret, and authentication methods where public key cryptography is directly used in authentication, or it is used for generating the shared secret.

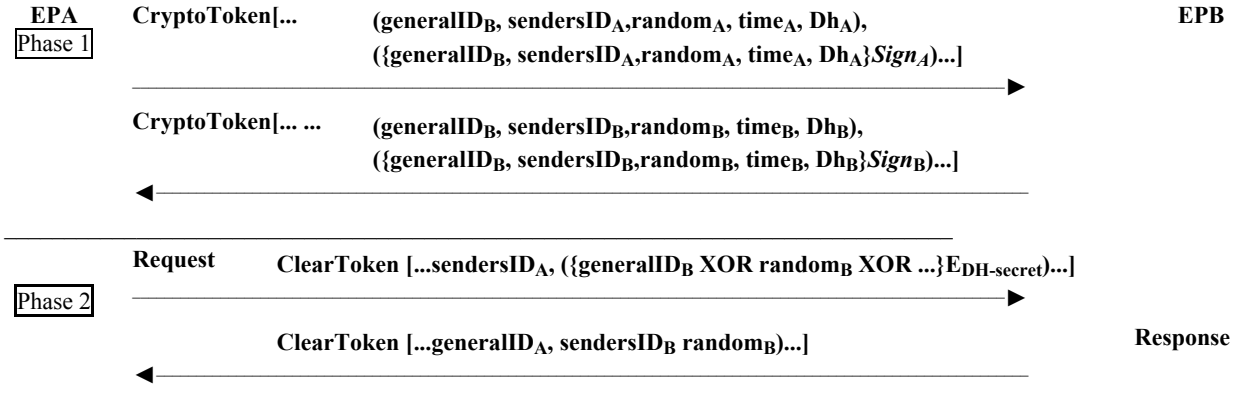
10.2 Diffie-Hellman with optional authentication

The intent is not to provide absolute, user-level authentication. This method provides signalling to generate a shared secret between two entities which may lead to keying material for private communications.

At the end of this exchange, both the entities will possess a shared secret key along with a chosen algorithm with which to utilize this key. This shared secret key may now be used on any subsequent request/response exchanges. It should be noted that in rare cases, the Diffie-Hellman exchange may generate known *weak* keys for particular algorithms. When this is the case, either entity should disconnect and reconnect to establish a new key set.

The first phase of Figure 1 demonstrates the data exchanged during the Diffie-Hellman. The second phase allows for application- or protocol-specific request messages to be authenticated by the responder. Note that a new random value may be returned with each response.

NOTE – If the messages are exchanged over an insecure channel, then digital signatures (or other message origin authentication method) must be used in order to authenticate the parties between whom the secret will be shared. An optional signature element may also be provided; these are illustrated in ***italics*** below.



[... ...] indicates a sequence of tokens.

() indicates a particular token, which may contain multiple elements.

{}E$_{DH\text{-}secret}$ indicates the contained values are encrypted utilizing the Diffie-Hellman secret.

EPB knows which shared secret key to use to decipher the **generalID$_B$** identifier by associating it with the **generalID$_A$**, which should also be passed in the message as **sendersID$_A$**. Note that the encrypted value in phase 2 is passed in the **generalID** field of a **clearToken** to simplify encoding.

Figure 1/H.235 – Diffie-Hellman with optional authentication

10.3 Subscription-based authentication

10.3.1 Introduction

Although the procedures outlined here (and the ISO algorithms from which they are derived) are bidirectional in nature, they may be utilized in only one direction if authentication is only needed in that direction. Both two-pass and three-pass procedures are described. The mutual two-pass authentication may be done only in one direction when the messages originating from the reverse direction need not be authenticated. These exchanges assume that each end possesses some well-known identifier (such as a text identifier) which uniquely identifies it. For the two-pass procedure, the further assumption is made that there is a mutually acceptable reference to time

(from which to derive timestamps). The amount of time skew that is acceptable is a local implementation matter. The three-pass procedure uses a randomly-generated, unpredictable challenge number (which may be augmented by a sequential counter 'random') as a challenge from the authenticator. This random number is intended to protect against replay attacks. Different to the two-pass procedures, the three-pass procedures do not authenticate the first, initial message holding the initiator's challenge.

There are three different variations that may be implemented depending on requirements:

1) password-based with symmetric encryption;

2) password-based with hashing;

3) certificate-based with signatures.

In all cases, the token will contain the information as described in the following clauses depending on the variation chosen. Note that, in all cases, the **generalID** may be known through configuration or directory lookup rather than in band protocol exchange. To simplify processing at the receiver, the sender should include its identity within **sendersID** and set the **generalID** to the identification of the recipient.

NOTE 1 – In all cases where timestamps are generated and passed as part of a security exchange, implementors should take the following precautions. The timestamp granularity should be fine enough that it is guaranteed to increment with each message. If this is not guaranteed, replay attacks are possible. (e.g., if the timestamp only increments by the minute, then an endpoint "C" can spoof endpoint "A" within duration of one minute after endpoint "A" has sent a message to endpoint "B").

NOTE 2 – If the message is multicast, then the message is not secured.

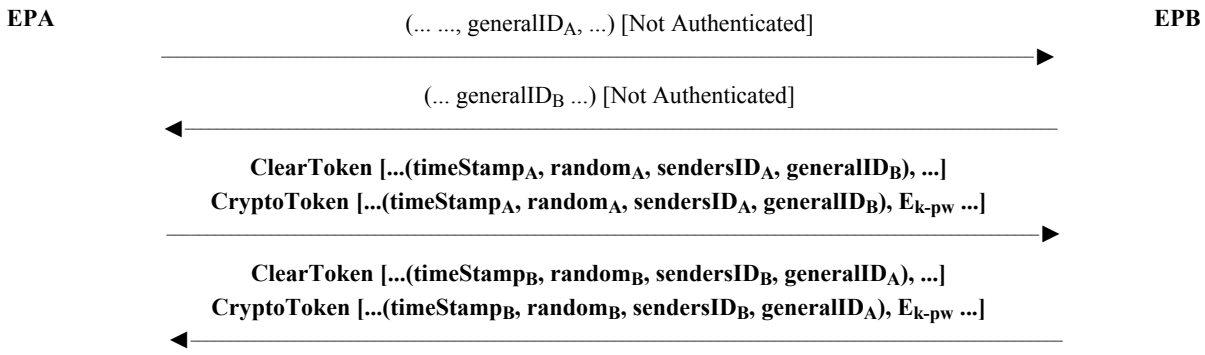
10.3.2 Password with symmetric encryption

Figures 2a and 2b show the token format and the message exchange required to perform this type of authentication in two passes or three passes, respectively. This protocol is based on 5.2.1 (two-pass) and 5.2.2 (three-pass) of ISO/IEC 9798-2; it is assumed that an identifier and associated password are exchanged during subscription. The encryption key is length N octets (as indicated by the AlgorithmID), and is formed as follows:

– If password length = N, Key = password;

– if password length < N, the key is padded with zeros;

– if password length > N, the first N octets are assigned to the key, then the N + M*th* octet of the password is XOR'd to the Mmod(N)*th* octet (for all octets beyond N) (i.e., all "extra" password octets are repeatedly folded back on the key by XORing).



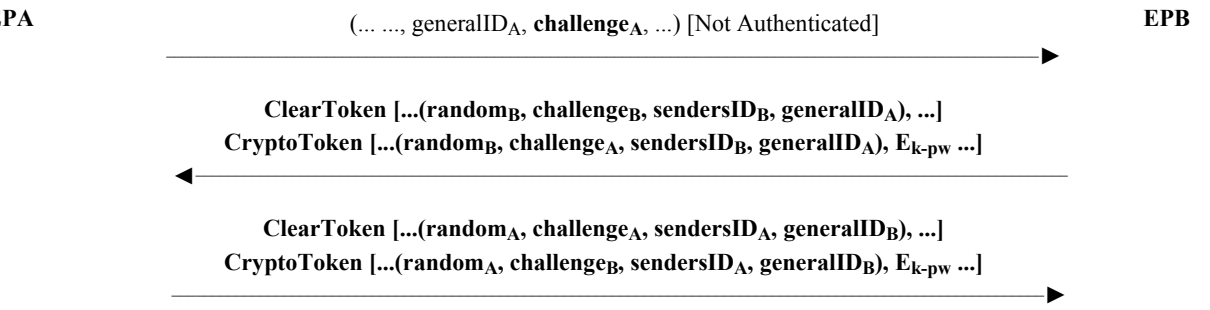
NOTE 1 – The return token from EPB is optional; if omitted, only one-way authentication is achieved.

NOTE 2 – **E$_{k-pw}$** indicates values that are encrypted using the key "k" derived from the password "pw".

NOTE 3 – **random** is a monotonically increasing counter making multiple message with the same timestamp unique.

NOTE 4 – In the third message, EPA provides a separate **ClearToken** that is identified through as same OID as the OID in the **CryptoToken**; similarly for the fourth message and vice versa.

Figure 2a/H.235 – Password with symmetric encryption; two passes



NOTE 1 – **challenge$_A$** and the return encrypted **CryptoToken** from B to A are not necessary if one-way authentication is desired.

NOTE 2 – **E$_{k-pw}$** indicates an encryption function that is encrypted using the key "k" derived from the password "pw".

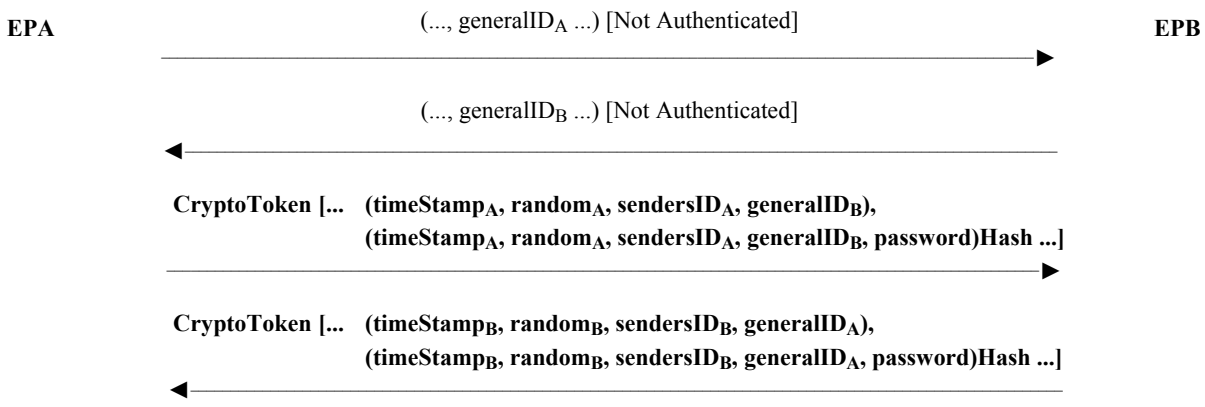
NOTE 3 – In the third message, EPA provides a a new **challenge$_A$** in plaintext in a separate **ClearToken**, that is identified through the same OID as the OID in the **CryptoToken**. EPA also returns the encrypted **challenge$_B$** as response; similarly for the second message and vice versa.

NOTE 4 – For multiple outstanding messages, **random** (i.e., a monotonically increasing counter) shall make a challenge unique.

Figure 2b/H.235 – Password with symmetric encryption; three passes

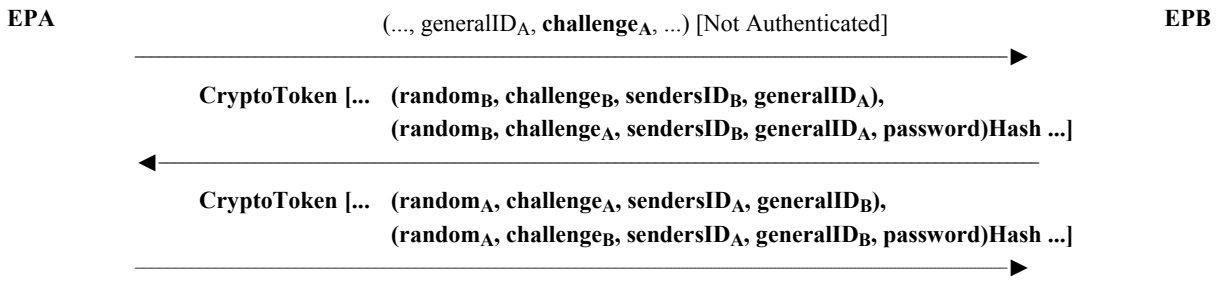
10.3.3 Password with hashing

Figures 3a and 3b show the token format and the message exchange required to perform this type of authentication for two pass or three passes, respectively. This protocol is based on 5.2.1 and 5.2.2 of ISO/IEC 9798-4; it is assumed that an identifier and associated password are exchanged during subscription. Annex D provides detailed description of the two-pass hashing procedure.



NOTE 1 – The return token from EPB is optional; if omitted, only one-way authentication is achieved.
NOTE 2 – **Hash** indicates a hashing function that operates on the contained values.
NOTE 3 – **random** is a monotonically increasing counter making multiple messages with the same timestamp unique.

Figure 3a/H.235 – Password with hashing; two passes



NOTE 1 – The return token from EPB is optional; if omitted, only one-way authentication is achieved.
NOTE 2 – **Hash** indicates a hashing function that operates on the contained values.
NOTE 3 – In the third message, EPA provides a new **challenge$_A$** in plaintext within the embedded **ClearToken** in **cryptoHashedToken**. EPA also returns the hashed **challenge$_B$** as response; similarly for the second message and vice versa.
NOTE 4 – For multiple outstanding messages, **random** (i.e., a monotonically increasing counter) shall make a challenge unique.

Figure 3b/H.235 – Password with hashing; three passes

NOTE 1 – The **cryptoHashedToken** structure is used to pass the parameters used in this exchange. Included in this structure are the 'clear' versions of parameters needed to compute the hashed value. Implementors shall include the timestamp in the **hashedVals** and shall *not* include the password. (For example, both the password and the '**generalID**' should be known *a priori* by the recipient; the former may be omitted.)

NOTE 2 – The hashing function shall be applied to the **EncodedGeneralToken** structure that includes at least the ID, timestamp and password fields. The password value shall NOT be passed in the **ClearToken**.

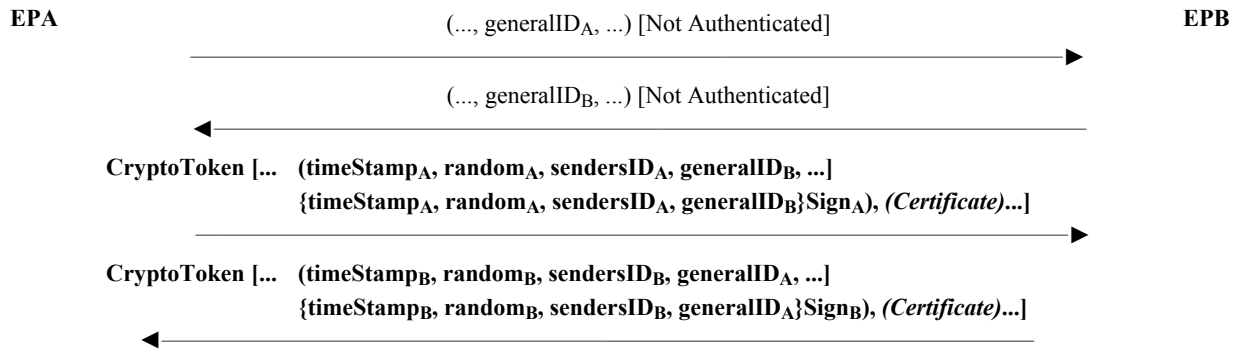
NOTE 3 – Implementations should ensure that user-entered passwords convey sufficient entropy. Passwords that are too short or that are susceptible to dictionary attacks should be rejected. Feeding the user-entered pass-phrase through a cryptographic hash function and using the output bits may be advantageous in certain cases.

10.3.4 Certificate-based with signatures

Figures 4a and 4b show the token format and the message exchange required to perform this type of authentication. This protocol is based on 5.2.1 of ISO/IEC 9798-3; it is assumed that an identifier and associated certificate are assigned/exchanged during subscription. Annex E provides detailed description of the two-pass signature procedure.

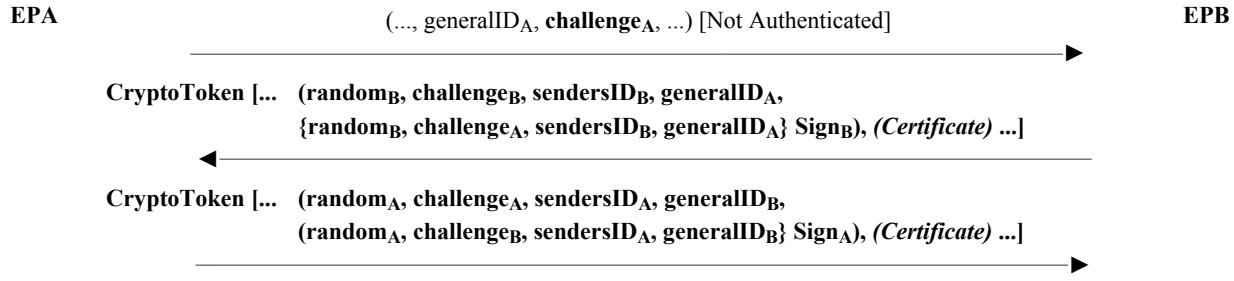
NOTE 1 – An optional certificate element may also be provided; these are illustrated in ***italics*** below.

NOTE 2 – If the message is multicast, then the identifier of the destination (**generalID$_B$** for messages originated at A and vice versa) should not be included in the **ClearToken**.



NOTE 1 – The return token from EPB is optional; if omitted, only one-way authentication is achieved.
NOTE 2 – A "payment" type certificate may be optionally included by the EPA originator.
NOTE 3 – **Sign** indicates a signing function (from associated certificate) performed on the contained values.
NOTE 4 – **random** is a monotonically increasing counter making multiple messages with the same timestamp.

Figure 4a/H.235 – Certificate-based with signatures; two passes



NOTE 1 – The return token from EPB is optional; if omitted, only one-way authentication is achieved.
NOTE 2 – A "payment" type certificate may be optionally included by the EPA originator.
NOTE 3 – **Sign** indicates a signing function (from associated certificate) performed on the contained values.
NOTE 4 – In the third message, EPA provides a new **challenge$_A$** in plaintext within the embedded encoded **GeneralToken**. EPA also returns the signed **challenge$_B$** as response; similarly for the second message and vice versa.
NOTE 5 – For multiple outstanding messages, **random** (i.e., a monotonically increasing counter) shall make a challenge unique.

Figure 4b/H.235 – Certificate-based with signatures; three passes

10.3.5 Usage of shared secret and passwords

This Recommendation applies certain symmetric cryptographic techniques for the purpose of authentication, integrity and confidentiality. This text uses the term password and shared secret 21

when applying symmetric techniques. Shared secret is understood as the generic term identifying an arbitrary bit string. The shared secret may be assigned or configured as part of the user's

subscription process, or may be part of in-band computation such as a Diffie-Hellman-derived shared secret.

A password could be viewed as an alphanumeric character string that users can memorize. It is obvious that using passwords should be done with care. Passwords are able to provide sufficient security only when they are chosen randomly from a large space, when they convey sufficient entropy such that they are unpredictable and when they are changed periodically. Rules for setting up and maintaining passwords do not fall within the scope of this Recommendation.

A good practice as to how to deploy the benefits from passwords and shared secrets is to transform the user password string into a fixed bit string as the shared secret using a cryptographically strong one-way hash function.

As a recommended example, when using the security profile of Annex D, the SHA1 when applied to the password string, yields to a 20-byte shared secret. An advantage is that the hashed result does not only conceal the actual password, but also defines a fixed length bit string format without really sacrificing entropy.

Thus,

shared secret := SHA1 (password)

11 Media stream encryption procedures

Media streams shall be encoded using the algorithm and key as presented in the H.245 channel. Figures 5 and 6 show the general flow. Note that the transport header is attached to the transport SDU after the SDU has been encrypted. The opaque segments indicate privacy. As new keys are received by the transmitter and used in the encryption, the SDU header shall indicate in some manner to the receiver that the new key is now in use. For example, in ITU-T Rec. H.323, the RTP header (SDU) will change its payload type to indicate the switch to the new key.

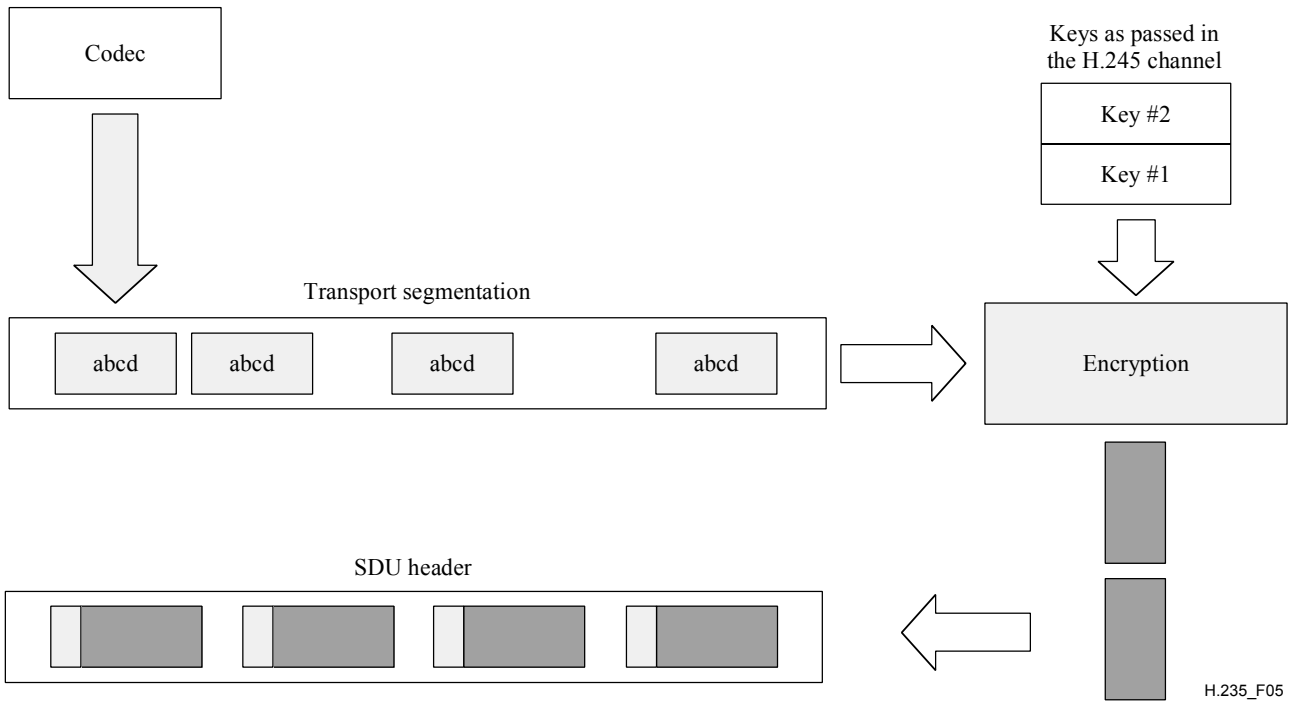


Figure 5/H.235 – Encryption of media

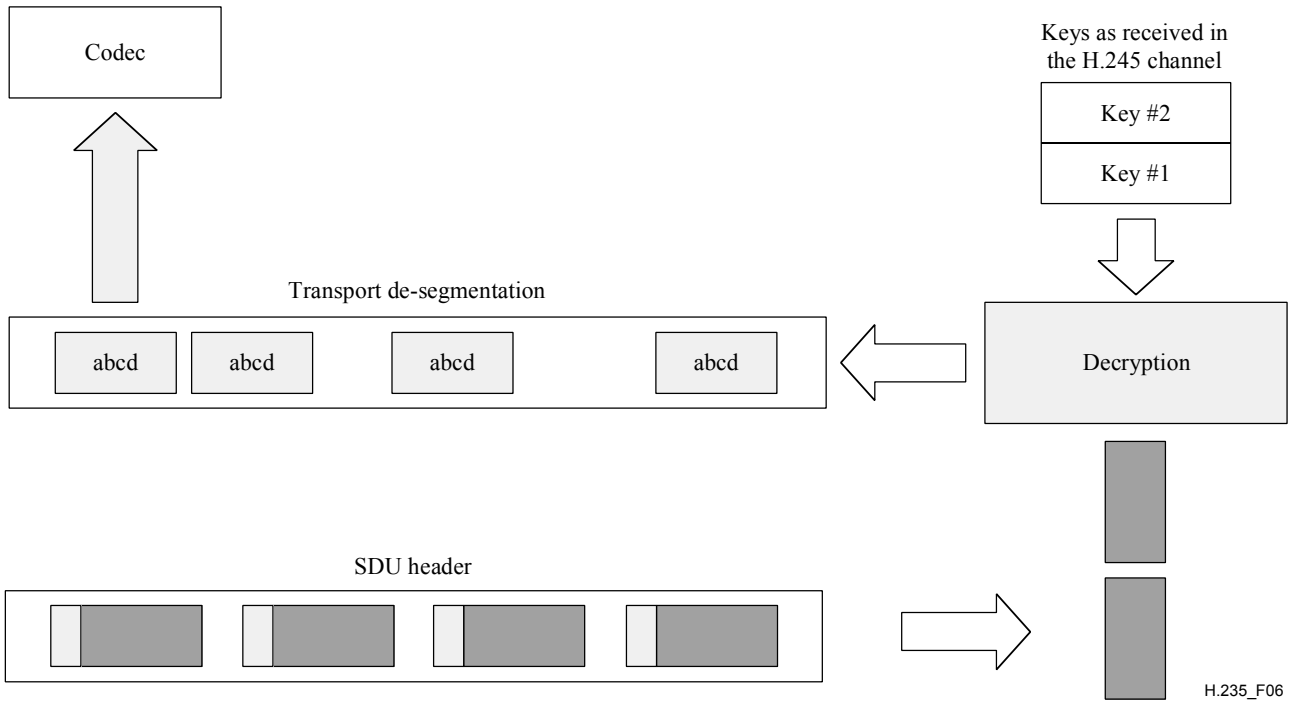


Figure 6/H.235 – Decryption of media

11.1 Media session keys

Included in the **encryptionUpdate** is the **h235Key**. The **h235Key** is ASN.1 encoded within the context of the H.235 ASN.1 tree, and passed as an opaque octet string with respect to H.245. The key may be protected by utilizing one of the three possible mechanisms as they are passed between two endpoints.

- If the H.245 channel is secure, no additional protection is applied to the key material. The key is passed "in the clear" with respect to this field; the ASN.1 choice of **secureChannel** is utilized.
- If a secret key and algorithm has been established outside the H.245 channel as a whole (i.e., outside H.323 or on an **h235Control** logical channel), the shared secret is used to encrypt the key material; the resultant enciphered key is included here. In this case, the ASN.1 choice of **sharedSecret** is used.
- Certificates may be used when the H.245 channel is not secure, but may also be used in addition to the secure H.245 channel. When certificates are utilized, the key material is enciphered using the certificate's public key and the ASN.1 construct **certProtectedKey**.

At any point in a conference, a receiver (or transmitter) may request a new key (**encryptionUpdateRequest**). One reason it might do this is if it suspects that it has lost synchronization of one of the logical channels. The master receiving this request shall generate new key(s) in response to this command. The master may also decide asynchronously to distribute new key(s), if so, it shall use the **encryptionUpdate** message.

After receiving an **encryptionUpdateRequest**, a master shall send out **encryptionUpdate**. If the conference is a multipoint one, the MC (also the master) should distribute the new key to all receivers before it gives this key to the transmitter. The transmitter of the data on the logical channel shall utilize the new key at the earliest possible time after receiving the message.

A transmitter (assuming it is not the master) may also request a new key. If the transmitter is part of a multipoint conference, the procedure shall be as follows:

- The transmitter shall send the **encryptionUpdateRequest** to the MC (master).

- The MC should generate a new key(s) and send an **encryptionUpdate** message to all conference participants except the transmitter.
- After distributing the new keys to all other participants, the MC shall send the **encryptionUpdate** to the transmitter. The transmitter shall then utilize the new key.

11.2 Media anti-spamming

The receiver of an RTP media stream may wish to counter denial-of-service and flooding attacks on discovered RTP/UDP ports. Receivers, when having implemented the anti-spam capability, can quickly determine whether an obtained RTP packet stems from an unauthorized source and discard it.

The anti-spamming capability, when set, indicates use of the anti-spamming mechanism either

- for plaintext media data without media encryption (see case 1 below); or
- in combination with encrypted media data when **EncryptionCapability** features an encryption algorithm (see case 2 below).

Both options provide a lightweight **RTP packet authentication** on selected fields through a computed message authentication code (MAC). The MAC may be computed using the object identifiers defined in 11.2.1. The cryptographic algorithms are by:

- an encryption algorithm (e.g., DES in MAC mode see ISO/IEC 9797). DES-MAC is indicated using the OID "N" while triple-DES-MAC is indicated using OID "O"; or
- using a cryptographic one-way function (e.g., SHA1). The OID to be used is "M".

The MAC algorithm is indicated in the object identifier of **antiSpamAlgorithm**. The algorithm OID implicitly indicates also the size of the MAC; e.g., 1 block = 64 bits for DES MAC. In order to save bandwidth, the MAC could be truncated, albeit sacrificing some security, e.g., to a 32-bit MAC; this then requires a different object identifier. The anti-spam method is independent of any additional payload encryption (see cases 1 and 2 below).

Anti-spamming uses the following RTP packet format (see Figure 7) where the RTP padding sequence is interpreted as follows (see A.5/H.225.0).

- The P bit in the RTP header shall be set to 1.
- Padding bytes shall be appended at the end of the payload with the following meaning:

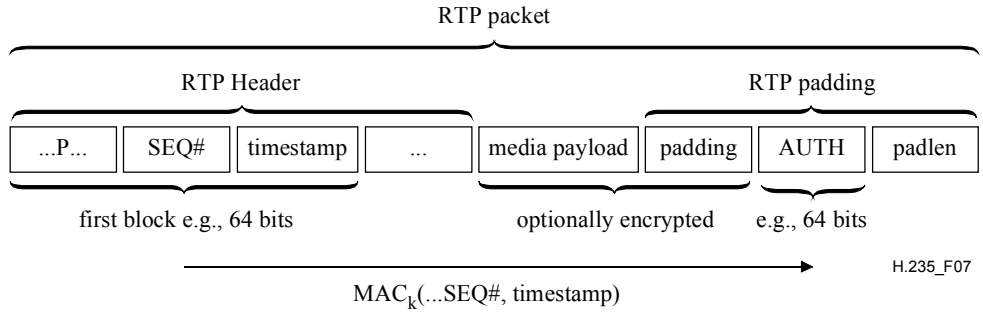


Figure 7/H.235 – RTP packet format for media anti-spamming

NOTE 1 – If anti-spamming is not used, then the AUTH and padlen fields are not used either and the usual RTP packet format applies.

1) *Case anti-spamming-only*

This case applies when the media data are not encrypted and the padding fields are left empty. The last octet of the RTP padding contains a count of how many padding octets

should be ignored at the end of the RTP packet. The other padding bytes carry the MAC. The MAC shall be computed over the first crypto block of the RTP header including the varying timestamp and sequence number using the negotiated MAC algorithm of **antiSpamAlgorithm** and applying the symmetric secret. A static or manually configured shared secret, or a dynamically negotiated shared secret *k* may be used according to the procedures of ITU-T Rec. H.235. For larger block sizes (more than 64 bits), some sufficient additional bits of the RTP header, or even the first media payload, shall be taken.

For the MAC computation, it is recommended to use the key that is obtained from the H.235 media session key distribution; although the session key applied is not used for payload encryption. Secure fast connect with key establishment (see Annex J/H.323) or manual keying may be used for key management. The sender computes the MAC as described above and includes the result in the MAC field in the RTP padding AUTH field. Sender and receiver know the size of the AUTH field and the length of the MAC by the **antiSpamAlgorithm**.

The MAC verification at the receiver side should be done as early as possible, if possible already within the RTP stack, or at latest, before decryption or decompressing the payload. The receiver first recomputes the MAC in the same way as the sender did and compares the computed MAC with the delivered MAC in the RTP padding. If the MACs do not match, the RTP header has been modified in transit or was sent by an unauthorized entity that does not possess the key. Thus, the mis-authenticated RTP packet shall be discarded, the event may be logged; this indicates a probable attempt of denial-of-service attack. Otherwise, the authenticated RTP packet can be processed further, the RTP padding is removed and the payload is fed through the codec.

NOTE 2 – The lightweight MAC computation/verification with DES encryption involves only a single encryption operation; alternatively, SHA1 MAC is computed on a short part of the packets of fixed length, thus the crypto operations consume absolutely minimal processing resources.

2) *Case anti-spam method and payload encryption*

This case applies when the media data are encrypted and the anti-spamming method is invoked. When the payload does not fall on even block boundaries, some additional padding bytes have to be appended to the payload in front of the MAC. The media payload encryption is according to this clause 11.

EncryptionCapability defines the payload encryption algorithm while **antiSpamAlgorithm** defines the anti-spamming method. For security reasons, the media encryption and the MAC shall use different session keys. The MAC key *k* is computed by feeding the encryption key *K* through the SHA1 one-way hash function;

k = SHA1(K); sufficient bits shall be taken from the hashed result in network byte order. When **antiSpamAlgorithm** indicates an encryption algorithm, then the collected bits shall be made a correct encryption key; e.g., setting DES parity bits.

After the receiver successfully verifies the authenticity of the RTP packet, the payload is decrypted and the RTP padding is then discarded. The general procedure is according to case 1 above.

11.2.1 List of object identifiers

Table 3 lists all the referenced OIDs.

Table 3/H.235 – Object identifiers used for anti-spamming

Object identifier reference	Object identifier value	Description
"M"	{itu-t (0) recommendation (0) h (8) 235 version (0) 2 8}	anti-spamming using HMAC-SHA1-96
"N"	{iso(1) identified-organization(3) oiw(14) secsig(3) algorithm(2) desMAC(10)}	anti-spamming using DES (56 bit) MAC (see ISO/IEC 9797) with 64-bit MAC
"O"	{iso(1) identified-organization(3) oiw(14) secsig(3) algorithm(2) desEDE(17)}	anti-spamming using triple-DES (168-bit) MAC (see ISO/IEC 9797)

12 Security error recovery

This Recommendation does not specify or recommend any methods by which endpoints may monitor their absolute privacy. It does, however, recommend actions to be taken when privacy loss is detected.

If either endpoint detects a breach in the security of the call connection channel (e.g., H.225.0 for H.323), it should immediately close the connection following the protocol procedures appropriate to the particular endpoint (for 8.5/H.323 with the exception of step B-5).

If either endpoint detects a breach in the security of the H.245 channel or the secured data (**h235Control**) logical channel, it should immediately close the connection following the protocol procedures appropriate to the particular endpoint (for 8.5/H.323 with the exception of step B-5).

If any endpoint detects a loss of privacy on one of the logical channels, it should immediately request a new key **(encryptionUpdateRequest)** and/or close the logical channel. At the discretion of the MC(U), a loss of privacy on one logical channel may cause all other logical channels to be closed and/or re-keyed at the discretion of the MC(U). MC(U) shall forward **encryptionUpdateRequest, encryptionUpdate** to any and all endpoints affected.

At the discretion of the MC(U), a security error on an individual channel may cause the connections to be closed on all of the conference endpoints, thus ending the conference.

13 Asymmetric authentication and key exchange using elliptic curve crypto systems

This Recommendation provides sophisticated elliptic curve techniques with applications to signature, key management and encryption. One of the primary advantages over "classical" asymmetric techniques such as RSA are:

- Shorter cryptographic keys yielding comparable security as RSA: Typical key lengths for elliptic curve crypto systems are 160 bits; i.e., equivalent in security to a 1024-bit RSA key. The shorter key consumes less memory for storage and makes elliptic curve crypto systems especially attractive for implementation in smart-cards, and in any other devices with low memory requirements. In the H.323 environment, Annex J/H.323-based secured audio simple endpoint types (SASETs) with their low price requirements are well-suited for deployment of elliptic curve techniques.
- Improved processing speed achieved both in software and in hardware implementations: The shorter keys contribute to the processing speed. This results in faster interactive (user) responses.

All the background information, explanation and processing procedures of elliptic curve cryptography can be found in (*ATM Security Specification Version 1.1*, section 8.7). It is

recommended to encode the elliptic points in their affine, uncompressed notation without using the point-compression/decompression method. Further information on this topic is available in ISO/IEC 15946-1 and ISO/IEC 15946-2.

13.1 Key management

Elliptic curve-based Diffie-Hellman key agreement schemes are similar to the classic mod-p case as defined in this Recommendation as well. There are two cases:

- elliptic curves over a prime field: **eckasdhp** holds the elliptic curve and Diffie-Hellman parameters;
- elliptic curves of characteristic 2: **eckasdh2** holds the elliptic curve and Diffie-Hellman parameters.

The ECKASDH structure holds either case. Some example elliptic curves are listed in ISO/IEC 15946-1. Any other suitable and appropriate elliptic curves could be used as well.

Due to the available sequenced structure of the **ClearToken** signalling, both **dhkey** and **eckasdhkey** should not occur at the same time; only one shall be present when the Diffie-Hellman key exchange is applied.

Remark – Do not confuse the randomly chosen secret parameters ***a*** by party A or ***b*** by party B with the common Weierstrass coefficients ***a, b***.

13.2 Digital signature

The **ECGDSASignature** field carries the values **r** and **s** of the computed elliptic curved-based digital signature. Section 8.7.3 of *ATM Security Specification Version 1.1* and chapter 5 of ISO 15946-2 provide further information on the signature algorithm EC-GDSA.

The elliptic curve-based digital signature **ECGDSA** shall be ASN.1 coded and then put into the **signature** field of the **SIGNED** macro of this Recommendation. For the digital signature, the sender shall include an object identifier into **algorithmOID** by which the recipient is able to determine usage of an elliptic curve digital signature.

Annex A

H.235 ASN.1

```
H235-SECURITY-MESSAGES DEFINITIONS AUTOMATIC TAGS ::=
BEGIN

-- EXPORTS All

ChallengeString         ::= OCTET STRING (SIZE(8..128))
TimeStamp               ::= INTEGER(1..4294967295)     -- seconds since 00:00
                                                       -- 1/1/1970 UTC
RandomVal               ::= INTEGER -- 32-bit Integer
Password                ::= BMPString (SIZE (1..128))
Identifier              ::= BMPString (SIZE (1..128))
KeyMaterial             ::= BIT STRING(SIZE(1..2048))

NonStandardParameter ::= SEQUENCE
{
     nonStandardIdentifier   OBJECT IDENTIFIER,
     data                    OCTET STRING
}
```

```
-- if local octet representations of these bit strings are used they shall
-- utilize standard Network Octet ordering (e.g., Big Endian)
DHset ::= SEQUENCE
{
     halfkey        BIT STRING (SIZE(0..2048)), -- = g^x mod n
     modSize        BIT STRING (SIZE(0..2048)), --  n
     generator      BIT STRING (SIZE(0..2048)), -- g
     ...
}

ECpoint ::= SEQUENCE -- uncompressed (x, y) affine coordinate representation of
                     -- an elliptic curve point
{
     x         BIT STRING (SIZE(0..511)) OPTIONAL,
     y         BIT STRING (SIZE(0..511)) OPTIONAL,
     ...
}

ECKASDH::= CHOICE -- parameters for elliptic curve key agreement scheme Diffie-
Hellman
{
     eckasdhp SEQUENCE -- parameters for elliptic curves of prime field
     {
          public-key    ECpoint, -- This field contains representation of
               -- the ECKAS-DHp public key value. This field contains the
               -- initiator's ECKAS-DHp public key value (aP) when this
               -- information element is sent from originator to receiver. This
               -- field contains the responder's ECKAS-DHp public key value (bP)
               -- when this information element is sent back from receiver to
               -- originator.
          modulus       BIT STRING (SIZE(0..511)), -- This field contains
               -- representation of the ECKAS-DHp public modulus value (p).
          base          ECpoint, -- This field contains representation of the
               -- ECKAS-DHp public base (P).
          weierstrassA  BIT STRING (SIZE(0..511)), -- This field contains
               -- representation of the ECKAS-DHp Weierstrass coefficient (a).
          weierstrassB  BIT STRING (SIZE(0..511)) -- This field contains
               -- representation of the ECKAS-DHp Weierstrass coefficient (b).
     },

     eckasdh2 SEQUENCE -- parameters for elliptic curves of characteristic 2
     {
          public-key    ECpoint, -- This field contains representation of
               -- the ECKAS-DH2 public key value.
               -- This field contains the initiator's ECKAS-DH2 public key value
               -- (aP) when this information element is sent from originator to
               -- receiver. This field contains the responder's ECKAS-DH2 public
               -- key value (bP) when this information element is sent back from
               -- receiver to originator.
          fieldSize     BIT STRING (SIZE(0..511)), -- This field contains
               -- representation of the ECKAS-DH2 field size value (m).
          base          ECpoint, -- This field contains representation of the
               -- ECKAS-DH2 public base (P).
          weierstrassA  BIT STRING (SIZE(0..511)), -- This field contains
               -- representation of the ECKAS-DH2 Weierstrass coefficient (a).
          weierstrassB  BIT STRING (SIZE(0..511)) -- This field contains
               -- representation of the ECKAS-DH2 Weierstrass coefficient (b).
     },
     ...
}
```

```
ECGDSASignature::= SEQUENCE -- parameters for elliptic curve digital signature
              -- algorithm
{
     r         BIT STRING (SIZE(0..511)), -- This field contains the
               -- representation of the r component of the ECGDSA digital
               -- signature.
     s         BIT STRING (SIZE(0..511)) -- This field contains the
               -- representation of the s component of the ECGDSA digital
               -- signature.
}

TypedCertificate ::= SEQUENCE
{
     type           OBJECT IDENTIFIER,
     certificate    OCTET STRING,
     ...
}

AuthenticationBES ::= CHOICE

{
     default        NULL, -- encrypted ClearToken
     radius         NULL, -- RADIUS-challenge/response
     ...

}

AuthenticationMechanism  ::= CHOICE
{
     dhExch         NULL, -- Diffie-Hellman
     pwdSymEnc      NULL, -- password with symmetric encryption
     pwdHash        NULL, -- password with hashing
     certSign       NULL, -- Certificate with signature
     ipsec          NULL, -- IPSEC based connection
     tls            NULL,
     nonStandard    NonStandardParameter, -- something else.
     ...,
     authenticationBES  AuthenticationBES -- user authentication for BES
}

ClearToken          ::= SEQUENCE  -- a "token" may contain multiple value types.
{
     tokenOID       OBJECT IDENTIFIER,
     timeStamp      TimeStamp OPTIONAL,
     password       Password OPTIONAL,
     dhkey          DHset OPTIONAL,
     challenge      ChallengeString OPTIONAL,
     random         RandomVal OPTIONAL,
     certificate    TypedCertificate OPTIONAL,
     generalID      Identifier OPTIONAL,
     nonStandard    NonStandardParameter OPTIONAL,
     ...,
     eckasdhkey     ECKASDH OPTIONAL,  -- elliptic curve Key Agreement
                                       -- Scheme-Diffie Hellman Analogue
                                       -- (ECKAS-DH)
     sendersID      Identifier OPTIONAL,
     h235Key        H235Key OPTIONAL -- central distributed key in V3
}

--   An object identifier should be placed in the tokenOID field when a
--   ClearToken is included directly in a message (as opposed to being
--   encrypted). In all other cases, an application should use the
--   object identifier { 0 0 } to indicate that the tokenOID value is not
--   present.
```

```
--   Start all the cryptographic parameterized types here...
--

SIGNED { ToBeSigned } ::= SEQUENCE {
     toBeSigned          ToBeSigned,
     algorithmOID        OBJECT IDENTIFIER,
     paramS              Params,   -- any "runtime" parameters
     signature           BIT STRING -- could be an RSA or an ASN.1 coded
ECGDSA Signature
} ( CONSTRAINED BY { -- Verify or Sign Certificate -- } )


ENCRYPTED { ToBeEncrypted } ::= SEQUENCE {
     algorithmOID        OBJECT IDENTIFIER,
     paramS              Params,   -- any "runtime" parameters
     encryptedData       OCTET STRING
} ( CONSTRAINED BY { -- Encrypt or Decrypt -- ToBeEncrypted } )

HASHED { ToBeHashed } ::= SEQUENCE {
     algorithmOID        OBJECT IDENTIFIER,
     paramS              Params,   -- any "runtime" parameters
     hash                BIT STRING
} ( CONSTRAINED BY { -- Hash -- ToBeHashed } )

IV8 ::= OCTET STRING (SIZE(8)) -- initial value for 64-bit block ciphers
IV16 ::= OCTET STRING (SIZE(16)) -- initial value for 128-bit block ciphers

-- signing algorithm used must select one of these types of parameters
-- needed by receiving end of signature.

Params ::= SEQUENCE {
     ranInt         INTEGER OPTIONAL, -- some integer value
     iv8            IV8 OPTIONAL, -- 8-octet initialization vector
     ...,
     iv16           IV16 OPTIONAL,-- 16-octet initialization vector
     iv             OCTET STRING OPTIONAL, -- arbitrary length initialization
     vector
     clearSalt      OCTET STRING OPTIONAL -- unencrypted salting key for
     encryption
}

EncodedGeneralToken ::= TYPE-IDENTIFIER.&Type (ClearToken -- general usage token
-- )
PwdCertToken ::= ClearToken (WITH COMPONENTS {..., timeStamp PRESENT, generalID
PRESENT})
EncodedPwdCertToken ::= TYPE-IDENTIFIER.&Type (PwdCertToken)

CryptoToken::= CHOICE
{

     cryptoEncryptedToken SEQUENCE -- General purpose/application specific token
     {
          tokenOID       OBJECT IDENTIFIER,
          token          ENCRYPTED { EncodedGeneralToken }
     },
     cryptoSignedToken  SEQUENCE -- General purpose/application specific token
     {
          tokenOID       OBJECT IDENTIFIER,
          token          SIGNED { EncodedGeneralToken }
     },
     cryptoHashedToken SEQUENCE -- General purpose/application specific token
```

```
    {
        tokenOID            OBJECT IDENTIFIER,
        hashedVals          ClearToken,
        token HASHED { EncodedGeneralToken }
    },
    cryptoPwdEncr ENCRYPTED { EncodedPwdCertToken },
    ...
}

-- These allow the passing of session keys within the H.245 OLC structure.
-- They are encoded as standalone ASN.1 and based as an OCTET STRING within
-- H.245
H235Key  ::=CHOICE  -- This is used with the H.245 or ClearToken "h235Key"
field
{
    secureChannel           KeyMaterial,
    sharedSecret            ENCRYPTED {EncodedKeySyncMaterial},
    certProtectedKey        SIGNED {EncodedKeySignedMaterial },
    ...,
    secureSharedSecret      V3KeySyncMaterial -- for H.235 V3 endpoints
}

KeySignedMaterial ::= SEQUENCE {
    generalId     Identifier, -- slave's alias
    mrandom       RandomVal, -- master's random value
    srandom       RandomVal OPTIONAL, -- slave's random value
    timeStamp     TimeStamp OPTIONAL, -- master's timestamp for unsolicited EU
    encrptval     ENCRYPTED { EncodedKeySyncMaterial }
}
EncodedKeySignedMaterial ::= TYPE-IDENTIFIER.&Type (KeySignedMaterial)

H235CertificateSignature    ::= SEQUENCE
{
    certificate             TypedCertificate,
    responseRandom          RandomVal,
    requesterRandom         RandomVal OPTIONAL,
    signature               SIGNED { EncodedReturnSig },
    ...
}

ReturnSig ::= SEQUENCE {
    generalId               Identifier, -- slave's alias
    responseRandom          RandomVal,
    requestRandom           RandomVal OPTIONAL,
    certificate             TypedCertificate OPTIONAL -- requested certificate
}

EncodedReturnSig ::= TYPE-IDENTIFIER.&Type (ReturnSig)
KeySyncMaterial    ::= SEQUENCE
{
    generalID     Identifier,
    keyMaterial   KeyMaterial,
    ...
}
EncodedKeySyncMaterial  ::=TYPE-IDENTIFIER.&Type (KeySyncMaterial)



V3KeySyncMaterial  ::= SEQUENCE
{
    generalID               Identifier OPTIONAL, -- peer terminal ID
    algorithmOID            OBJECT IDENTIFIER OPTIONAL, -- encryption algorithm
    paramS                  Params, -- IV
    encryptedSessionKey     OCTET STRING OPTIONAL, -- encrypted session key
```

```
    encryptedSaltingKey     OCTET STRING OPTIONAL, -- encrypted media salting
                                                   -- key
    clearSaltingKey         OCTET STRING OPTIONAL, -- unencrypted media salting
                                                   -- key
    paramSsalt              Params OPTIONAL, -- IV (and clear salt) for salting
                                                   -- key encryption
    keyDerivationOID        OBJECT IDENTIFIER OPTIONAL, -- key derivation
                                                   -- method
    ...
}

END  -- End of H235-SECURITY-MESSAGES DEFINITIONS
```

Annex B

H.323 specific topics

B.1 Background

Figure B.1 gives an overview of the scope of this Recommendation within ITU-T Rec. H.323.

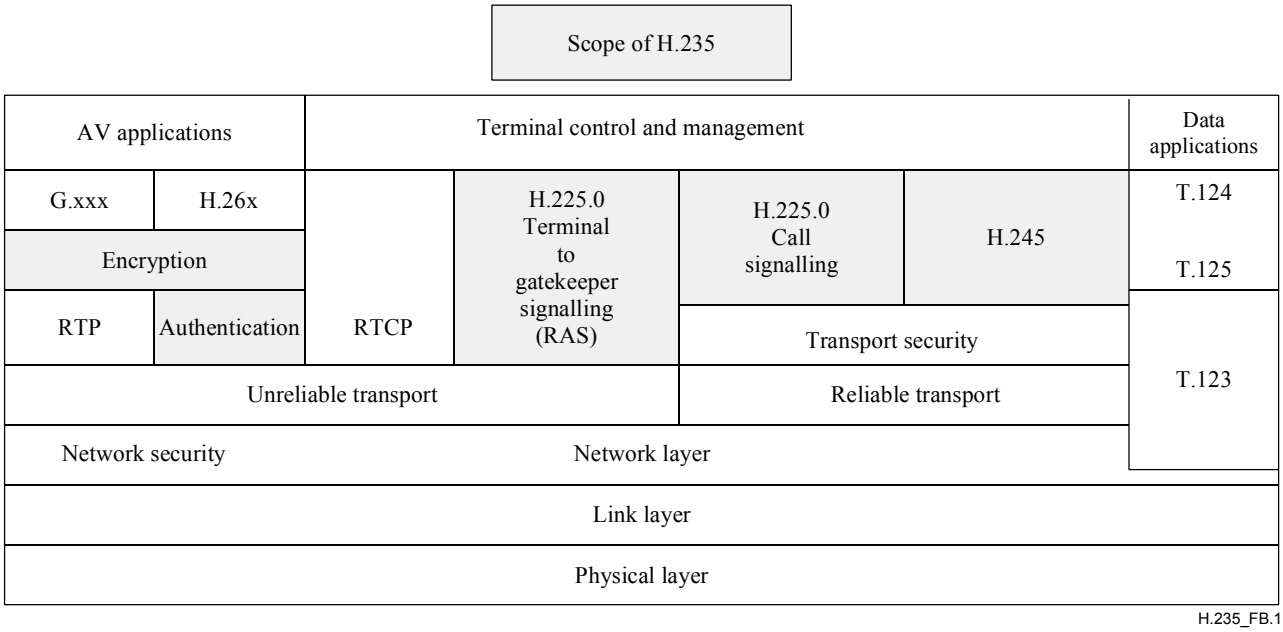


Figure B.1/H.235 – Overview

For ITU-T Rec. H.323, the signalling of usage of TLS, IPSEC or a proprietary mechanism on the H.245 control channel shall occur on the secured or unsecured H.225.0 channel during the initial Q.931 message exchange.

B.2 Signalling and procedures

The procedures outlined in clause 8/H.323 (Call signalling procedures) shall be followed. The H.323 endpoints shall have the ability to encode and recognize the presence (or absence) of security requirements (for the H.245 channel) signalled in the H.225.0 messages.

In the case where the H.225.0 channel itself is to be secured, the same procedures in clause 8/H.323, shall be followed. The difference in operation is that the communications shall only occur after

connecting to the secure TSAP identifier and using the predetermined security modes (e.g., TLS). Due to the fact that the H.225.0 messages are the first exchanged when establishing H.323 communications, there can be no security negotiations "in band" for H.225.0. In other words, both parties must know *a priori* that they are using a particular security mode. For H.323 on IP, an alternative Well Known Port (1300) is utilized for TLS secured communications.

One purpose of H.225.0 exchanges as they relate to H.323 security is to provide a mechanism to set up the secure H.245 channel. Optionally, authentication may occur during the exchange of H.225.0 messages. This authentication may be certificate- or password-based, utilizing encryption and/or hashing (i.e., signing). The specifics of these modes of operation are described in 10.2 to 10.3.4.

An H.323 endpoint that receives a SETUP message with the **h245SecurityCapability** set shall respond with the corresponding acceptable **h245SecurityMode** in the CONNECT message. In the cases in which there are no overlapping capabilities, the called terminal may refuse the connection by sending a **Release Complete** with the reason code set to ***SecurityDenied***. This error is intended to convey no information about any security mismatch and the calling terminal will have to determine the problem by some other means. In cases where the calling terminal receives a CONNECT message without sufficient, or an acceptable, security mode, it may terminate the call with a **Release Complete** with ***SecurityDenied***. In cases where the calling terminal receives a CONNECT message without any security capabilities, it may terminate the call with a **Release Complete** with ***undefinedReason***.

If the calling terminal receives an acceptable **h245Security** mode, it shall open and operate the H.245 channel in the indicated secure mode. Failure to set up the H.245 channel in the secure mode determined here should be considered a protocol error and the connection terminated.

B.2.1 Revision 1 compatibility

A security capable endpoint shall not return any security-related fields, indications or status to the non-security capable endpoint. If a caller receives a SETUP message that does not contain the **H245Security** capabilities and/or authentication token, it may return a **Release Complete** to refuse the connection; but it shall use the reason code of ***undefinedReason*** in this case. In a corresponding manner, if a caller receives a CONNECT message without an **h245SecurityMode** and/or authentication token having sent a SETUP message with **h245Security** and/or authentication token, it may also terminate the connection by issuing a **Release Complete** with a reason code of ***UndefinedReason***.

B.2.2 Error signalling

A security capable gatekeeper or other security enhanced H.225.0 entity shall provide error indications. The security error indicates that the entity was not able to correctly process the received message. Whenever possible, a detailed error code shall be provided.

- **securityWrongSyncTime** shall indicate that the sender found a security problem with inappropriate timestamps. This could be caused due to a problem with the time server, lost synchronization or due to excessive network delay.
- **securityReplay** shall indicate that a replay attack has been encountered. This is the case when the same sequence number occurs more than once for a given timestamp.
- **securityWrongGeneralID** shall indicate a mismatch of the general ID in the message. This could be caused due to wrong addressing.
- **securityWrongSendersID** shall indicate a mismatch of the sender's ID in the message. This could be caused due to user's erroneous entry.
- **securityIntegrityFailed** shall indicate that the integrity/signature check failed. For Annex D, this could be caused due to a wrong or mistyped password during the initial request or due to an encountered active attack. For Annexes E/F, this shall indicate that the

digital signature check upon the message failed. This could be caused due to a wrong private/public key applied or due to an encountered active attack.

- **securityWrongOID** shall indicate any mismatch in token OIDs (clear or crypto token) or crypto algorithm OIDs. This indicates different security algorithms/profiles implemented.
- **securityDHmismatch** shall indicate any mismatch in the Diffie-Hellman parameters exchanged. This might indicate different DH-parameter sets or even different voice encryption algorithms implemented.
- **securityCertificateExpired** shall indicate that a certificate has expired.
- **securityCertificateDateInvalid** shall indicate that a certificate is not yet valid.
- **securityCertificateRevoked** shall indicate that a certificate was found revoked.
- **securityCertificateNotReadable** shall indicate that a certificate could not be correctly ASN.1 decoded or is in other bad shape.
- **securityCertificateSignatureInvalid** shall indicate that the signature in the certificate is not correct.
- **securityCertificateMissing** shall indicate that a certificate was expected but found missing or that the certificate could not be located otherwise.
- **securityCertificateIncomplete** shall indicate that some expected certificate extensions were not present.
- **securityUnsupportedCertificateAlgOID** shall indicate that certain crypto algorithms such as hash or digital signatures used within the certificate are not understood or are not supported. As part of the returned response, the sender may provide a list of acceptable certificates in separate tokens in order to facilitate selection of an appropriate one by the recipient.
- **securityUnknownCA** shall indicate that the CA/root certificate could not be found or that the certificate could not be matched with a trusted CA.

In any other case where the H.235 security operation has failed, **securityDenial** for H.225.0 RAS (**securityDenied** for H.225.0 call signalling resp.) shall be returned.

NOTE 1 – securityWrongSyncTime, securityReplay, securityWrongGeneralID, securityWrongSendersID, SecurityIntegrityFailed, securityDHmismatch, and securityWrongOID may occur in Annex D, Annex E or in Annex F security profiles.

NOTE 2 – securityCertificateExpired, securityCertificateDateInvalid, securityCertificateRevoked, securityCertificateNotReadable, securityCertificateSignatureInvalid, securityCertificateMissing, securityCertificateIncomplete, securityUnsupportedCertificateAlgOID and securityUnknownCA may occur in Annex E or in Annex F security profiles.

B.2.3 Version 3 feature indication

H.235 version 3 and higher endpoints provide improved security procedures on the media path that H.235 version 1 and H.235 version 2 do not support. These improved security procedures are:

- the improved key transport (**V3KeySyncMaterial**, see B.2.4.1);
- the improved key update, see B.2.6.2.

Since endpoints usually do not know about their mutual support of H.235 version 3, an explicit version indication is added during call setup.

H.235 version 3 and higher endpoints should always use the procedure described in this clause for determining version 3 capability (improved key transport, improved encryption sync). Depending on the outcome of the logical signalling procedure, the endpoints may use the procedures (see B.2.4) for backward compatibility with H.235 version 1 or with version 2 endpoints.

In order to indicate whether to use the improved H.235 version 3 procedures, the calling and the called endpoint shall include an additional **ClearToken** indicating version 3 capability during the call signalling (SETUP, CONNECT, etc). Absence of such a **ClearToken** would indicate support of only H.235 version 1 or version 2. In this case, the endpoint shall use the procedure in B.2.4. Otherwise, the endpoint may use the improved procedures as described in B.2.4.1, or use the H.235 version 1 or version 2 procedure in B.2.4.

That **ClearToken** shall use **tokenOID** set to "V3" and is assigned the following value.

"V3"	{itu-t (0) recommendation (0) h (8) 235 version (0) 3 24}	Version 3 capability indicator in ClearToken during call signalling.

Any other fields in that **ClearToken** shall remain unused, unless being used to convey DH parameters.

B.2.4 Key transport

The master shall generate session key material and distribute it to the peer(s). Two procedures are offered for key transport:

- a procedure primarily for H.235 version 1 or version 2 endpoints; described in this clause.
- an improved procedure for H.235 version 3 and higher endpoints, described in B.2.4.1.

H.235 version 1 or version 2 endpoints apply the following procedure for session key transport:

KeySyncMaterial holds the endpoint identifier of the master within **generalID** and carries the session key material within **keyMaterial**. The **generalID** value should be included to provide a minimal level of authentication of the source of the session key (see also D.7.2). The recipient should verify correctness of the received **generalID**.

NOTE – This Recommendation assumes that each endpoint has registered with a gatekeeper and has obtained an endpoint identifier that can be conveyed within **generalID**. This Recommendation does not support scenarios without gatekeepers; this is left as for further study.

KeySyncMaterial shall be encrypted using the negotiated master key. The **KeySyncMaterial** shall always be padded to a multiple of blocks before encryption where the last octet shall be set to the number of padding octets (including the last). The value of the pad should be determined by the normal convention of the cipher algorithm. The encryption result shall be stored in **sharedSecret** of **H235Key**.

B.2.4.1 Improved key transport in H.235 version 3

It has been observed that the ASN.1 syntax definition of **KeySyncMaterial** and the way that the ENCRYPTED{} operation is applied to the data in H.235 versions 1 and 2, reveals plenty of known plaintext: first of all, the **generalID** of the master, but also some known coding bits for the structure. The **generalID**, even while being encrypted, is known from other non-encrypted parts of the signalling message (e.g., **senderID**). It is believed that the presence of such known plaintext significantly weakens the security scheme in such a way that an attacker could more easily crack the session key by "brute force", especially for a block cipher that has a shorter block size, such as DES-56 or RC2-compatible.

Further, version 3 of H.235 shall be capable of transporting additional key material:

- Secure transport of a salting key to the peer(s). Such a salting key is being introduced for the enhanced OFB mode; see B.2.5.

H.235 version 3 extends **H235Key** with **secureSharedSecret** containing **V3KeySyncMaterial** that holds the following parameters:

generalID holds the endpoint identifier of the originating sender if available, otherwise this field remains unused.

algorithmOID indicates the applied encryption algorithm and the operation mode.

paramS holds the initialization value, that is applied for encryption of the conveyed key(s).

NOTE 1 – The IV within **paramS** should not be confused with the per RTP packet IV that is not being signalled. **ClearSalt** optionally holds an unencrypted salting key for session key encryption (e.g., for EOFB).

encryptedSessionKey holds the ciphertext of the encrypted raw session key.

encryptedSaltingKey holds the ciphertext of the encrypted raw media salting key, if any. The salting key is necessary for the enhanced OFB mode.

clearSaltingKey may hold the unencrypted raw media salting key. Implementations shall ascertain that **encryptedSaltingKey** and **clearSaltingKey** shall not be used simultaneously.

paramSsalt holds the initial value for encrypting the salting key. **ClearSalt** optionally holds an unencrypted salting key for salting key encryption (e.g., for EOFB).

NOTE 2 – **generalID**, **algorithmOID** and **paramS** are always transmitted in plaintext, whereas **encryptedSessionKey**, **encryptedSaltingKey** hold the ciphertext of the encrypted key material.

The master generates the key(s) according to the negotiated terminal capabilities and sends the key(s) using **V3KeySyncMaterial** to the peer endpoint(s). Thus, **V3KeySyncMaterial** shall be forwarded unchanged by intermediate gatekeepers when present.

H.235 version 3 or higher endpoints should always use **secureSharedSecret** within **H235Key** but, depending on the outcome of the logical signalling procedure in B.2.3, using the indicating version 3 **ClearToken**, may use **sharedSecret** for backwards compatibility with H.235 version 1 or with version 2 endpoints.

B.2.5 Enhanced OFB mode

OFB mode (ISO/IEC 10116) defines an operation mode that deploys a stream cipher using block encryption algorithms. The OFB mode provides:

- improved performance through reduced encryption processing delay;
- easier and less complex handling of incomplete blocks;
- good error resiliency against bit errors.

Enhanced OFB mode is a slightly modified OFB mode called herein "Enhanced Output Feedback Mode" (EOFB) that deploys the same features as OFB but in addition to that:

1) uses a salting key KS in addition to the encryption key KE; and
2) introduces an implicit packet index.

Usage of an additional secret salting key KS that is being XORed to the feedback yields additional security against known-plaintext analysis. This is a major security benefit that other standard operation (such as CBC, OFB etc.) modes do not provide. Usage of the EOFB mode would thus yield increased security strength against high-redundancy plaintexts and also against known-plaintext analysis.

EOFB is defined as $C_i = P_i \oplus S_i$ with $S_i = E_{KE}(KS \oplus S_{i-1})$ for $i = 1 \ldots n$ and $S_0 = IV$ where C_i is the ith ciphertext block, P_i the ith plaintext block, S_i the ith key stream block, KE the encryption key and $\oplus$ bitwise XOR. EOFB is illustrated in Figure I.4.1.

EOFB may also run in standard OFB mode, making EOFB backwards compatible with OFB. In those cases where backwards compatibility with standard OFB mode is desired, the salting key KS shall be either set to all zeroes or equally, leaving **encryptedSaltingKey** within **V3KeySyncMaterial** empty. However, usage of an actual salting key is highly recommended for

those cases when encrypting RTP payloads with a block cipher that has a shorter block size such as DES-56 or RC2-compatible.

After at most 2^{48} packets have been processed, a new session encryption key KE and a new salting key KS shall be used, otherwise key stream reuse would occur, thereby compromising the security.

Annex D defines object identifiers for DES-56-EOFB, RC2-compatible-EOFB, 3-DES-EOFB and AES-EOFB.

B.2.6 Key update and synchronization

Media session keys do not live forever. At some point in time, each session key expires. A new session key should be used then for protecting an ongoing security session. In conferencing environments, a new group session key should be defined and distributed when group members join or leave a secured conference, thereby preventing them from accessing past or future data.

- Payload-type-based key update and synchronization defines a new dynamic payload type for that new session key; see B.2.6.1, B.2.6.2 and B.2.6.3.

For key update, this Recommendation offers an unacknowledged handshake that is applicable also for H.235 version 1 and version 2 endpoints and also a robust, acknowledged handshake for H.235 version 3 and higher endpoints.

B.2.6.1 Unacknowledged key update

Figure B.1.1 shows the unacknowledged handshake for session key distribution/key update. If the slave desires an updated session key, the slave may request a new session key from the master by issuing an **encryptionUpdateRequest** to the master. The master shall send a new session key (with or without prior **encryptionUpdateRequest** from the slave) to the slave within an **EncryptionUpdate** message.

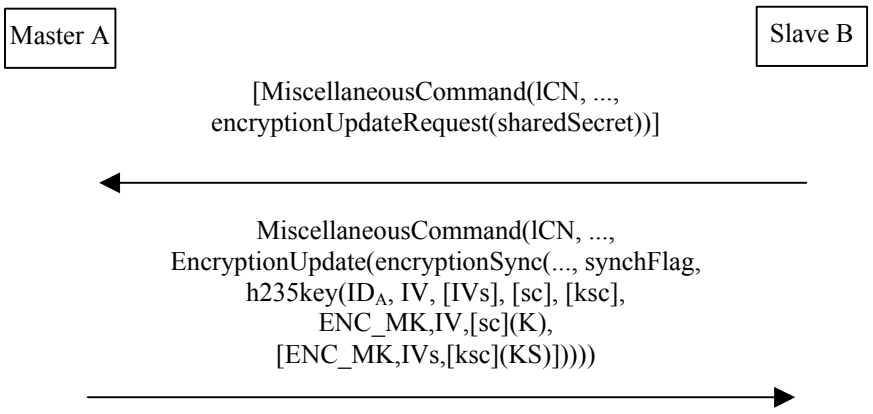


Figure B.1.1/H.235 – Unacknowledged session key distribution/key update from the master to the slave(s)

where:

lCN is the logical channel number;

synchFlag is the new dynamic RTP payload number;

ID_A is the **generalID** of the source;

IV is the initial value/vector for encryption of the sessionkey;

IVs is the initial value/vector for encryption of the salting key;

ENC_MK,IV,sc(K) means encryption of plaintext *K* using key *M*, initial vector *IV* [and a salting key *sc*, only for EOFB];

KS is the salting key for the media (for EOFB mode only);

K is the plaintext session key;

sc is the unencrypted salting key when EOFB mode is being used for encrypting the session key;

ksc is the unencrypted salting key when EOFB mode is being used for encrypting the salting key;

s2M/m2S is the **direction** flag (H.235 v3 only) (s2m = slave-to-master, m2s = master-to-slave);

[] represents an optional part.

The key update methods as described in the following clauses may deploy EOFB encryption mode for protecting the transmitted key material. In order to deploy EOFB mode for protection of the key material in the same manner as for protection of the media payload, an additional salting key (sc or ksc) is to be used.

B.2.6.2 Improved key update

H.235 version 3 and higher endpoints shall perform an explicit/implicit acknowledged key update procedure. This is to provide reliable key update methods that are based upon the unacknowledged key update method as provided by pre-H.235v3-based versions. The capability for such procedure shall be negotiated using the version 3 feature indication according to B.2.3.

Figure B.1.2 shows the key update procedures for a logical channel owned by the slave. In case the slave initiates the key update and requests a new session key from the master, the slave shall send a **MiscellaneousCommand** to the master where **logicalChannelNumber** shall hold the logical channel number (as defined by the slave), **sharedSecret** shall be set to true, the **direction** flag shall be set to **slaveToMaster** and the new dynamic payload number shall be requested in **synchFlag** within **EncryptionUpdateRequest**. If, otherwise, the master initiates the key update, this **EncryptionUpdateRequest** message shall not be sent.

The master, either responding to the slave's request or on its own behalf, shall issue an **EncryptionUpdateCommand** where the **logicalChannelNumber** shall hold the logical channel number, **direction** shall be set to **slaveToMaster** within **MiscellaneousCommand** and **synchFlag** within **encryptionSync** reflects the new dynamic payload number. **h235key** shall carry the new session key. **h235key** shall hold the identity of the master in **generalID** and the applied initial vector *IV* in **paramS**. The encrypted media session key shall be conveyed within **encryptedSessionKey**, where the encryption function shall apply the master session key and the initial value in **paramS** to the session key *K*. For EOFB, an unencrypted salting key is conveyed in **ClearSalt** within **paramS** (*sc*). **encryptedSaltingKey** shall convey the encrypted media salting key, where the encryption function shall apply the master session key and the initial value **paramSsaltIV** to the media salting key *KS*. For EOFB, an unencrypted salting key (*ksc*) is conveyed in **ClearSalt** within **paramSsalt**. **clearSaltingKey** may hold an unencrypted media salting key in which case, **encryptedSaltingKey** shall remain empty and vice versa. The transmission of an unencrypted salting key shall only be achieved if the security does not suffer, in any other case, it is recommended that the media salting key be encrypted.

The master shall be prepared to receive encrypted media under the new session key upon submitting the **EncryptionUpdateCommand** but should continue using the old session key until reception of the **EncryptionUpdateAck**. The master may apply the new session beginning with reception of the **encryptionUpdateAck**, while the slave may apply the new session key beginning with reception of the **EncryptionUpdateCommand**.

NOTE 1 – The master may choose any dynamic payload type value for the slave since the payload type is just tied to the port of the media channel.

NOTE 2 – There is no need for the slave to explicitly acknowledge reception of the new key. The master is able to deduce the reception of the issued key by the slave, when receiving media encrypted under the new payload type.

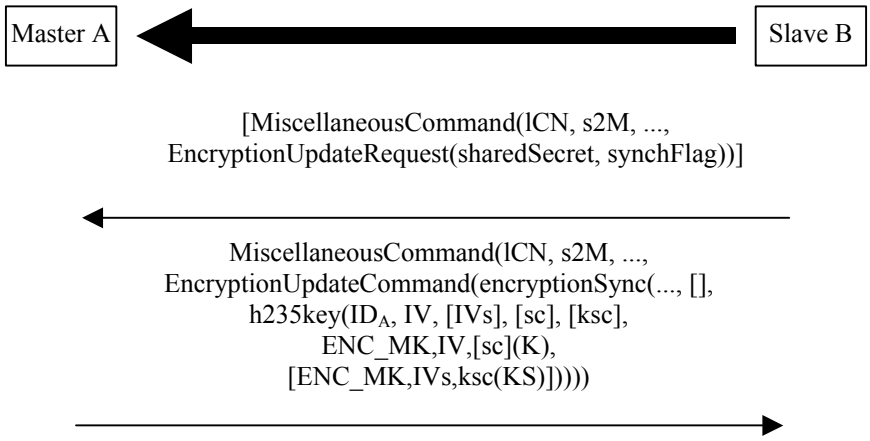


Figure B.1.2/H.235 – Session key update on slave's logical channel

Figure B.1.3 shows the key update procedures for a logical channel owned by the master. In case the slave initiates the key update and requests a new session key from the master, the slave shall send a **MiscellaneousCommand** to the master where **logicalChannelNumber** shall hold the logical channel number (as defined by the master), **sharedSecret** shall be set to true, the **direction** flag shall be set to **masterToSlave**. If, otherwise, the master initiates the key update, this **EncryptionUpdateRequest** message shall not be sent.

The master, either responding to the slave's request or on its own behalf, shall issue an **EncryptionUpdateCommand** where the **logicalChannelNumber** shall hold the logical channel number, **direction** shall be set to **masterToSlave, encryptionSync** shall provide the **synchFlag** with the new dynamic payload number. **h235key** shall carry the new session key. **h235key** shall hold the identity of the master in **generalID** and the applied initial vector *IV* in **paramS**. The encrypted media session key shall be conveyed within **encryptedSessionKey**, where the encryption function shall apply the master key and the initial value in **paramS** to the session key *K*. For EOFB, an unencrypted salting key is conveyed in **ClearSalt** within **paramS** (*sc*). For EOFB, **encryptedSaltingKey** shall convey the encrypted media salting key, where the encryption function shall apply the master session key and the initial value **paramSsaltIV** to the salting key *KS*. For EOFB, an unencrypted salting key (*ksc*) is conveyed in **ClearSalt** within **paramSsalt**. **clearSaltingKey** may hold an unencrypted media salting key in which case **encryptedSaltingKey** shall remain empty and vice versa. The transmission of an unencrypted salting key shall only be achieved if the security does not suffer, in any other case, it is recommended that the media salting key be encrypted.

The slave shall acknowledge reception of the new session key by responding with **MiscellaneousCommand** where the **logicalChannelNumber** shall hold the logical channel number, and **encryptionUpdateAck** shall reflect the new dynamic payload number in **synchFlag**.

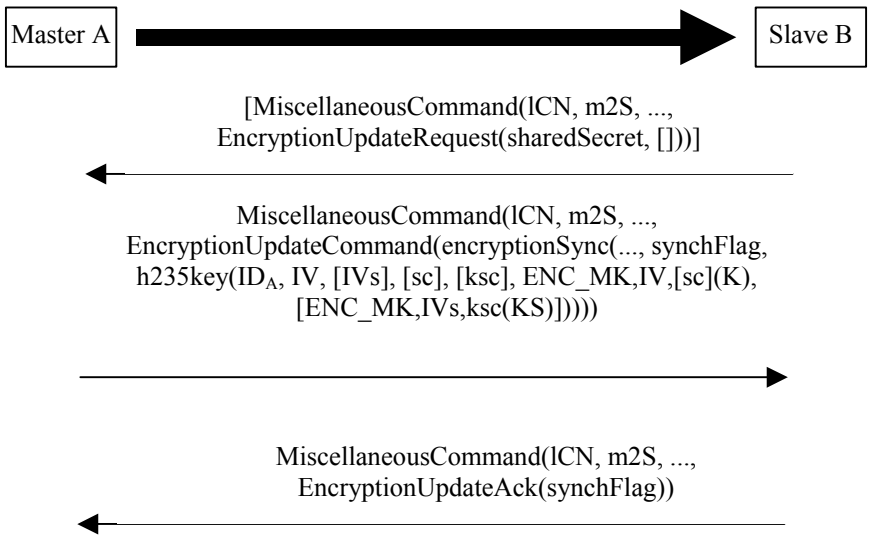


Figure B.1.3/H.235 – Session key update on master's logical channel

B.2.6.3 Payload-type-based key update and synchronization

Initial encryption key is presented by the master in conjunction with the dynamic payload number in **synchFlag** (via **EncryptionSync** in ITU-T Rec. H.245). The receiver(s) of the media stream shall start initial use of the key upon receipt of this payload number in the RTP header.

If the negotiated logical channel carries only a single payload type, the value of the **synchFlag** may replace the negotiated payload type in the RTP header. If, on the other hand, the negotiated logical channel may carry more than one payload type (even if only in separate RTP packets), then the RTP packets shall be formatted as described in RFC 2198, with the **synchFlag** value acting as the encapsulating payload type, and the actual payload type(s) residing in the additional header block(s) as specified by RFC 2198.

New key(s) may be distributed at any time by the master endpoint. The synchronization of the newer key with the media stream shall be indicated by the changing of the payload type to a new dynamic value.

NOTE – The specific values do not matter, as long as they change for every new key that is distributed.

B.3 RTP/RTCP issues

The use of encryption on the RTP stream will follow the general methodology recommended in the document referenced in [RTP]. The encryption of the media shall occur in an independent, packet-by-packet basis[1]. The RTP header shall not be encrypted. For audio/video codecs, the entire audio/video codec payload including any audio/video payload header(s) shall be encrypted. Synchronization of new keys and encrypted text is based upon dynamic payload type (see B.2.6.3).

It is assumed that encryption is applied just to the payload in each RTP packet, the RTP headers remaining in the clear. It is assumed that all RTP packets must be a multiple of whole octets. How the RTP packets are encapsulated at the transport or network layer is not relevant to this Recommendation. All modes must allow for lost (or out-of-sequence) packets, in addition to padding packets to an appropriate multiple of octets.

---

[1] It should be noted that if RTP packet size is larger than MTU size, partial loss (of fragment) will cause the whole RTP packet to be indecipherable.

Deciphering the stream must be stateless due to the fact that packets may be lost; each packet should be deciphered independently. Two requirements of block algorithm mode shall operate as follows:

B.3.1 Initialization vectors

Most block modes involve some "chaining"; each encryption cycle depends in some way on the output of the previous cycle. Therefore, at the beginning of a packet, some initial block value (usually called an Initialization Vector (IV)) must be provided in order to start the encryption process. Independent of how many stream octets are processed on each encryption cycle, the length of the IV is always equal to the length of a block. All modes except Electronic Code Book (ECB) mode require an IV.

B.3.1.1 CBC initialization vector

An Initialization Vector (IV) is required when using a block cipher in CBC mode to encrypt RTP packet payloads. The size of an IV is the same as the block size for the particular block cipher. For example, the IV size for DES and 3-DES is 64 bits, while for AES it is 128 bits.

For the CBC case, an IV shall be constructed from the first *B* (where *B* is the block size) octets of: Seq# concatenated with Timestamp. This forms the pattern, *SSTTTT*, where *SS* is the 2-octet RTP Seq# and *TTTT* is the 4-octet RTP timestamp. This pattern shall be repeated until *B* octets have been generated, truncating as necessary. For example, 64- and 128-bit IVs would contain *SSTTTTSS* and *SSTTTTSSTTTTSSTT*, respectively. It should be noted that the IV generated in this manner may produce a key pattern that is considered "weak" for a particular algorithm.

B.3.1.2 EOFB initialization vector

The unique initial vector IV for each RTP packet in EOFB mode shall be computed as follows:

Each RTP packet is associated with an implicit 48-bit packet index *i* as defined in [SRTP] where $i = 2^{16} \times \text{ROC} + \text{SEQ}$ with SEQ the sequence number taken from the RTP header and ROC is the 32-bit rollover counter counting how often the sequence number SEQ has been wrapped around through 65535.

Initially, the rollover counter ROC shall be set to zero. Each time the SEQ wraps modulo 2^{16}, the sender shall increment ROC by one modulo 2^{32}.

The initial vector IV is computed as ($i \parallel T$ [$\parallel i \parallel T \parallel \ldots$]) with the 48-bit index *i* and 32-bit timestamp *T* taken from the RTP header concatenated several times until the block size is filled-up. The $\parallel$ symbol represents concatenation.

NOTE – The rollover counter and IV are maintained and computed locally at each peer side and do not get transmitted.

The receiver, when facing lost or reordered packets, should compute an estimated index *i* as:

$i = 2^{16} \times v + \text{SEQ}$ where *v* is chosen from the set {ROC-1, ROC, ROC+1} modulo 2^{32} such that *v* is closest (in 2^{48} sense) to the value $2^{16} \times \text{ROC} + s_l$ where s_l is the maintained sequence number at the receiver. After the packet has been processed using the estimated index, the receiver shall decide if s_l and ROC should be updated. For instance, a simple (but not error robust) method is to simply set s_l to SEQ (if SEQ > s_l) and, if the value $v = \text{ROC} + 1$ was used, to update ROC to *v*; see also [SRTP, section 3.2.1] for more information.

B.3.2 Padding

ECB and CBC modes always process the input stream a block at a time and, while CFB and OFB can process the input in any number of octets, N ($\leq B$), it is recommended that $N = B$.

Two methods are available to handle packets whose payload is not a multiple of blocks:

1) Ciphertext Stealing for incomplete blocks for ECB and CBC; no padding for CFB and EOFB.

2) Padding in the manner prescribed by [RTP, section 5.1].

[RTP, section 5.1] describes a method of padding in which the payload shall be padded to a multiple of blocks. The last octet shall be set to the number of padding octets (including the last), and the *P* bit set in the RTP header. The value of the pad should be determined by the normal convention of the cipher algorithm.

All H.235 implementations shall support both schemes. The scheme in use can be deduced as follows: if the *P* bit is set in the RTP header, then the packet is padded; if the packet is not a multiple of *B* and the *P* bit is not set, then Ciphertext Stealing applies, else the packet is a multiple of *B*, and padding does not apply.

B.3.3 RTCP protection

Application of cryptographic techniques to RTCP elements is for further study.

B.3.4 Secured payload stream

H.323-based networks, when being used, for example, for Modem-over-IP transmission, deploy H.245 signalling to establish and negotiate a voiceband data channel and RTP for packetization of a Multiple Payload Stream (MPS).

For a single media stream with a single payload type or FEC for another channel, the dynamic payload type in **encryptionSync** shall replace the default payload type.

For encapsulating streams, (i.e., redundancy encoding or RFC 2198 encoded FEC) the dynamic payload type within **encryptionSync** shall replace the encapsulating payload type.

For multiple payload streams, the dynamic payload type in **syncFlag** of **encryptionSync** shall be ignored and the (optional) payload types within the **multiplePayloadStreamElement**(s) shall be used instead.

The **EncryptionUpdateCommand** shall be used for the improved key update procedure to distribute new session key material (see B.2.6.2). **multiplePayloadStream** is only used when a multiple payload stream is to be re-keyed, in which case the dynamic payload type within **EncryptionSync** shall be ignored.

B.3.5 Interworking with J.170

For further study.

B.4 RAS signalling/procedures for authentication

B.4.1 Introduction

This annex will not explicitly provide any form of message privacy between gatekeepers and endpoints. There are two types of authentication that may be utilized. The first type is symmetric encryption-based that requires no prior contact between the endpoint and gatekeeper. The second type is subscription-based and will have two forms: password or certificate. All of these forms are derived from the procedures shown in 10.1, 10.3.2, 10.3.3 and 10.3.4. In this annex, the generic labels (EPA and EPB) shown in the aforementioned clauses will represent the endpoint and gatekeeper respectively.

B.4.2 Endpoint-gatekeeper authentication (non-subscription-based)

This mechanism may provide the gatekeeper with a cryptographic link that a particular endpoint which previously registered, is the same one that issues subsequent RAS messages. It should be noted that this may not provide any authentication of the gatekeeper to the endpoint, unless the

optional signature element is included. The establishment of the identity relationship occurs when the terminal issues the **GRQ**, as outlined in 7.2.1/H.323. The Diffie-Hellman exchange shall occur in conjunction with the **GRQ** and **GCF** messages as shown in the first phase of 10.1. This shared secret key shall now be used on any subsequent **RRQ**/**URQ** from the terminal to the gatekeeper. If a gatekeeper operates in this mode and receives a **GRQ** without a token containing the *DHset* or an acceptable algorithm value, it shall return a **securityDenial** reason code or other appropriate security error code according to B.2.2 in the **DRJ**.

The Diffie-Hellman shared secret key as created during the **GRQ**/**GCF** exchange may be used for authentication on subsequent **xRQ** messages. The following procedures shall be used to complete this mode of authentication.

Terminal **(xRQ)**

1) The terminal shall provide all of the information in the message as described in the appropriate clauses of ITU-T Rec. H.225.0.

2) The terminal shall encrypt the **GatekeeperIdentifier** (as returned in the **GCF**) using the shared secret key that was negotiated. This shall be passed in a **clearToken** (see 10.2) as the **generalID**.

The 16 bits of the **random** and then the **requestSeqNum** shall be XOR'd with each 16 bits of the **GatekeeperIdentifier**. If the **GatekeeperIdentifier** does not end on an even 16 boundary, the last 8 bits of the **GatekeeperIdentifier** shall be XOR'd with the least significant octet of the random value and then **requestSeqNum**. The **GatekeeperIdentifier** shall be encrypted using the selected algorithm in the **GCF** (algorithmOID) and utilizing the entire shared secret.

The following example illustrates this procedure:

RND16: 16-bit value of the Random Value

SQN16: 16-bit value of requestSeqNum

BMPX: the Xth BMP character of GatekeeperIdentifier

BMP1' = (BMP1) XOR (RND16) XOR (SQN16)

BMP2' = (BMP2) XOR (RND16) XOR (SQN16)

BMP3' = (BMP3) XOR (RND16) XOR (SQN16)

BMP4' = (BMP4) XOR (RND16) XOR (SQN16)

BMP5' = (BMP5) XOR (RND16) XOR (SQN16)

:

:

BMPn' = (BMPn) XOR (RND16) XOR (SQN16)

In order to cryptographically link this and subsequent messages with the original registrant (the endpoint that issued the **RRQ**), the most recent **random** value returned shall be utilized (this value may be one newer than the value returned in the **RCF** from a later **xCF** message).

Gatekeeper **(xCF/xRJ)**

1) Gatekeeper shall encrypt its **GatekeeperIdentifier** (following the above procedure) with the shared secret key associated with the endpoint alias and compare this to the value in the **xRQ**.

2) Gatekeeper shall return **xRJ** if the two encrypted values do not match.

3) If **GatekeeperIdentifier** matches, gatekeeper shall apply any local logic and respond with **xCF** or **xRJ**.

4) If an **xCF** is sent by the gatekeeper, it should contain an assigned **EndpointIdentifier** and a new random value in the **random** field of a **clearToken**.

Refer to the second phase of Figure 1 for a graphical representation of this exchange. The gatekeeper knows which shared secret key to use to decipher the gatekeeper identifier by the alias name in the message.

B.4.3 Endpoint-gatekeeper authentication (subscription-based)

All RAS messages other than GRQ/GCF should contain the authentication tokens required by the specific mode of operation. There are three different variations that may be implemented depending on requirements and environment:

1) password-based with symmetric encryption;

2) password-based with hashing;

3) certificate-based with signatures.

In all cases, the token will contain the information as described in the following subclauses depending on the variation chosen. If a gatekeeper operates in a secure mode and receives a RAS message without an acceptable token value, it shall return a **securityDenial** reason code or other appropriate security error code according to B.2.2 in the reject message. In all cases, the return token from GK is optional; if omitted, only one-way authentication is achieved.

B.4.3.1 Password with symmetric encryption

The gatekeeper discovery phase (GRQ, GCF and GRJ) may be unsecured as shown in Figure B.2, or may be secured using the **cryptoTokens**.

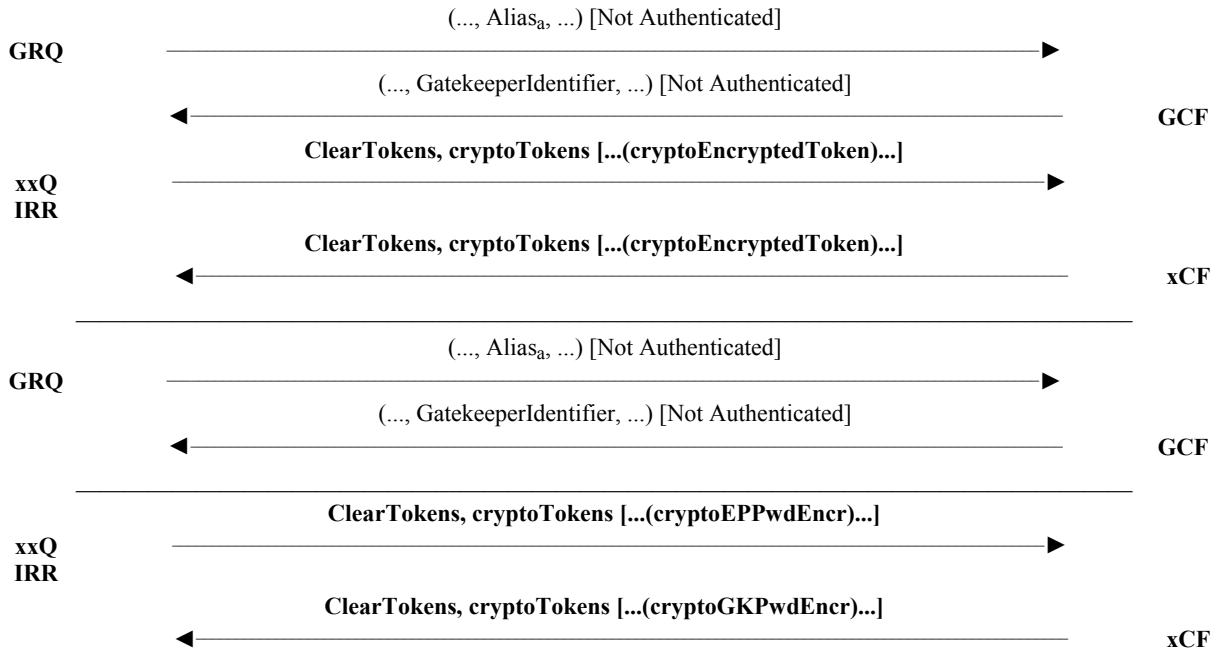


Figure B.2/H.235 – Password with symmetric encryption

B.4.3.2 Password with hashing

The gatekeeper discovery phase (GRQ, GCF and GRJ) may be unsecured as shown in Figure B.3, or may be secured according to Annex D using the **cryptoTokens**.

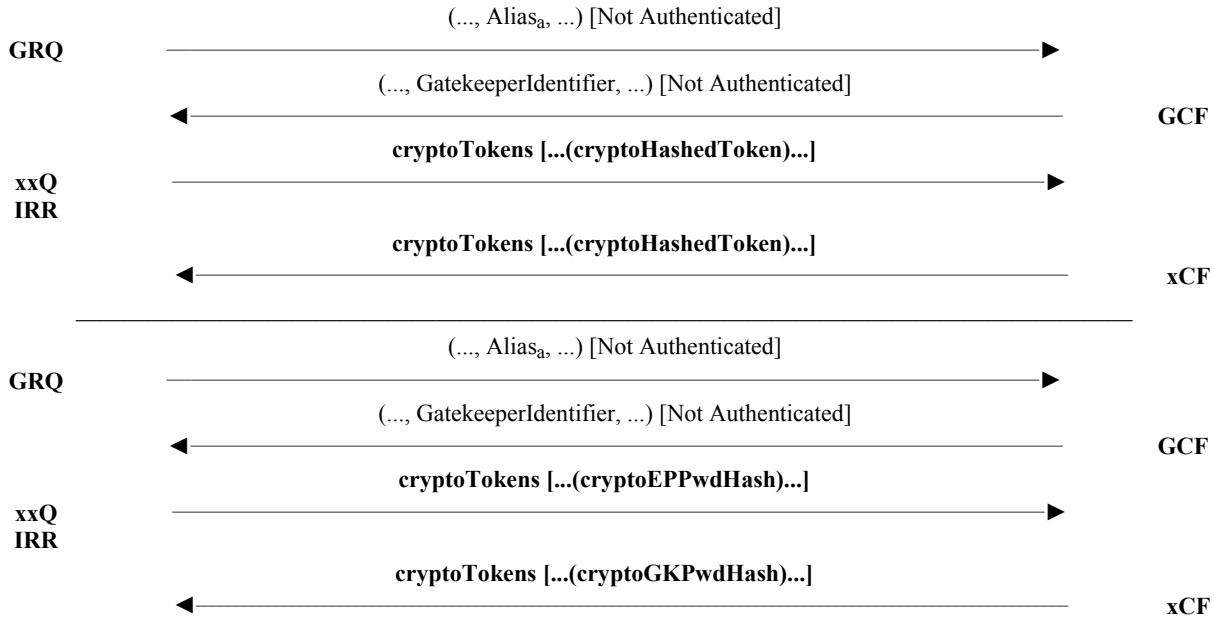


Figure B.3/H.235 – Password with hashing

B.4.3.3 Certificate-based with signatures

The gatekeeper discovery phase (GRQ, GCF and GRJ) may be unsecured as shown in Figure B.4, or may be secured according to Annex E using the **cryptoTokens**.

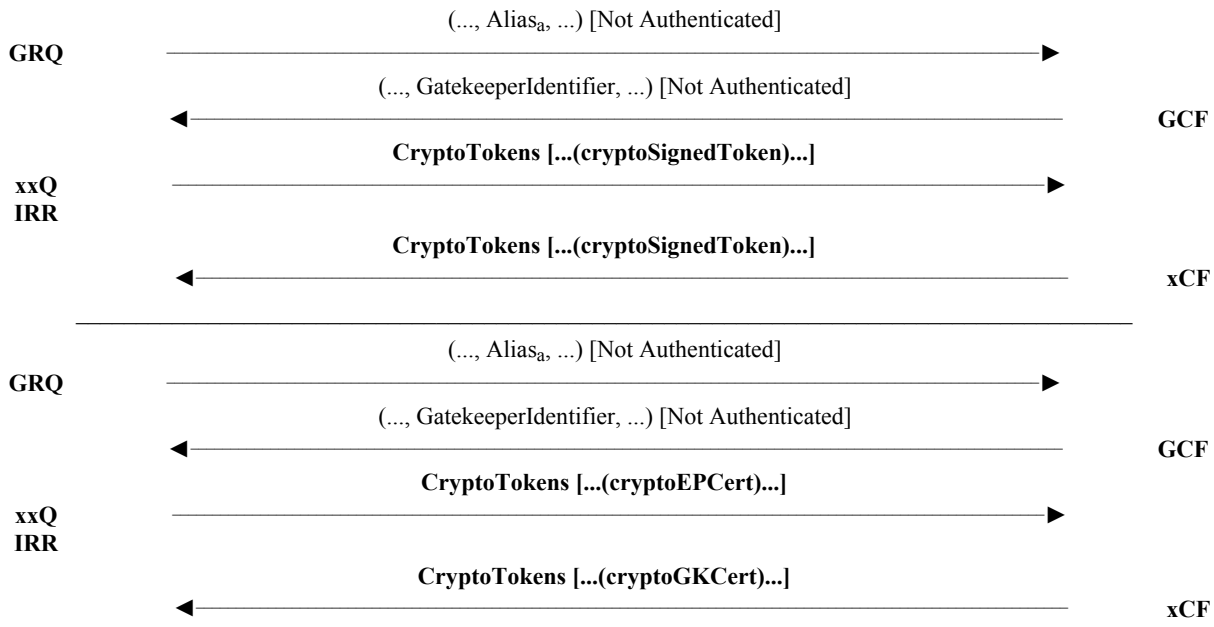


Figure B.4/H.235 – Certificate-based with signatures

B.5 Non-terminal interactions

B.5.1 Gateway

As stated in 6.6, an H.323 gateway should be considered a trusted element. This includes protocol gateways (H.323-H.320 etc., …) and security gateways (proxy/firewalls). The media privacy can be assured between the communicating endpoint and the gateway device; but what occurs on the far side of the gateway should be considered insecure by default.

B.6 Key management on the RAS channel

In some circumstances, it is desirable to distribute (RAS) session keys from a gatekeeper to one or more endpoints under its control, or from one endpoint to another. The proposed mechanism assumes that the gatekeeper and the endpoint share a strong, secret key or know each other's public key. One example of such a case would be for a routing gatekeeper to issue a session key to an endpoint in a RAS message, such as **RCF** or **ACF**, for use in encrypting a gatekeeper-routed signalling channel. Another example might be one in which the gatekeeper issues a session key for use in encrypting succeeding RAS communications (e.g., **RRQ** or **ARQ**).

This mechanism is similar to that used for distribution of media session keys. It may be used to avoid the overhead of key negotiation in certain circumstances.

For key transport, the optional **h235Key** field of the **ClearToken** should be used in H.235v3. The flexibility of the **H235Key** element will permit the transport of encryption key material using:

- a secure channel (the **secureChannel** option) assuming the RAS or call signalling channel is secured by other means (IPSEC/SSL, etc.);
- a shared encryption secret over a clear channel (the **sharedSecret** choice), or similarly but preferably the **secureSharedSecret** choice;
- a public-key encryption and certificate over a clear channel (the **certProtectedKey** option).

The usage of the exchanged RAS session key and its application to RAS, call signalling messages and/or transport channels is left as for further study.

B.7 Pseudo-Random Function (PRF)

This clause defines a pseudo-random function for the purpose of deriving dynamic keys from a static key material and a random value.

NOTE – This PRF is identical to the MIKEY PRF (see [MIKEY]/RFC xxxx).

The key derivation method has the following input parameters:

- *inkey*: the input key to the derivation function.
- *inkey_len*: the length in bits of the input key.
- *label*: a specific label, dependent on the type of the key to be derived and the random **challenge** value.
- *outkey_len*: desired length in bits of the output key.

The pseudo-random function has the following output:

- *outkey*: the output key of desired length.

Let HMAC (see RFC 2104) be the SHA1- [(see ISO/IEC 10118-3)] based message authentication function. Similar to RFC 2246, define:

$$P(s, label, m) = \text{HMAC}(s, A_1 \,||\, label) \,||\, \text{HMAC}(s, A_2 \,||\, label) \,||\, \ldots \,||\, \text{HMAC}(s, A_m \,||\, label)$$

where:

A_0 = label,

A_i = HMAC (s, A_{i-1}).

While SHA1 ISO/IEC 10118-3 is the default, HMAC using other hash functions may be used; this is left as for further study.

The following procedure describes a pseudo-random function, denoted *PRF(inkey, label)*, applied to compute the output key, outkey:

- let $_n$ = *inkey_len/512*, rounded up to the nearest integer;
- split the *inkey* into *n* blocks, $inkey = s_1 \,||\, \ldots \,||\, s_n$, where all s_i, except possibly s_n, are 512 bits each;
- let m = *outkey_len/160*, rounded up to the nearest integer.

Then, the output key, *outkey*, is obtained as the *outkey_len* most significant bits of:

$$PRF(inkey, label) = P(s_1, label, m) \text{ XOR } P(s_2, label, m) \text{ XOR } \ldots \text{ XOR } P(s_n, label, m).$$

Annex C

H.324 specific topics

For further study.

Annex D

Baseline security profile

D.1 Introduction

This annex describes simple, baseline security profiles. The specified security profiles are based upon ITU-T Rec. H.235, available ETSI and IMTC security profiles. The security profiles select appropriate security features from ITU-T Rec. H.235 with its rich set of options.

D.2 Specification conventions

Some explanation is useful for understanding the terms used in this annex:

The annex defines a **baseline security profile**. The baseline security profile provides basic security by simple means using secure password-based cryptographic techniques. The baseline security profile may use the **voice encryption security profile** for achieving voice confidentiality if

necessary. A more sophisticated security profile that applies digital signatures and overcomes the limitations of the baseline security profile can be found in Annex E.

This annex uses H.235 fields for provisioning authentication/integrity security services upon H.323 signalling messages. Different object identifiers (see D.11) determine which security service is actually selected and which protocol version of this Recommendation is being used. Procedure I specifies how to implement the security services by certain security mechanisms such as symmetric (keyed hashing) techniques. The object identifiers are referenced through a symbolic reference in the text (e.g., "A"), see also clause 5.

While the message integrity service also always provides message authentication, the reverse is not always true. In practice, combined authentication and integrity service exploit the same key material without introducing a security weakness.

Moreover, all hop-by-hop security information is put into the **CryptoHashedToken** element. This information is re-computed at every hop.

Generally, what password, session key and shared secret all have in common is that they are used in symmetric cryptography among two (or more) entities. The difference between a password and a session key/shared secret is how the keys are actually applied, e.g., passwords for authentication and authorization, session keys for encryption. The term "shared secret" is kind of neutral as it does not actually refer to any specific usage.

The **password** (could be viewed also as a shared secret) is used for the authentication/integrity for RAS and H.225.0, as this item could be entered by the user. The password usually has a longer-term lifetime; the password is known *a priori* and may be defined as part of the overall user subscription process. Some algorithm (e.g., piping the password through a hash algorithm) may transform the password for more convenient processing in the protocols in order to result in a fixed length.

The **session key** for encrypting media streams, on the other hand, is generated by the master just for a specific RTP session (on an OLC), at longest for one call. The generated session key is encrypted with a key that is derived from the agreed Diffie-Hellman **shared secret** that both endpoints have computed. In this case, the DH-shared secret acts as Master Key for protection of the session key(s).

The H.235 **ClearToken** offers a field called **random** holding a 32-bit integer. This field is used in the following sense: **random** is actually a monotonically increasing number starting at any value and increasing with every outgoing message. The **random** field is used as an additional "randomization" value for input to the keyed-hashed function in the case when several messages are issued shortly one after another, yet convey identical timestamps. This could happen when the UTC clock does not provide sufficient clock resolution. In essence, the produced hash value or integrity check value look different due to the changing **random** value. This is to counter replay attacks. For implementation simplicity, an increasing counter is preferred over a truly random sequence here. The recipient may keep received **timestamp**/**random** pairs during the period defined by a local time window[2]. Replay attacks can be identified when the same **timestamp/random** pair occurs twice.

This profile defines to "set the **generalID** in the **ClearToken** to the identifier of the recipient". This actually means that, for RAS messages destined for the gatekeeper, this is the GK identifier; for RAS messages destined for the endpoint, this is the endpoint identifier, for H.225.0 call signalling messages destined for the gatekeeper, this is the GK identifier and for H.225.0 call signalling messages destined for the endpoint, this is the called endpoint identifier, see also clause D.10.

The **sendersID** shall be set to the identification string of the sender. This actually means that for RAS messages destined for the gatekeeper, this is the endpoint identifier; for RAS messages

---

2 The time window compensates for variances of the synchronized time and for the network transit delay.

destined for the endpoint, this is the gatekeeper identifier; for H.225.0 call signalling messages destined for the gatekeeper, this is the GK identifier and for H.225.0 call signalling messages destined for the endpoint, this is the called endpoint identifier, see also clause D.10.

A **block** refers to the basic unit of packed bits that the block cipher is able to encrypt/decrypt with an elementary crypto operation; for DES and triple-DES, the block size is 64 bits, for AES the block size is 128 bits.

This annex may apply message integrity protection that spans the entire message. For H.225.0 RAS, the integrity protection covers the entire RAS message; for call signalling, this covers the entire H.225.0 call signalling message, including the Q.931 headers.

In order to avoid references to a trademark (RC2®), this annex actually references an "RC2-compatible" encryption algorithm.

This Recommendation uses well-known security terms such as key, key management and SET, which have different meanings in other contexts (e.g., touch key pad, Q.931/Q.932 feature key management, and Secure Electronic Transaction protocol).

D.3 Scope

This annex describes simple security for H.323 entities. The security profile may be applied by secured H.323 terminals including **secure simple telephone terminal** (Secure Audio Simple Endpoint Type) defined in this annex (see D.6); the security profile may be applied by other H.323 entities such as gateways, gatekeepers, MCUs.

D.4 Abbreviations

This annex uses the following abbreviations:

AES	Advanced Encryption Algorithm
BES	Back-end Server
CBC	Cipher Block Chaining
DES	Data Encryption Standard
DH	Diffie-Hellman
ECB	Electronic Code Book
EP	Endpoint
ETSI	European Telecommunications Standards Institute
GK	Gatekeeper
HMAC	Hashed Message Authentication Code
IMTC	International Multimedia Teleconferencing Consortium
IPSEC	Internet Protocol Security
ITU	International Telecommunication Union
IV	Initialization Vector
KS	Salting Key in EOFB mode
MAC	Message Authentication Code
MD5	Message Digest 5
NAT	Network Address Translation
OID	Object Identifier
PFS	Perfect Forward Secrecy
RAS	Registration, Admission and Status
RSA	Rivest, Shamir and Adleman

RTP	Real-Time Transport Protocol
SASET	Secure Audio Simple Endpoint Type
SET	Simple Endpoint Type
SHA	Secure Hash Algorithm
TCP	Transmission Control Protocol
TIPHON	Telecommunications and Internet Protocol Harmonization Over Networks
TLS	Transport Layer Security
VoIP	Voice over Internet Protocol

D.5 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; 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. The reference to a document within this Recommendation does not give it, as a stand-alone document, the status of a Recommendation.

AES [FIPS-197]	US National Institute of Standards, "*Advanced Encryption Algorithm (AES)*", Federal Information Processing Standard, (FIPS) Publication 197, November 2001, http://csrc.nist.gov/publications/fips/fips197/fips-197.pdf.
DES [FIPS-46-2]	US National Institute of Standards, "*Data Encryption Standard*", Federal Information Processing Standard, (FIPS) Publication 46-2, December 1993, http://www.itl.nist.gov/div897/pubs/fip46-2.htm.
DES [FIPS-74]	USNational Institute of Standards, "*Guidelines for Implementing and Using the Data Encryption Standard*", Federal Information Processing Standard (FIPS) Publication 74, April 1981, http://www.itl.nist.gov/div897/pubs/fip74.htm.
DES [FIPS-81]	US National Bureau of Standards, "*DES Modes of Operation*", Federal Information Processing Standard (FIPS) Publication 81, December 1980, http://www.itl.nist.gov/div897/pubs/fip81.htm.
[ISO/IEC 10118-3]	ISO/IEC 10118-3:2004, *Information Technology – Security techniques – Hash-functions – Part 3: Dedicated hash-functions*.
[H.225.0]	ITU-T Rec. H.225.0 version 5 (2003), *Call signalling protocols and media stream packetization for packet-based multimedia communication systems*.
[H.235v1]	ITU-T Rec. H.235 version 1 (1998), *Security and encryption for H-series (H.323 and other H.245-based) multimedia terminals*.
[H.235v2]	ITU-T Rec. H.235 version 2 (2000), *Security and encryption for H-series (H.323 and other H.245-based) multimedia terminals*.
[H.245]	ITU-T Rec. H.245 version 10 (2003), *Control protocol for multimedia communication*.
[H.323]	ITU-T Rec. H.323 version 5 (2003), *Packet-based multimedia communication systems*.
[H.323 Annex F]	ITU-T Rec. H.323 Annex F (1999), *Simple endpoint types*.
[RFC 2268]	RFC 2268 (1998), *A Description of the RC2® Encryption Algorithm*.

D.6 Baseline security profile

This clause describes a baseline for the simple security profile.

D.6.1 Overview

The baseline security profile mandates the GK-routed model. The baseline security is applicable in administered environments with symmetric keys/passwords assigned among the entities (terminal-gatekeeper, gatekeeper-gatekeeper, gateway-gatekeeper).

The features provided by these profiles include:

– For RAS, H.225.0 and H.245 messages:
  - User authentication to a desired entity irrespective of the number of application level hops[3] that the message traverses.
  - Integrity of the signalling message itself, including the critical portions (fields) of messages arriving at an entity irrespective of the number of application level hops that the message traverses.
  - Application level hop-by-hop signalling message authentication and integrity provides these security services for the entire message.

– For the media stream:
  - Confidentiality of the media stream is provided by symmetric encryption.

Several attacks are thwarted by providing the above security services in a suitable fashion. These include:

- Denial-of-service attacks: Rapid checking of cryptographic hash values can prevent such attacks.
- Man-in-the-middle attacks: Application level hop-by-hop message authentication and integrity prevents against such attacks when the man in the middle is between an application level hop, say, a hostile router.
- Replay attacks: Use of timestamps and sequence numbers prevent such attacks.
- Spoofing: User authentication prevents such attacks.
- Connection hijacking: Use of authentication/integrity for each signalling message prevents such attacks.
- Eavesdropping of media stream is countered by encryption and use of secret keys.

Other highlights of the simple security profile include:

- Use of robust, well-known and widely deployed algorithms based on IMTC/ETSI/IETF material.
- Capability of deployment in stages based on the security requirement of the business model.
- Applicablility to various deployment scenarios such as in closed groups and for scaleable environments and in multipoint conferences.
- The authentication-only security profile is applicable when providing some security for NAT/firewall traversal.

Table D.1 summarizes all the procedures defined in this annex by the security profiles dealing with different security requirements. The table includes the baseline security profile (vertical shading –

---

[3] Hop is understood here in the sense of a trusted H.235 network element (e.g., GK, GW, MCU, proxy, firewall). Thus, application level hop-by-hop security, when used with symmetric techniques, does not provide true end-to-end security between terminals.

blue in the electronic copy) and the voice encryption security profile (horizontal shading – green in the electronic copy). The optional authentication-only security profile is shown as diagonal shading – blue in the electronic copy.

Table D.1/H.235 – Summary of Annex D security profiles

<table>
<tr><th rowspan="2">Security services</th><th colspan="5">Call functions</th></tr>
<tr><th>RAS</th><th colspan="2">H.225.0</th><th>H.245 (Note)</th><th>RTP</th></tr>
<tr><td>Authentication</td><td>Password HMAC-SHA1-96</td><td colspan="2">Password HMAC-SHA1-96</td><td>Password HMAC-SHA1-96</td><td></td></tr>
<tr><td>Authentication-only</td><td>Password HMAC-SHA1-96</td><td colspan="2">Password HMAC-SHA1-96</td><td>Password HMAC-SHA1-96</td><td></td></tr>
<tr><td>Non-repudiation</td><td></td><td colspan="2"></td><td></td><td></td></tr>
<tr><td>Integrity</td><td>Password HMAC-SHA1-96</td><td colspan="2">Password HMAC-SHA1-96</td><td>Password HMAC-SHA1-96</td><td></td></tr>
<tr><td></td><td></td><td colspan="2"></td><td></td><td>56-bit DES | 56-bit RC2-compatible | 168-bit triple-DES | 128-bit AES</td></tr>
<tr><td>Confidentiality</td><td></td><td colspan="2"></td><td></td><td>CBC-mode or EOFB-mode</td></tr>
<tr><td>Access control</td><td></td><td colspan="2"></td><td></td><td></td></tr>
<tr><td>Key management</td><td>Subscription-based password assignment</td><td>Subscription-based password assignment</td><td>Authenticated Diffie-Hellman key-exchange</td><td>Integrated H.235 session key management (key distribution, key update using 56-bit DES/56-bit RC2-compatible/ 168-bit triple-DES, 128-bit AES)</td><td></td></tr>
<tr><td colspan="6">NOTE – Tunnelled H.245 or embedded H.245 inside H.225.0 fast connect.</td></tr>
</table>

For authentication, the user shall use a password-based scheme. The password-based scheme is highly recommended for authentication due to its simplicity and ease of implementation. Hashing all the fields in the H.225.0 RAS and call signalling messages is the recommended approach for integrity of the messages (also using the password scheme).

Secure H.323 entities with this security profile realize authentication in conjunction with integrity using the same common security mechanism.

For optional voice confidentiality, the suggested scheme is encryption using AES-128, RC2-compatible, DES or triple-DES based on the business model and exportability requirement. Some environments that are already offering a certain degree of confidentiality may not require voice encryption. In this case, Diffie-Hellman key agreement and other key management procedures are not necessary as well.

H.323 entities, when deploying the voice encryption security profile, shall implement 56-bit DES as the default encryption algorithm; they may implement 128-bit AES or 168-bit triple-DES or they may implement exportable encryption using 56-bit RC2-compatible.

Access control means are not explicitly described; they can be implemented locally upon the received information conveyed within H.235 signalling fields (ClearToken, CryptoToken).

This Recommendation does not describe procedures for subscription-based password/secret key assignment with management and administration. Such procedures may take place by means that are beyond the scope of this annex.

The communication entities involved are able to implicitly determine usage of either the baseline security or the signature security profile by evaluating the signalled security object identifiers in the messages (**tokenOID**, and **algorithmOID**; see also D.11).

D.6.1.1 Baseline security profile

The baseline security profile is applicable in an environment where subscribed passwords/symmetric keys can be assigned to the secured H.323 entities (terminals) and network elements (GKs, proxies). It provides authentication and integrity, or authentication-only for H.225.0 RAS and call signalling, H.225.0 and tunnelled H.245 using password-based HMAC-SHA1-96 hash as specified by Procedure I. H.225.0 call establishment using FastStart (GK-to-GK or terminal-to-terminal) includes integrated key management with Diffie-Hellman.

The vertically shaded area (blue in electronic copy) in Table D.2 represents the baseline security profile.

Table D.2/H.235 – Baseline security profile

Security services	Call functions			
	RAS	H.225.0	H.245	RTP
Authentication and integrity[4]	Password HMAC-SHA1-96	Password HMAC-SHA1-96	Password HMAC-SHA1-96	
Non-repudiation				
Confidentiality				
Access control				
Key management	Subscription-based password assignment	Subscription-based password assignment		

Optionally, the voice encryption security profile can be combined smoothly with the baseline security profile. Audio streams may be encrypted using the voice encryption security profile deploying DES, RC2-compatible or triple-DES and using the authenticated Diffie-Hellman key-exchange procedure.

The baseline security profile mandates the fast connect procedure with integrated key management elements. Signalling means are provided also for tunnelled H.245 key-update and synchronization. For long duration calls, these messages require tunnelling of H.245 within H.225.0 messages.

D.6.1.2 Voice encryption security profile

The voice encryption security profile is not an independent profile as is the baseline security profile. It is rather an option of the aforementioned security profile and may be used in conjunction with it. This profile also relies on certain security services as part of the call signalling and connection setup procedures; e.g., the Diffie-Hellman key agreement and other key management functions.

H.323 entities may implement the voice encryption profile for achieving voice confidentiality. Four encryption algorithms are offered: the suggested schemes are encryption using AES, RC2-compatible, DES or triple-DES based on the business model and exportability requirement. In addition to the CBC-encryption mode, H.323 entities may implement the EOFB stream-cipher encryption mode. Some environments that are already offering a certain degree of confidentiality may not require voice encryption. In this case, Diffie-Hellman key agreement and other key management procedures are not necessary.

---

4 The authentication-only security profile does not feature message integrity.

H.323 entities, when deploying the voice encryption security profile, shall implement 56-bit DES as the default encryption algorithm; they may implement 128-bit AES or 168-bit triple-DES or they may implement exportable encryption using 56-bit RC2-compatible.

The voice encryption profile is specified in clause D.2.

Table D.3/H.235 – Voice encryption profile

<table>
<tr><th rowspan="2">Security services</th><th colspan="4">Call functions</th></tr>
<tr><th>RAS</th><th>H.225.0</th><th>H.245</th><th>RTP</th></tr>
<tr><td>Authentication and integrity</td><td></td><td></td><td></td><td></td></tr>
<tr><td>Non-repudiation</td><td></td><td></td><td></td><td></td></tr>
<tr><td></td><td></td><td></td><td></td><td>56-bit DES | 56-bit RC2-compatible | 168-bit triple-DES | 128-bit AES</td></tr>
<tr><td>Confidentiality</td><td></td><td></td><td></td><td>CBC-mode or EOFB-mode</td></tr>
<tr><td>Access control</td><td></td><td></td><td></td><td></td></tr>
<tr><td>Key management</td><td></td><td>Authenticated Diffie-Hellman key-exchange</td><td>Integrated H.235 session key management (key distribution, key update)</td><td></td></tr>
</table>

D.6.2 Authentication and integrity

This annex uses the following terms for provisioning the security services.

- **authentication and integrity**: This is a combined security service part of the baseline profile that supports message integrity in conjunction with user authentication. The user may ensure authentication by correctly applying a shared secret key procedure. Both security services are provided by the same security mechanism.
- **Authentication-only**: This security service offered by the baseline security profile as an option supports authentication of selected fields only, but does not provide full message integrity. The authentication-only security profile is applicable for signalling messages traversing NAT/firewall devices. The user may ensure authentication by correctly applying a shared secret key procedure.

When using symmetric key techniques, the security services authentication/integrity only apply on a hop-by-hop basis.

D.6.3 H.323 requirements

H.323 entities that implement this baseline security profile are assumed to support the following H.323 features:

- Fast connect;
- GK-routed model.

D.6.3.1 Overview

The following procedure is described for use in this profile.

Procedure I is a simple symmetric-key-based signalling message authentication mechanism based on a shared password between two entities (e.g., Gatekeeper and H.323 endpoint). This procedure provides authentication and integrity of the RAS, Q.931 and H.245 messages (see D.6.3.2).

Procedure IA is a simple symmetric-key-based authentication-only mechanism based on a shared password between two entities (e.g., Gatekeeper and H.323 endpoint). This procedure provides only

authentication but does not provide full message integrity. The authentication-only option is applicable in scenarios where H.323 signalling messages traverse NATs/firewalls.

Depending on the security policy, authentication may be unilateral or mutual by applying the authentication/integrity in the reverse direction as well and thereby providing higher security. The Gatekeeper decides whether to apply authentication/integrity in the reverse direction as well.

Gatekeepers detecting failed authentication and/or failed integrity validation in a RAS or Call signalling message received from a secured endpoint or peer gatekeeper, respond with a corresponding reject message indicating security failure by setting the reject reason to **securityDenial**, or other appropriate security error code, according to B.2.2. Depending on the ability to recognize an attack, and the most appropriate way to react to it, a gatekeeper receiving a secured **xRQ** with undefined object identifiers (**tokenOID**, **algorithmOID**) may respond with an unsecured **xRJ** and reject with reason set to **securityDenial**, or it may discard that message. The encountered security event should be logged. On the other hand, the endpoint shall discard the received unsecured message, time out and may retry once again by considering to choose different OIDs. Likewise, a gatekeeper receiving a secured H.225.0 SETUP message with undefined object identifiers (**tokenOID**, **algorithmOID**) may respond with an unsecured RELEASE COMPLETE and reject with reason set to **securityDenied**, or may discard that message. Similarly, the encountered security event should be logged.

There is implicit H.235 signalling for indicating use of Procedure I and the applied security mechanism, based upon the value of the object identifiers (see also D.11) and the message fields filled in.

This profile does not use the H.235 ICV fields; rather cryptographic integrity check values are treated as cryptographic hash values and are put into the hash fields of the **CryptoToken**.

D.6.3.2 Symmetric-key-based signalling message authentication details (Procedure I)

The procedures below shall be followed when Procedure I is employed:

- The HMAC-SHA1-96 algorithm generates a 12-byte (96-bit) hash value as the resulting authenticator. If the key is generated from a password, the mechanism described in 10.3.5 *shall* be used for computing the key from the password.

  NOTE 1 – When the secret key is derived from a user-entered password, care should be taken to ensure sufficient randomness. It is recommended, for example, to use truly random secrets for the secret key, or to ensure that random passwords are sufficiently long.

- The **CryptoH323Token** field in each RAS/H.225.0 message shall contain the following fields:
  - **nestedCryptoToken** containing a **CryptoToken** which itself contains the **cryptoHashedToken** containing the following fields:
    - **tokenOID** set to "A", indicating that the authentication/integrity computation includes all fields in the H.225.0 RAS and call signalling message.
    - **hashedVals** containing the **ClearToken** field used with the following fields:
      - **tokenOID** set to "T", indicating that the baseline **ClearToken** as shown below is being used for message authentication and replay protection and optionally also for Diffie-Hellman key management as described in D.7.1. Alternatively, other ClearTokens with other OIDs may be used in place of the baseline ClearToken.
      - **timeStamp** contains the timestamp.
      - **random** contains a monotonically increasing sequence number. This number allows the construction of two messages with the same timestamp (within the clock resolution).

- **generalID** contains the identifier of the recipient (only in case of unicast messages).
- **sendersID** contains the identifier of the sender.
- **dhkey**, used to pass the Diffie-Hellman parameters as specified in this Recommendation during **Setup-**to-**Connect**.
  - **halfkey** contains the random public key of one party.
  - **modsize** contains the DH-prime (see Table D.4).
  - **generator** contains the DH-group (see Table D.4).

NOTE 2 – When the baseline security profile is used without the voice encryption security profile, then no Diffie-Hellman parameters should be sent and **dhkey** should be absent; **halfkey**, **modsize** and **generator** may be set to {'0'B,'0'B,'0'B}.

- **token** containing **HASHED** with the fields:
  - **algorithmOID** set to "U" indicating the use of HMAC-SHA1-96.
  - **paramS** set to NULL.
  - **hash** containing the authenticator computed using HMAC-SHA1-96. The authenticator can be computed over.
    - all the H.225.0 RAS and call signalling fields of the message if **tokenOID** in the **CryptoHashedToken** is set to "A" (indicating authentication and integrity).

tokenOID "A" is used for protection of tunnelled H323-UU-PDUs including all H.245 message contents; the hash computation shall be done over the entire **H.225.0** call signalling message with all fields according to the procedure described in D.6.3.3.2.

- The authenticator is verified at the end of each channel terminating leg (EP1-GK1, GK1-GK2, GK2-EP2, EP1-GK2, GK1-EP2 or EP1-EP2 as the case may be), and recomputed prior to sending the message out on the subsequent leg.

NOTE 3 – The authenticator is computed on a per-message basis.

NOTE 4 – The padding method within the SHA1 standard (ISO/IEC 10118-3) shall be used.

NOTE 5 – When the combined authentication and integrity is being used, the authenticator is computed over the entire message.

NOTE 6 – In order to prevent the possibility of replay attacks, it is highly recommended that implementations ensure that the password (key) is changed prior to a turn-around (or cycle completion) of the monotonically increasing sequence number.

NOTE 7 – The recipient is able to detect usage of Procedure I by evaluating the **tokenOID** within the hashed **EncodedGeneralToken** (detecting presence of "AB").

D.6.3.3 Computation of the password-based hash

Both sender and receiver of an authenticated/integrity protected message compute a keyed hash over all the ASN.1-coded message fields (using OID "A"). For the authentication-only profile, both sender and receiver compute a keyed hash over all the ASN.1 coded ClearToken (using OID "B").

D.6.3.3.1 HMAC-SHA1-96

HMAC-SHA1-96 is the truncated 96-bit cryptographic hash value of the 160-bit SHA1 computation. The 96 leftmost bits of the network byte order representation of the hash value shall be used as the result. RFC 2104 describes the procedure with the secret key *K* set to the shared secret (= SHA1-hashed password) and *text* set to the message buffer.

D.6.3.3.2 Authentication and integrity

For authentication and message integrity (in case OID "A" is applied), the procedure is as follows.

The sender of a message shall compute the hash as follows:

1) Set the hash value to a specific default pattern with a length of 96 bits. The exact bit pattern does not matter, but a good choice is a unique bit pattern that does not occur in the remaining message.
2) ASN.1 – encode the entire message; for RAS this shall include the entire H.225.0 RAS message; for call signalling, this shall include the entire H.225.0 call signalling message.
3) Locate[5] the default pattern in the encoded message; overwrite the found bit pattern all with 96 zero bits.
4) Compute the cryptographic hash value upon the ASN.1 encoded message using HMAC-SHA1-96 (see D.6.3.3.1).
5) Substitute the default pattern in the encoded message with the computed hash value.

The recipient receives the message and then proceeds as follows:

1) ASN.1 – decode the message.
2) Extract the received hash value and keep it in a local variable RV.
3) Search and locate the hash value RV in the received encoded message.

NOTE – In rare circumstances where the hash value substring might occur several times in the entire message, steps 3-6 have to be iterated successively with a different starting search position.

4) Overwrite the bit pattern in the encoded message all with 96 zeros.
5) Compute the cryptographic hash value upon the encoded message using HMAC-SHA1-96 (see D.6.3.3.1).
6) Compare RV with the computed hash value. The message is considered uncorrupted only if both hash values are equal; in this case, the authentication is successful and the procedure stops.
7) Otherwise, repeat steps 3-7 by restoring RV to the previous location and search for another match. If none of the matches yield a correct hash value comparison, then the authentication has failed and the message has been altered (accidentally or intentionally) during transit.

D.6.3.3.3 Authentication-only (Procedure IA)

Terminals may choose to implement authentication-only (using OID "B", see E.18). In this case, the authenticator is computed just over a subset (**ClearToken** inside **CryptoToken**) of the RAS/H.225.0 message. Authentication-only may be useful for traversing NAT/firewalls that change IP addresses/ports within the H.323 payloads.

Since the authentication spans only a very limited portion of the message, authentication-only does not provide message integrity as Procedure I features. Thus, authentication-only provides less security.

For authentication-only, the following fields shall be used in the protected messages:

- The **CryptoH323Token** field in each RAS/H.225.0 message shall contain the following fields:
  - **nestedCryptoToken** containing a **CryptoToken** which itself contains the **cryptoHashedToken** containing the following fields:

[5] This may involve some trial-and-error steps in the rare case when the default pattern occurs more than once in the message.

- **tokenOID** set to:
  - "B" (see E.18) indicating that the authentication-only computation includes all fields in the **ClearToken**.
- **hashedVals** containing the **ClearToken** field used with the following fields:
  - **tokenOID** set to:
    - "T" (as the baseline ClearToken example for the remainder of ClearToken contents) or any suitable OID for any other purposes.
  - **timeStamp** contains the timestamp;
  - **random** contains a monotonically increasing sequence number. This number allows the construction of two messages with the same timestamp (within the clock resolution);
  - **generalID** contains the identifier of the recipient (only in case of unicast messages);
  - **sendersID** contains the identifier of the sender;
  - **dhkey**, used to pass the Diffie-Hellman parameters as specified in H.235 during **Setup**-to-**Connect**.
    - **halfkey** contains the random public key of one party;
    - **modsize** contains the DH-prime (see Table D.4);
    - **generator** contains the DH-group (see Table D.4).

    NOTE 1 – When the baseline security profile is used without the voice encryption security profile, then no Diffie-Hellman parameters should be sent and **dhkey** should be absent; **halfkey**, **modsize** and **generator** may be set to {'0'B,'0'B,'0'B}.

– **token** containing **HASHED** with the fields:
  - **algorithmOID** set to "U" indicating the use of HMAC-SHA1-96;
  - **paramS** set to NULL;
  - **hash** containing the authenticator computed using HMAC-SHA1-96. The authenticator shall be computed over:
    - all the fields of the **ClearToken** if **tokenOID** in the **CryptoHashedToken** is set to "B" (indicating authentication-only).

- The authenticator is verified at the end of each channel terminating leg (EP1-GK1, GK1-GK2, GK2-EP2, EP1-GK2, GK1-EP2 or EP1-EP2 as the case may be), and recomputed prior to sending the message out on the subsequent leg.

NOTE 2 – The authenticator is computed just on the **ClearToken**.

NOTE 3 – The padding method within the SHA1 standard (ISO/IEC 10118-3) shall be used.

NOTE 4 – In order to prevent the possibility of replay attacks, it is highly recommended that implementations ensure that the password (key) is changed prior to a turn-around (or cycle completion) of the monotonically increasing sequence number.

NOTE 5 – The recipient is able to detect usage of Procedure IA by evaluating the **OID** "B" within the **tokenOID**.

The authenticator shall be computed just over the **ClearToken** inside the **CryptoH323Token** (i.e., **ClearToken**) of the **token** of the **cryptoHashedToken**. The cryptographic hash shall be computed over the ASN.1 encoded bitstring of **ClearToken**.

H.235 version 1 and version 2 endpoints may use authentication-only, in which case the corresponding OIDs for "B" shall be used. H.235 version 1 endpoints shall adhere to the procedure described in D.6.6.

D.6.3.4 Usage illustration for Procedure I

Figures D.1 through D.3 depict the presence of shared keys at the end of communicating channels for the different combinations of gatekeeper and direct-routed H.225.0 channels. Irrespective of the call model, a secret key is always present between an EP and its GK in order to provide for RAS message authentication and integrity. When a RAS channel and an H.225.0 channel terminate between the same two nodes, the same key may be used to provide authentication and integrity for both RAS and H.225.0 messages.

Figure D.1 shows the most scaleable scenario where both endpoints are within zones that apply the GK-routed model. All the involved GKs share keys mutually. In order to be scaleable, the scenario depicted in Figure D.1 is recommended.

NOTE 1 – This scenario does not provide true end-to-end security between endpoints; all security depends on the trusted intermediate gatekeepers.

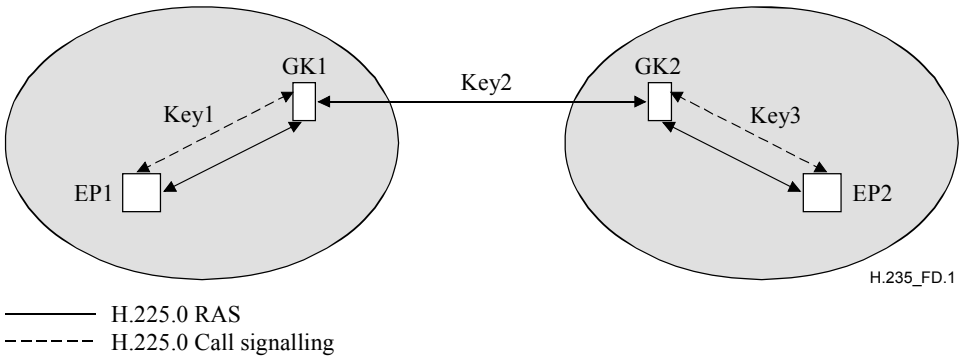


Figure D.1/H.235 – Illustrating Procedure I usage in a GK-GK scenario with both EPs in GK-routed zones

Figure D.2 shows a mixed scenario where one EP is within a zone applying the GK-routed model while the other EP is in a zone applying the direct-routed model. This scenario could occur in closed environments where the number of EP2s and GK1s is limited.

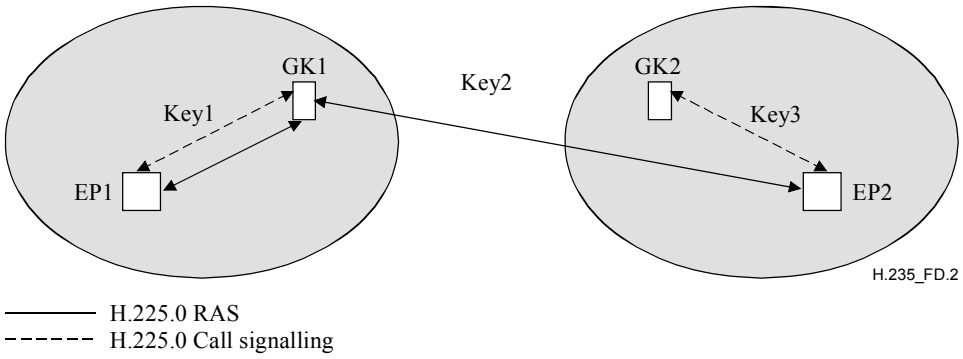


Figure D.2/H.235 – Illustrating Procedure I usage in a mixed scenario with EP1 in a GK-routed zone and EP2 in a direct-routed zone

Figure D.3 shows a scenario where both EPs are within zones applying the direct-routed GK model. This scenario is not very scaleable when many EPs are involved. In principle, usage of Annex E with Procedures II/III is recommended instead. For this specific scenario and Procedures I, II or III

additional security measures[6], which are not described in this Recommendation, are necessary as well; this is for further study.

NOTE 2 – This scenario provides true end-to-end security among endpoints without relying on trusted intermediate nodes.

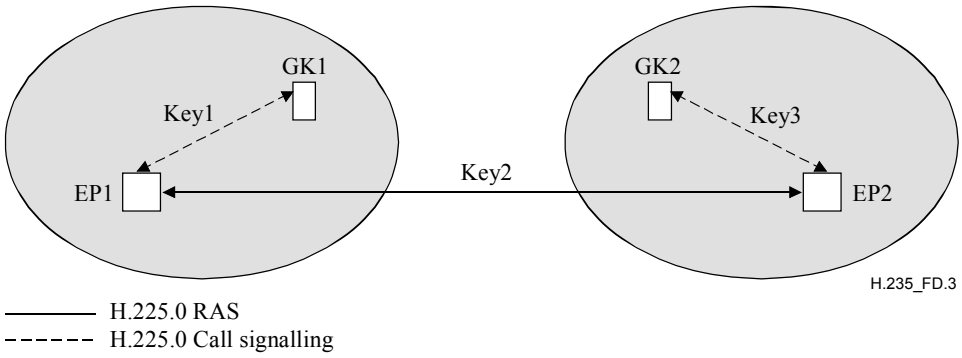


Figure D.3/H.235 – Illustrating Procedure I usage in a scenario with both EPs in zones using a direct-routed GK

Consider the case in Figure D.1 where three passwords are pair-wise shared between EP1-GK1, between GK1-GK2 and between GK2-EP2. Three 20-byte keys – *Key1*, *Key2* and *Key3* – are generated from these passwords based on the procedure described in 10.3.5. For maximum security it is recommended to make each of the three random passwords/keys independent.

Below, we illustrate the procedure details for RAS, H.225.0 and H.245 message authentication and integrity. The description example depicts specific parameters in a GK-routed model; other useful and valid combinations of object identifiers in different scenarios are possible as well.

NOTE 3 – The scenarios shown in the Figures 1 to 3 do not scale well in the case where the number of shared symmetric keys (passwords) between GKs (Figure D.1), between GKs and remote EPs (Figure D.2), or between the EPs (Figure D.3) becomes too large.

D.6.3.4.1 RAS message authentication and integrity

Consider the case where EP1 wishes to send a RAS message, say an **ARQ** message, to GK1. EP1 generates a timestamp and a sequence number and includes it in the **timeStamp** and **random** fields respectively, along with GK1's alias in the **generalID** and the EP's ID in the **sendersID** field. These fields are present in the **ClearToken** field of **hashedVals** present in the **cryptoHashedToken** of the **CryptoToken** field of the **cryptoH323Token** of the **ARQ** message.

The **tokenOID** within the **cryptoHashedToken** is set to "A", indicating that all the fields in the **ARQ** message are hashed. The **HASHED** within **token** in **cryptoHashedToken** has **algorithmOID** set to "U" indicating the use of HMAC-SHA1-96 and **paramS** set to NULL. EP1 then computes the authenticator based on the HMAC-SHA1-96 using the 20-byte key *Key1*. The authenticator is computed over the entire RAS message.

EP1 includes the computed authenticator within **hash** in the **token** field of the **cryptoHashedToken** field of the **CryptoToken** present in the **cryptoH323Token** of the **ARQ** message. The **ARQ** message is then sent to GK1.

Upon receiving the **ARQ** message, GK1 verifies the authenticator based on several criteria that include:

---

6 Protecting against call fraud and misuse by means of call authorization with access tokens at H.323 gateways for example.

- liveness of the **timeStamp**, uniqueness of the **random**;
- identity of the **generalID** and own identifier;
- matching of authenticator in **ARQ** message with that computed by GK1.

D.6.3.4.2 H.225.0 message authentication and integrity

Consider the case where EP1 wishes to send an H.225.0 message, say a **Setup** message, to EP2. EP1 generates a timestamp and a sequence number and includes it in the **timeStamp** and **random** fields respectively, along with GK1's alias in the **generalID** and the EP's ID in the **sendersID** field. EP1 computes also a Diffie-Hellman half-key and includes the Diffie-Hellman parameters **halfkey**, **modsize** and **generator** in the **dhkey** field of the **ClearToken**. These fields are present in the **ClearToken** field of **hashedVals** present in the **cryptoHashedToken** of the **CryptoToken** field of the **cryptoH323Token** of the **Setup** message.

The **tokenOID** within the **cryptoHashedToken** is set to "A", indicating that all the fields in the H.225.0 call signalling message are hashed. The **HASHED** within **token** in **cryptoHashedToken** has **algorithmOID** set to "U" indicating the use of HMAC-SHA1-96 and **paramS** set to NULL. EP1 then computes the authenticator based on the HMAC-SHA1 algorithm using the 20-byte key *Key1*. The authenticator is computed according to the hash method chosen (A) taking into account the entire H.225.0 call signalling message.

EP1 includes the computed authenticator within **hash** in the **token** field of the **cryptoHashedToken** field of the **CryptoToken** present in the **cryptoH323Token** of the **Setup** message. The **Setup** message is then sent to GK1.

Upon receiving the **Setup** message, GK1 verifies the authenticator based on several criteria that include:

- liveness of the **timeStamp**, uniqueness of the **random**;
- identity of the **generalID** and own identifier;
- verification of Diffie-Hellman parameters, e.g., testing whether the 1024-bit prime and generator are correct. Testing of whether the DH-parameters are secure is a time-consuming process and may be done only when local policy requires it;
- matching of authenticator in **Setup** message with that computed by GK1.

If the authenticator is successfully verified, GK1 computes a new authenticator to insert (replace) in the **Setup** message before forwarding it to GK2 as follows. GK1 replaces the **timeStamp**, **random**, **sendersID** and **generalID** fields in the **ClearToken** field of **hashedVals** using values relevant to the GK1-GK2 leg. The **timeStamp** field contains the current timestamp, the **random** field contains the next monotonically increasing sequence number for the GK1-GK2 leg, the **generalID** field contains the alias of GK2 and the **sendersID** contains the alias of GK1. GK1 includes also the received Diffie-Hellman parameters into the **dhkey** field of the **ClearToken**.

GK1 then computes a new authenticator for this H.225.0 call signalling message using key *Key2* and algorithm HMAC-SHA1-96 (**algorithmOID**="U"), inserts it in **hash** within **token** and passes the **Setup** message on to GK2.

Upon receiving the **Setup** message, GK2 verifies the authenticator, computes a new authenticator after modifying the **ClearToken** fields in **hashedVals** suitably, inserts it in the **hash** field and passes the **Setup** message on to EP2.

D.6.3.4.3 H.245 message authentication and integrity

Consider the case where EP1 wishes to send an H.245 message, say a **TerminalCapabilitySet** message, to EP2. EP1 checks to see if an H.225.0 message needs to be sent to GK1. If so, then the H.245 message is tunnelled within that H.225.0 message. The fields within the H.225.0 message are

set as described earlier for the transmission of an H.225.0 message. Since the H.245 message is tunnelled, the **h323-uu-pdu** in the **h323-UserInformation** message has its fields set as follows:

- **h323-message-body** field is set to the H.225.0 message type that is being transmitted.
- **h245Tunnelling** set to TRUE.
- **h245Control** contains the H.245 PDU octet string.

EP1 generates a **CryptoToken** for the H.225.0 message, sets **tokenOID** to "A", indicating authentication and integrity, sets **timeStamp**, **random**, **sendersID**, **generalID** and **tokenOID** to "T" in the **ClearToken** of the **hashedVals**, set **algorithmOID** to "U", indicating the use of HMAC-SHA1-96 and **hash** to the computed hash authenticator over all the fields of the H.225.0 call signalling message.

However, if no H.225.0 message transmission is pending, then the H.245 message is tunnelled within an ad hoc H.225.0 **facility** message. The **h323-uu-pdu** in the **h323-UserInformation** message has its fields set as follows:

- **h323-message-body** field is set to **facility** which contains:
  - **reason** set to **undefinedReason**;
  - **tokens** and **cryptoTokens** set as for any H.225.0 message.
- **h245Tunnelling** set to TRUE.
- **h245Control** contains the H.245 PDU octet string.

As described above, EP1 generates a **CryptoToken** as part of the H.225.0 **facility** message. The **facility** message is then transmitted by EP1 to GK1.

In either case (whether a H.225.0 message transmission is pending or an ad hoc H.225.0 **facility** message is used), GK1 verifies the authenticator upon receiving the message. Then, if an H.225.0 message transmission is pending for the GK1-GK2 leg, the H.245 message is tunnelled within that message; otherwise, it is tunnelled within an ad hoc H.225.0 **facility** message. As in the case of transmission of any H.225.0 message, a new authenticator is computed for the H.225.0 message prior to its transmission from GK1 to GK2. The process repeats for the GK2-EP2 leg.

D.6.4 Direct-routed scenario

Secured H.323 entities may communicate not only within the GK-routed environment as outlined in this Recommendation, but may also deploy the direct-routed model. This direct-routed model requires additional security measures (access tokens) that are not necessary in the simpler GK-routed environments. Securing the direct-routed model is thus for further study.

D.6.5 Back-end-service support

Secured H.323 entities may use back-end services according to the procedure described in I.4.6.

D.6.6 H.235 version 1 compatibility

While these security profiles are developed with H.235 version 2 (ITU-T Rec. H.235 (2000)) in mind, it is also possible to apply the security profiles for H.235 version 1 (ITU-T Rec. H.235 (1998)) with some minor modifications. A recipient is able to detect the presence of the sender's H.235 protocol version by evaluating the security profile object identifiers (see D.11).

H.235 version 1 (ITU-T Rec. H.235 (1998)) implementations:

- do not set or evaluate the **sendersID** in the **ClearToken**.
- cannot use back-end services as in D.6.5.

D.6.7 Multicast behaviour

H.225.0 multicast messages such as GRQ or LRQ shall not include a CryptoToken according to Procedure I. When such messages are sent unicast, then the message shall include a CryptoToken.

D.7 Voice encryption security profile

The general procedure establishes a shared secret (Diffie-Hellman exchange) between the two communicating parties at connection initiation. This shared secret is then used to protect (a set of) media keys that are used to encrypt the media (RTP) sessions.

The voice encryption security profile is an optional enhancement to the baseline security profile and to the signature security profile; its use can be negotiated as part of the terminal security capability negotiation. In environments where voice confidentiality is assured by other means, there is no need to implement the media encryption and the related key management procedures (Diffie-Hellman key agreement, key update and synchronization).

The encryption algorithms chosen are AES, RC2-compatible, DES and triple-DES.

NOTE – Since an implementation of triple-DES can also be used for the DES algorithm, this results in a compact implementation.

Irrespective of the choice of the specific media encryption algorithm, the options below shall be followed explicitly.

- Initialization Vector (IV) generated, if needed, as specified in B.3.1.
- Padding, if needed, is to occur as described in B.3.2.

The audio payload shall be encrypted using the negotiated encryption algorithm ("X", "Y", "Z3" or "Z") according to the procedures described in clause 11 and in Annex B, and the ciphertext padding methods of I.1. The audio payload may be encrypted using the negotiated encryption algorithm ("X1", "Y1", "Z1" or "Z2") operating in a stream cipher mode (EOFB).

D.7.1 Key management

Endpoints conforming to this annex should use the fast connect procedure according to 8.6.1. If fast start is not applied, then H.245 tunnelling shall be used to secure the H.245 call control messages by this annex. The fast start procedures allow the establishment of either one or two unidirectional logical channels. The fast start procedure cares for negotiation of the security capabilities, for distribution of a common shared secret (shared DH secret) which acts as a master key, and for secure distribution of an encryption key.

Table D.4 provides the allocated OIDs for the various encryption algorithms and relates them with the allocated OIDs for the Diffie-Hellman group. Three DH groups are identified through an OID:

- "DHdummy": An instance of this DH group should be applied whenever exportable (512 bit) security is of concern or any or non-standard DH group is being used.

  NOTE 1 – No particular DH group is defined; the OID references any non-standard DH group.
- An instance of a 512-bit DH group shall be used to generate a master key for distribution of session key(s) for RC2-compatible ("X") or for DES-56 bit encryption algorithms ("Y").
- "DH1024": This DH group is to be applied when high (1024 bit) security is of concern. The OID references a standardized, fixed DH group. This DH group shall be used to generate a master key for distribution of session key(s) for triple-DES ("Z") encryption algorithms.
- "DH1536": This DH group is offered as an option for version 3 endpoints having very high security requirements that exceed the security of a 1024-bit DH group. The OID references a fixed DH group. This DH group shall be used to generate a master key for distribution of session key(s) for triple-DES ("Z", "Z1") or for AES-128 ("Z2", "Z3") encryption algorithms.

It is recommended to apply the defined 1024- or optionally, 1536-bit DH groups unless other security needs would make other Diffie-Hellman parameters preferential. Further, it is recommended to consider using the defined OIDs identifying the DH groups, see 8.8. Nevertheless, implementations should be prepared to obtain the DH group parameters literally without explicit OID indication. In this case, implementations should ascertain that the correct DH group is being conveyed according to Table D.4.

Endpoints may use non-standard DH group parameters. Using OID "DHdummy" should indicate such non-standard DH groups. It is left to the decision of the callee whether to accept such DH groups.

NOTE 2 – The choice of the DH group does not eliminate the need to negotiate the actual media encryption algorithm. This shall be accomplished with the H.245 Terminal capability negotiation procedure.

NOTE 3 – During connection establishment (SETUP-to-CONNECT) usage of the encryption algorithm OIDs shall not be used to indicate a Diffie-Hellman instance.

Table D.4/H.235 – Diffie-Hellman groups

Encryption Algorithm OID	DH-OID	D-H group description
"X", "X1" (RC2-compatible), "Y", "Y1" (DES)	"DHdummy"	Mod-P, any suitable 512-bit prime
"Z", "Z1" (triple-DES), "Z2", "Z3" (AES)	"DH1024"	Mod-P, 1024-bit prime Prime = $2^{1024} - 2^{960} - 1 + 2^{64} \times \{ [2^{894} \text{ pi}] + 129093 \}$ = (17976931348623159077083915679378745319786029604875601170644442368419718021615851936894783379586492554150218056548598050364644054819923910005079287700335581663922955313623907650873575991482257486257500742530207744771258955095793777842444242661733472762929938766870920560605027081084290769293201912819446762700 7)$_{10}$ Generator (Note) = 2
"Z", "Z1" (triple-DES), "Z2", "Z3" (AES)	"DH1536"	Mod-P, 1536-bit prime Prime = $2^{1536} - 2^{1472} - 1 + 2^{64} \times \{ [2^{1406} \text{ pi}] + 741804 \}$ = (24103124269210325885520760221975660748569505485024599426541169419581088316826122288900938582613416146732271414779040121965036489570505826319427307068050092230627347453410734066962460145893616597740410271692494532003787294341703258437786591981437631937768598695240889401955773461198435453015470437472077499697637500843089263392955599688824578724129938101291302945929999479263652640592846472097303849472116814344647144384885209401274598442888593365268963209196339 19)$_{10}$ Generator (Note) = 2
NOTE – The generator is used to generate the DH token.		

D.7.2 Key update and synchronization

For 64-bit block ciphers, the key refresh rate *shall* be such that no more than 2^{32} blocks are encrypted using the same key. Implementations *should* refresh keys before 2^{30} blocks have been encrypted using the same key (see 11.1). For 128-bit block ciphers, the key refresh rate *shall* be such that no more than 2^{64} blocks are encrypted using the same key. Implementations *should* refresh keys before 2^{62} blocks have been encrypted using the same key (see 11.1). Both involved entities are free to change the media session key as often as considered necessary due to their security policy. For example, the master may distribute a new session key using **encryptionUpdate** or **encryptionUpdateCommand** of the **miscellaneousCommand** message. On the other hand, the slave can request a new session key from the master by using the **encryptionUpdateRequest** of the **miscellaneousCommand** message; see also B.2.6.

The **MiscellaneousCommand** message contains the **encryptionUpdate** and **encryptionUpdateCommand** of which the **encryptionSynch** is set with the following parameters:

- **synchFlag**: the new dynamic RTP payload number indicating key changeover.
- **h235key**: carrying the new encrypted session key. This is an H.235 ASN.1 encoded **H235Key** passed as an octet string.

The **sharedSecret** field within the **H235Key** structure uses the following fields:

- **algorithmOID**: set to "X", "X1" for the 56-bit RC2-compatible, set to "Y", "Y1" for 56-bit DES or set to "Z", "Z1" for 168-bit triple-DES or set to "Z3" for 128-bit AES.

  NOTE 1 – The session key encryption algorithm is the same as the negotiated media encryption algorithm.
- **paramS**: set to the initial value. For 64-bit block stream ciphers, **iv8** holds a random 64-bit block bit pattern that the initiator generates. For 128-bit block stream ciphers, **iv16** holds a random 128-bit block bit pattern that the initiator generates. This field shall not be used for the CBC mode and shall be set to NULL, meaning that the CBC-IV for session key encryption shall be set to 0; it shall only be used for carrying the IV for EOFB mode.
- **encryptedData**: set to the result of the encrypted **KeySyncMaterial**.

As part of the **KeySyncMaterial**:

- **generalID**: identifier of the source distributing the key.

  NOTE 2 – This Recommendation assumes that each endpoint has registered with a gatekeeper and has obtained an endpoint identifier that can be conveyed within **generalID**. This Recommendation does not support scenarios without gatekeepers; this is left as for further study.
- **keyMaterial**: set to the new session key. For DES and RC2-compatible this is a 56-bit key, for triple-DES this is a 168-bit key and for AES this is a 128-bit key. The master shall generate a new session key that meets at least the following security criteria: it is not a weak or semi-weak DES-key and uses a sufficiently secure random source.

The **MiscellaneousCommand** message contains the **encryptionUpdateRequest** that contains **keyProtectionMethod** where the flag **sharedSecret** is set to TRUE.

NOTE 3 – Since the key update and synchronization relies on H.245 messages that are not piggy-backed during fast connect, this requires H.245 tunnelling to be used for secured H.323 entities.

D.7.3 Triple-DES in outer CBC mode

168-bit triple-DES in outer CBC mode, as illustrated in Figure D.4, s*hould* be used within this security profile. In the figure, each k_i refers to a 56-bit key. A different 56-bit key *shall* be used within each encryption (E) and decryption (D) block. None of the 64 weak keys for DES are known to cause any weakness within triple-DES. However, implementations complying with this profile should reject the key when a weak DES key is involved (see RFC 2405).

More information on triple-DES may be obtained from [Schneier] and RFC 2405.

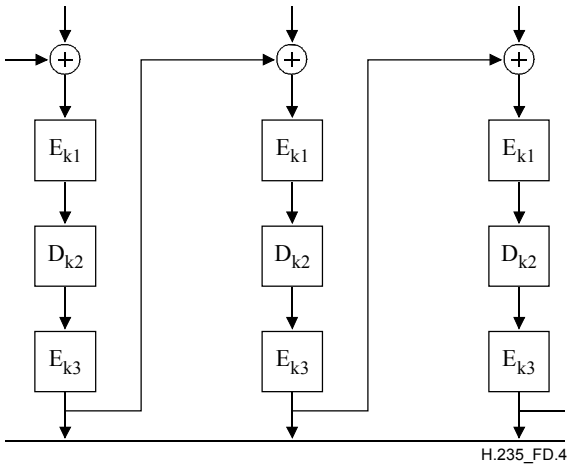


Figure D.4/H.235 – Triple-DES encryption in outer CBC mode

D.7.4 DES algorithm operating in EOFB mode

Voice may be encrypted using the DES algorithm operating in the EOFB stream cipher block-chaining mode. EOFB mode allows exploiting parallelism in implementations. When operating in EOFB mode, it is recommended, for both performance and security reasons, to feedback the entire crypto block (i.e., the full 64-bits for DES for example with $n = j = 64$). However, due to the fact that EOFB does not provide chaining across the blocks and bits, EOFB may be susceptible to specific attacks depending on the statistical properties of the input plaintext data. Thus, key updating (see D.7.2) should be performed regularly but, at latest, before the initial value wraps around. For the computation of the initial value see B.3.1.2.

D.7.5 Triple-DES in outer EOFB mode

168-bit triple-DES in outer EOFB mode, as illustrated in Figure D.5, may be used within this security profile. In the figure, each k_i refers to a 56-bit key. A different 56-bit key *shall* be used within each encryption (E) and decryption (D) block. None of the 64 weak keys for DES are known to cause any weakness within triple-DES. However, implementations complying with this profile should reject the key when a weak DES key is involved [RFC 2405].

More information on triple-DES may be obtained from [Schneier] [RFC 2405].

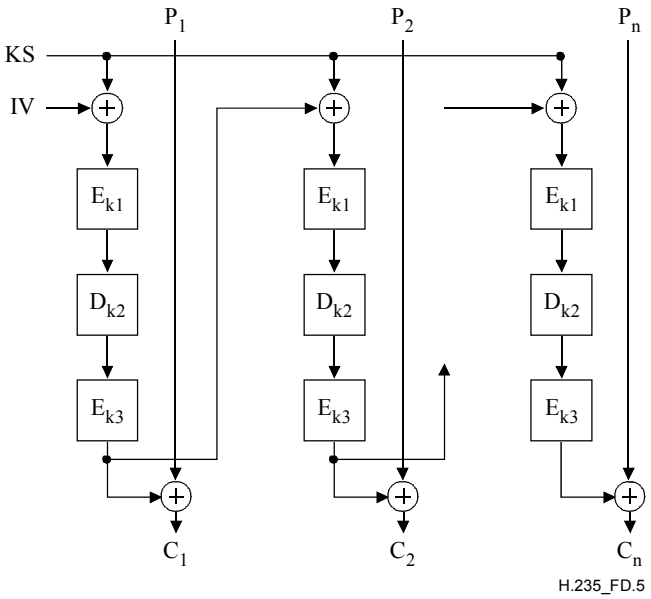


Figure D.5/H.235 – Triple-DES encryption in outer EOFB mode

D.8 Lawful interception

For further study (see [LI]).

D.9 List of secured signalling messages

This clause provides a summary of how, and by which means, Annex D secures the various H.323 signalling messages.

D.9.1 H.225.0 RAS

H.225.0 RAS message	H.235 signalling fields	Authentication and integrity
Any	cryptoTokens	Procedure I

D.9.2 H.225.0 call signalling

H.225.0 call signalling message	H.235 signalling fields	Authentication and integrity
Alerting-UUIE, CallProceeding-UUIE, Connect-UUIE, Setup-UUIE, Facility-UUIE, Progress-UUIE, Information-UUIE, ReleaseComplete-UUIE, Status-UUIE, StatusInquiry-UUIE, SetupAcknowledge-UUIE, Notify-UUIE	cryptoTokens	Procedure I

D.9.3 H.245 call control

H.245 messages to and from secured H.323 entities shall either be piggy-backed as part of the secured fast-connect, or shall be tunnelled using the secured H.225.0 **Facility-UUIE**.

D.10 Usage of sendersID and generalID

The **ClearToken** holds **sendersID** and **generalID** fields. When identification information is available, the **sendersID** shall be set to the gatekeeper identifier (GKID) for the gatekeeper-initiated message and to the endpoint identifier (EPID) for the endpoint-initiated messages. When identification information is available, the **generalID** shall be set to the GKID for endpoint-initiated

messages and to EPID for the gatekeeper-initiated messages. When the identification information is not available, or in case of broadcast/multicast is ambiguous, the field is missing or shall contain a null string. Table D.5 summarizes the situation.

Table D.5/H.235 – Object identifiers used by Annex D

Message	sendersID	generalID
Unicast **GRQ**	**EPID** if available, otherwise **NULL**	**GKID**
Multicast **GRQ**	**EPID** if available, otherwise **NULL**	
GCF, **GRJ**	**GKID**	**EPID** if available, otherwise **NULL**
Initial **RRQ**	**EPID** if available, otherwise **NULL**	**GKID**
RCF	**GKID**	**EPID**
RRJ	**GKID**	
URQ, UCF, URJ, BRQ, BCF, BRJ, DRQ, DCF, DRJ, NSM, RIP, SCI, SCR, XRS (EP-to-GK)	**EPID**	**GKID**
URQ, UCF, URJ, BRQ, BCF, BRJ, DRQ, DCF, DRJ, NSM, RIP, SCI, SCR, XRS (GK-to-EP)	**GKID**	**EPID**
ARQ, IRQ, RAI	**EPID**	**GKID**
ACF, ARJ, BCF, LCF, LRJ, IRR, IRQ, RAC, LCF, LRJ, IACK, INAK	**GKID**	**EPID**
Unicast **LRQ** (EP-to-GK)	**EPID**	**GKID**
Unicast **LRQ** (GK-to-GK)	**GKID**	**GKID**
Multicast **LRQ**	**EPID**	
NOTE – GKID stands for gatekeeper identifier, EPID stands for endpoint identifier. Blank indicates a missing or null identification string.		

D.11 List of object identifiers

Table D.6 lists all the referenced OIDs (see also [OIW] and [WEBOIDs]). There are object identifiers for H.235v1 [H.235v1] and for H.235v2 [H.235v2].

Table D.6/H.235 – Object identifiers used by Annex D

Object identifier reference	Object identifier value(s)	Description
"A"	{itu-t (0) recommendation (0) h (8) 235 version (0) 2 1} {itu-t (0) recommendation (0) h (8) 235 version (0) 1 1}	Used in Procedure I for the CryptoToken-tokenOID, indicating that the hash includes all fields in the H.225.0 RAS and call signalling message (authentication and integrity).
"E"	{itu-t (0) recommendation (0) h (8) 235 version (0) 3 9} {itu-t (0) recommendation (0) h (8) 235 version (0) 2 9}	End-to-end ClearToken carrying sendersID for verification at the recipient side.
"T"	{itu-t (0) recommendation (0) h (8) 235 version (0) 2 5} {itu-t (0) recommendation (0) h (8) 235 version (0) 1 5}	Used in Procedures I and IA as the baseline ClearToken for the message authentication and replay protection and optionally also for Diffie-Hellman key management as described in D.7.1.
"U"	{itu-t (0) recommendation (0) h (8) 235 version (0) 2 6} {itu-t (0) recommendation (0) h (8) 235 version (0) 1 6}	Used in Procedure I for the Algorithm OID, indicating use of HMAC-SHA1-96.
"DHdummy"	{itu-t (0) recommendation (0) h (8) 235 version (0) 2 40} {itu-t (0) recommendation (0) h (8) 235 version (0) 3 40}	Non-standard DH-group explicitly provided
"DH1024"	{itu-t (0) recommendation (0) h (8) 235 version (0) 2 43} {itu-t (0) recommendation (0) h (8) 235 version (0) 3 43}	1024-bit DH group
"DH1536"	{itu-t (0) recommendation (0) h (8) 235 version (0) 3 44}	1536-bit DH group
"X"	{iso(1) member-body(2) us(840) rsadsi(113549) encryptionalgorithm(3) 2}	Voice encryption using RC2-compatible (56-bit) or RC2-compatible in CBC mode and 512-bit DH-group.
"X1"	{itu-t (0) recommendation (0) h (8) 235 version (0) 3 27}	Voice encryption using RC2-compatible (56-bit) or RC2-compatible in EOFB mode and 512-bit DH-group
"Y"	{iso(1) identified-organization(3) oiw(14) secsig(3) algorithm(2) descbc(7)}	Voice encryption using DES (56-bit) in CBC mode and 512-bit DH-group.
"Y1"	{itu-t (0) recommendation (0) h (8) 235 version (0) 3 28}	Voice encryption using DES (56-bit) in EOFB mode and 512-bit DH-group with 64-bit feedback

Table D.6/H.235 – Object identifiers used by Annex D

Object identifier reference	Object identifier value(s)	Description
"Z1"	{itu-t (0) recommendation (0) h (8) 235 version (0) 3 29}	Voice encryption using triple-DES (168-bit) in outer-EOFB mode and 1024-bit DH-group with 64-bit feedback
"Z2"	{itu-t (0) recommendation (0) h (8) 235 version (0) 3 30}	Voice encryption using AES (128-bit) in EOFB mode and 1024-bit DH-group
"Z3"	{joint-iso-itu-t(2) country(16) us(840) organization(1) gov(101) 3 nistAlgorithm(4) aes(1) cbc(2)}	Voice encryption using AES (128-bit) in CBC mode and 1024-bit DH-group
"Z"	{iso(1) identified-organization(3) oiw(14) secsig(3) algorithm(2) desEDE(17)}	Voice encryption using triple-DES (168-bit) in outer-CBC mode and 1024-bit DH-group.

D.12 Bibliography

[FIPS PUB 180-1] NIST, FIPS PUB 180-1: Secure Hash Standard, April 1995 http://csrc.nist.gov/fips/fip180-1.ps.

[LI] Draft DRT/TIPHON-08003 V0.0.9, "Lawful Interception – Internal LI Interface", August 2000.

[OIW] Stable Implementation – Agreements for Open Systems Interconnection Protocols: Part 12 – OS Security; Output from the December 1994 Open Systems Environment Implementors' Workshop (OIW); http://nemo.ncsl.nist.gov/oiw/agreements/stable/OSI/12s_9412.txt.

[RFC 2405] C. Madson, N. Doraswamy "The ESP DES-CBC Cipher Algorithm With Explicit IV", RFC 2405, Internet Engineering Task Force, 1998.

[WEBOIDs] http://www.alvestrand.no/objectid/top.html.

Annex E

Signature security profile

E.1 Overview

This annex describes a security profile deploying digital signatures that is suggested as an option. H.323 security entities (terminals, gatekeepers, gateways, MCUs, etc.) may implement this signature security profile for improved security or whenever required.

The signature security profile mandates the GK-routed model and is based upon the H.245 tunnelling techniques; support for non GK-routed models is for further study.

The signature security profile is applicable for scaleable "global" IP telephony; this security profile overcomes the limitations of the simple, baseline security profile of Annex D. For example, the

signature security profile does not depend on the administration of mutual shared secrets of the hops in different domains. It provides tunnelling of H.245 messages for H.245 message integrity and also provisions for non-repudiation of messages. The signature security profile supports hop-by-hop security as well as true end-to-end authentication with simultaneous use of H.235 proxies or intermediate gatekeepers.

The features provided by these profiles include, for RAS, H.225.0 and H.245 messages:

- User authentication to a desired entity irrespective of the number of application level hops[7] that the message traverses.
- Integrity of all or critical portions (fields) of messages arriving at an entity irrespective of the number of application level hops that the message traverses. Integrity of the message itself using a strongly generated random number is also optional.
- Application level hop-by-hop message authentication, integrity and non-repudiation provide these security services for the entire message.
- Non-repudiation of messages exchanged between two entities irrespective of the number of application level hops that the message traverses can also be provided. Specifically, the non-repudiation is provided for critical portions (fields) of the message. For instance, this may be the case when an EP sends a SETUP message to its GK and the two (EP and GK) are separated by one or more proxies.

Several attacks are thwarted by providing the above security services in a suitable fashion. These include:

- Denial-of-service attacks: Rapid checking of digital signatures can prevent such attacks.
- Man-in-the-middle attacks: Application level hop-by-hop message authentication and integrity prevents against such attacks when the man in the middle is between an application level hop, say, a hostile router. When the man in the middle is an application level entity, such attacks are prevented by the presence of end-to-end user authentication and integrity for selected portions of the message.
- Replay attacks: Use of timestamps and sequence numbers prevent such attacks.
- Spoofing: User authentication prevents such attacks.
- Connection hijacking: Use of authentication/integrity for each signalling message prevents such attacks.

E.2 Specification conventions

The signature security profile may use the **voice encryption security profile** of Annex D for achieving voice confidentiality if necessary.

Procedures II and III specify how to implement the security services for different scenarios as hop-by-hop and end-to-end with different security mechanisms such as asymmetric cryptographic (digital signature) techniques.

While the message integrity service always provides message authentication, the reverse is not always true. For the authentication-only mode, the integrity assured spans only a certain subset of message fields. This applies to integrity services realized by asymmetric means (e.g., digital signatures). Thus, in practice, a combined authentication and integrity service uses the same key material without introducing a security weakness.

---

[7] "Hop" is understood here in the sense of a trusted H.235 network element (e.g., GK, GW, MCU, proxy, firewall). Thus, application level hop-by-hop security when used with symmetric techniques does not provide true end-to-end security between terminals.

Moreover, all hop-by-hop security information is put into the **CryptoSignedToken** element. This information is recomputed at every hop according to Procedure II.

End-to-end security information on the other hand (only possible when using the H.323 proxy and Procedure III), basically computes similar information as put in the **CryptoSignedToken**, but stores that information in a separate **CryptoToken** of the message. This information is not changed in transit. A separate object identifier allows distinguishing between hop-by-hop and end-to-end **CryptoTokens**.

Certification Authorities: Certification Authorities (CAs), when used in the context of electronic signature, certify public verification keys by issuing "Certificates".

Certificate Repositories: Certificate repositories (e.g., an X.500 Directory) hold user certificates and Certificate Revocation Lists (CRLs). They are trusted to make that information accessible but are not responsible for the content or accuracy of the information they receive from the CAs or the RAs.

Digital signature: Is a cryptographic transformation (using an asymmetric cryptographic technique) of the numerical representation of a data message, such that any person having the signed message and the relevant public key can determine that:

i) the transformation was created using the private key corresponding to the relevant public key; and

ii) the signed message has not been altered since the cryptographic transformation.

On-line Certificate Status Providers: The On-line Certificate Status Protocol (OCSP) enables applications to determine the revocation state of an identified certificate. OCSP may be used to satisfy some of the operational requirements of providing revocation information in a more timely way than is possible with CRLs. On-line certificate status providers can be seen as an alternative to the use of off-line CRLs.

Proxy: The proxy is an intermediate H.323 entity similar to a gatekeeper. The proxy may be a separate network node or may be collocated with the functionality of an H.323 entity such as of the gatekeeper. The proxy may perform security tasks such as signature and certificate verification and access control.

Registration Authorities: Registration authorities act as intermediaries between users and CAs. They receive requests from users and transmit them to the CAs in an appropriate form.

Time Stamping Authorities: Time stamping authorities are mandatory for non-repudiation in case of key loss or key compromise. In practice, they provide a counter-signature to anyone, including a reliable time, over a hash and a hash identifier.

Trust Service Provider: An entity, which can be used by other entities as a trusted intermediary in a communication or verification process, or as a trusted information service provider.

The signature security profile is suggested as an option. This security profile is applicable in environments with potentially many terminals where password/symmetric key assignment is not feasible, e.g., in large-scale or global-scale scenarios. The signature security profile provides additional security services for non-repudiation using digital signatures and certificates. The digital signatures could use SHA1 or MD5 hashing and provides authentication and/or integrity (see Procedures II and III).

H.323 entities using authentication and integrity, or authentication-only on a hop-by-hop basis, shall use Procedure II. H.323 entities using just authentication-only would not implement integrity. The authentication-only H.323 entities shall use Procedure III for true end-to-end authentication.

This annex may apply message integrity protection that spans the entire message. For H.225.0 RAS the integrity protection covers the entire RAS message; for call signalling this covers the entire H.225.0 call signalling message including the Q.931 headers.

The signature security profile allows to securely tunnel H.245 call control PDUs within H.225.0 facility messages. The H.245 key update and synchronization mechanisms require tunnelling, e.g., useful for very long duration calls[8].

The vertically shaded area (yellow in the electronic copy) in Table E.1 represents the scope of the signature security profile. When omitting the integrity indicated by the horizontally shaded area (blue in the electronic copy), the authentication-only security profile results. An option within the signature security profile is to choose between RSA-SHA1 or RSA-MD5 digital signatures. The voice encryption security profile of Annex D (see clause D.7) could be optionally used in conjunction to the signature security profile.

Table E.1/H.235 – Signature security profile

Security services	Call functions			
	RAS	H.225.0	H.245 (Note)	RTP
Authentication	SHA1/ MD5	SHA1/ MD5	SHA1/ MD5	
	digital signature	digital signature	digital signature	
Non-repudiation	SHA1/ MD5	SHA1/ MD5	SHA1/ MD5	
	digital signature	digital signature	digital signature	
Integrity	SHA1/ MD5	SHA1/ MD5	SHA1/ MD5	
	digital signature	digital signature	digital signature	
Confidentiality				
Access control				
Key management	certificate allocation	certificate allocation		
NOTE – Tunnelled H.245 or embedded H.245 inside H.225.0 fast connect.				

NOTE 1 – The signature security profile has to be supported also by other H.235 entities (e.g., gatekeepers, gateways and H.235 proxies).

NOTE 2 – Available key usage bits in the certificate could also determine the security service provided by a terminal (e.g., non-repudiation asserted).

For authentication, the user should use a public/private key signature scheme. Such a scheme usually provides for better integrity and non-repudiation of the call.

This Recommendation does not describe procedures for:

- Registration, certification and certificate allocation from a trust centre and private/public key assignment, directory services, specific CA parameters, certificate revocation, key pair

[8] Key-update for secure G.711 speech coding should occur latest after transmission of 2^{30} 64-bit blocks, i.e., more than 12 days of ongoing conversation.

update/recovery and other certificate operational or management procedures such as certificate or public/private key and certificate delivery and installation in terminals.

Such procedures may happen by means that are not part of this annex.

The communication entities involved are able to implicitly determine usage of either the Annex D baseline security profiles or this signature security profile by evaluating the signalled security object identifiers in the messages (**tokenOID**, and **algorithmOID**; see also E.18).

E.3 H.323 requirements

H.323 entities that implement this signature profile are assumed to support the following H.323 features:

- Fast connect;
- GK-routed model.

E.4 Security services

This annex uses the following terms for provisioning the security services.

- **Authentication-only**: This security service of the signature security profile supports user authentication where the user authenticates when correctly digitally signing some piece of data by the private key. Note that this security service does not provide countermeasures against arbitrary cut and paste, message manipulation or tampering attacks. Authentication-only may be useful for security proxies that verify authenticity of the message (data origin authentication) when forwarding[9] the message to another destination (e.g., Gatekeeper). Nevertheless, authentication-only can be applied on a hop-by-hop basis as well. Procedure III specifies this security service for an end-to-end scenario while Procedure II specifies this security service for the hop-by-hop case.
- **Authentication and integrity**: This is a combined security service that supports message integrity in conjunction with user authentication. The user authenticates when correctly digitally signing some piece of data by the private key. In addition to that, the message is protected against tampering. Both security services are provided by the same security mechanism. Combined authentication and integrity is possible only on a hop-to-hop basis. Procedure II specifies this security service.

NOTE – When digital signatures are applied, a non-repudiation security service may be supported; this depends also on the settings of the key usage bits of the signing key in the certificate (see also RFC 3280).

Asymmetric techniques using digital signatures may apply on a hop-by-hop and/or also on an end-to-end basis.

We describe the following procedures for use in this profile:

Procedure II is based on digital signatures using a private/public key pair for providing authentication, integrity and non-repudiation of RAS, Q.931 and H.245 messages. Terminals may use this method if non-repudiation and sophisticated integrity is required.

Depending on the security policy, authentication may be unilateral or mutual applying the authentication/integrity in the reverse direction as well and providing higher security thereby. The security policy of a terminal may allow "authentication-only" without computing cryptographic integrity (see clause E.7).

Gatekeepers detecting failed authentication and/or failed integrity validation in a RAS/call signalling message received from a terminal/peer gatekeeper respond with a corresponding reject

---

9 The forwarding usually changes certain parts of the message; thus end-to-end integrity cannot be realized.

message indicating security failure by setting the reject reason to **securityDenial** or other appropriate security error code according to B.2.2. Depending on the ability to recognize an attack, and the most appropriate way to react to it, a gatekeeper receiving a secured **xRQ** with undefined object identifiers (**tokenOID**, **algorithmOID**) should respond with an unsecured **xRJ**, or may discard that message. The encountered security event should be logged. On the other hand, the endpoint shall discard the received unsecured message, time out and may retry once again by considering to choose different OIDs. Likewise, a gatekeeper receiving a secured H.225.0 SETUP message with undefined object identifiers (**tokenOID**, **algorithmOID**) should respond with an unsecured RELEASE COMPLETE and reason set to **securityDenied** or may discard that message. Similarly, the encountered security event should be logged.

There is implicit H.235 signalling for indicating use of Procedure II and the applied security mechanism based upon the value of the object identifiers (see also clause E.18) and the message fields filled in. Object identifiers are referenced symbolically through letters (e.g., "A") in this text.

This profile does not use the H.235 ICV fields; rather cryptographic integrity check values are put into the **signature** field of the **token** in the **cryptoSignedToken**.

E.5 Digital signatures with public/private key pairs details (Procedure II)

The following procedures shall be adhered to if Procedure II is employed for hop-by-hop security:

- SHA1 or MD5 along with the RSA algorithm should be used to generate the digital signature. Adherence to PKCS #1 and PKCS #7 facilitates interoperability in this regard.

  The **CryptoH323Token** field in each RAS/H.225.0 message shall contain the following fields:

  – **nestedCryptoToken** containing a **CryptoToken** which itself contains the **cryptoSignedToken** containing the following fields:

    - **tokenOID** set to:
      – "A", indicating that the authentication/ integrity computation includes all fields in the H.225.0 RAS or call signalling message (see clause E.9);
      – "B", indicating that the authentication/ integrity computation includes only a subset of fields (see clause E.8) in the RAS/H.225.0 message for authentication-only.
    - **token** containing the fields:
      – **toBeSigned** containing the **EncodedGeneralToken** which actually is a **ClearToken** with the following fields set:
        - **tokenOID** set to "S", indicating that **ClearToken** is being used for message authentication/integrity/non-repudiation;
        - **timeStamp** contains the timestamp;
        - **random** contains a monotonically increasing sequence number;
        - **generalID** contains the identifier of the recipient (only in case of unicast messages);
        - **sendersID** contains the identifier of the sender;
        - **dhkey**, used to pass the Diffie-Hellman parameters as specified in this Recommendation during **Setup**-to-**Connect**:
          – **halfkey** contains the random public key of one party;
          – **modsize** contains the DH-prime (see Table D.4);
          – **generator** contains the DH-group (see Table D.4).

NOTE 1 – When the signature security profile is used without the voice encryption security profile then no Diffie-Hellman parameters should be sent and **dhkey** should be absent; **halfkey**, **modsize** and **generator** may be set to {'0'B,'0'B,'0'B}.

- **certificate** contains the digital certificate of the sender where type indicates the certificate type ("V" for MD5-RSA certificates or "W" for SHA1-RSA certificates) and **certificate** carries the actual certificate (see clause E.12).
- **algorithmOID** set to:
  - "V" indicating the use of MD5-RSA signature;
  - "W" indicating the use of SHA1-RSA signature.
- **paramS** set to NULL.
- **signature** containing the signature computed using SHA1 or MD5 RSA on all the fields (if tokenOID is "A", see clause E.9) or certain critical fields (if **tokenOID** is "B", see clause E.8) of the H.225.0 RAS or call signalling message.

When **tokenOID** "A" is used for protection of tunnelled H323-UU-PDUs including all H.245 message contents, then the signature computation shall be done over the entire H.225.0 call signalling message with all fields according to the procedure described in clause E.9. In case **tokenOID** "B" is used, authentication-only of the **CryptoToken** is achieved when applying Procedure III (see clause E.8).

- An entity (which may be one or more application hops away) for whom the signature is meant, verifies the signature.

NOTE 2 – The recipient is able to detect usage of Procedure II by evaluating the **algorithmOID** within the token of the **cryptoSignedToken** (detecting presence of "V" or "W").

E.6 Multipoint conferencing procedures

MCUs shall support secured distribution of certificates upon request from terminals by the tunnelled H.245 **ConferenceRequest** and **ConferenceResponse** commands as described in 9.1. This allows terminals to request certificates from other terminals in a multipoint conference environment and thereby obtain certainty about the other participants' identity in the conference.

ConferenceRequest conveys **requestTerminalCertificate** of which the following fields are set:

- **terminalLabel**: used as addressing means of the remote terminal through the MCU;
- **certSelectionCriteria**: the sender may request certificates only of specific types;
- **sRandom**: a random challenge generated by the requesting sender.

ConferenceResponse conveys **terminalCertificateResponse** of which the following fields are set:

- **terminalLabel**: allows association of the returned certificate to the terminal.
- **CertificateResponse**: conveys the response from the MCU with fields set to:
  - **terminalLabel**: identification of the remote terminal;
  - **certificateResponse**: this is actually an octet string ASN.1 encoded from the **EncodedReturnSig** as:
    - **generalID**: identification of the destination terminal;
    - **responseRandom**: random challenge value generated by the MCU;
    - **requestRandom**: **sRandom** played back;
    - **certificate**: conveys the returned certificate where **type** indicates the certificate type as OID and **certificate** carries the digital certificate (see clause E.12).

E.7 End-to-end authentication (Procedure III)

Figure E.1 shows a scenario with proxies separating GKs and EPs where two different CryptoTokens are used for hop-by-hop as well as end-to-end authentication and/or hop-by-hop integrity. The **CryptoToken** for hop-by-hop authentication applies only to the leg between two entities and has to be recomputed on every other leg. On the other hand, the **CryptoToken** for end-to-end authentication is generated just once by the sending endpoint and is not changed in transit by intermediate nodes. Intermediate nodes may validate signatures and certificates conveyed in end-to-end **CryptoTokens** and should forward the **CryptoToken** in transit.

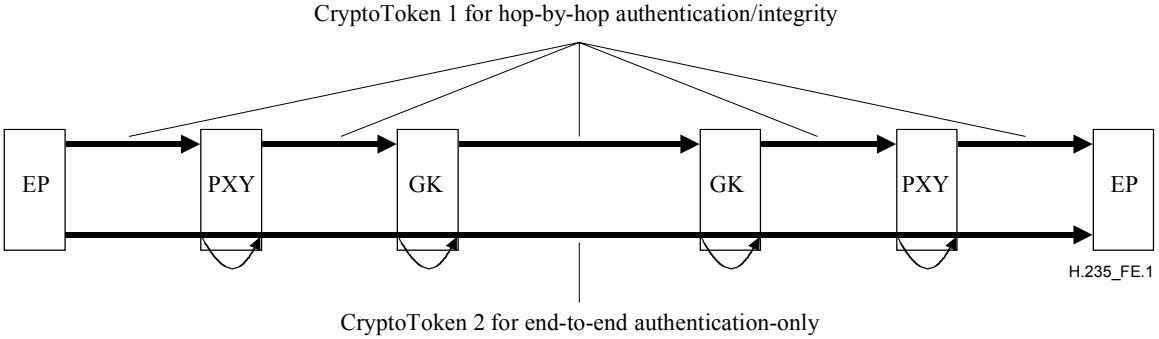


Figure E.1/H.235 – Simultaneous use of hop-by-hop security and end-to-end authentication

NOTE 1 – The proxy may be a separate network node as shown in Figure E.1 or may be collocated with the functionality of an H.323 entity, e.g., as part of the GK.

NOTE 2 – Depending on the signalled **tokenOID**, the proxy is able to determine whether the received **CryptoToken** is destined for the proxy ("S") or some other recipient ("R").

NOTE 3 – Due to the fact that intermediate entities change signalling message contents on every leg, end-to-end integrity is not possible.

For true end-to-end authentication across H.323 proxies or intermediate network elements, the sending endpoint/terminal shall compute a digital signature as follows.

The **CryptoH323Token** field in each RAS/H.225.0 message shall contain the following fields:

- **nestedCryptoToken** containing a **CryptoToken** which itself contains the **cryptoSignedToken** containing the following fields:
  - **tokenOID** set to:
    - "A", indicating that the hop-by-hop authentication/integrity computation includes all fields in the RAS/H.225.0 message (see clause E.9);
    - "B", indicating that the authentication computation includes only a subset of fields (see clause E.8) in the H.225.0 RAS or call signalling message for authentication only.
- **token** containing the fields:
  - **toBeSigned** containing the **ClearToken** field used with the following fields:
    - **tokenOID** set to "R" indicating that **ClearToken** is being used for authentication-only/non-repudiation[10] on an end-to-end basis;
    - **random** contains a monotonically increasing sequence number;

[10] Which security service is actually being applied depends also on the key usage bits in the certificate.

- **timeStamp** optionally for enhanced security only when the terminating end entities are time synchronized;
- **generalID** contains the endpoint identifier of the recipient (only in case of unicast). In case of hop-by-hop this is the identifier of the next hop; in case of end-to-end this is the far-end endpoint identifier;
- **sendersID** contains the endpoint sender;
- **certificate** contains the digital certificate of the sender where **type** indicates the certificate type ("V" for MD5-RSA certificates or "W" for SHA1-RSA certificates) and **certificate** carries the actual certificate (see clause E.12);
- **dhkey**, used to pass the Diffie-Hellman parameters as specified in this Recommendation during **Setup**-to-**Connect**:
  - **halfkey** contains the random public key of one party;
  - **modsize** contains the DH-prime (see Table D.4);
  - **generator** contains the DH-group (see Table D.4).

NOTE 4 – When the signature security profile is used without the voice encryption security profile, then no Diffie-Hellman parameters should be sent and **dhkey** should be absent; **halfkey**, **modsize** and **generator** may be set to {'0'B,'0'B,'0'B}.

- **Token** with the field:
  - **algorithmOID** set to:
    - "V", indicating the use of MD5-RSA signature;
    - "W", indicating the use of SHA1-RSA signature.
  - **paramS** set to NULL.
  - **signature** containing the signature computed using SHA1-RSA or MD5-RSA on all the fields (if **tokenOID** is "A") or certain critical fields (if **tokenOID** is "B") of the H.225.0 RAS or call signalling message.

The proxy may verify any obtained digital signature and/or certificate and may discard the message if not considered appropriate according to the local policy or the proxy shall forward the received **CryptoToken** further on. The proxy has to generate new H.235 signalling information elements for the hop-by-hop security according to Procedures II or III.

The entity terminating the leg (this could be a terminal), should verify received security information in the **CryptoToken** and depending on the presence of end-to-end security elements, may additionally evaluate the end-to-end **CryptoToken** information. The exact verification procedures in a terminal or an intermediate H.323 entity may vary according to local policy.

E.8 Authentication-only

Terminals may choose to implement authentication-only (using OID "B"). In this case, the authenticator is computed just over a subset (**ClearToken** inside **CryptoToken**) of the RAS/H.225.0 message. Authentication-only may be useful for true end-to-end authentication (see clause E.7). The following fields in the **ClearToken** structure are used as the subset:

- **tokenOID**: There is a separate token object identifier (tokenOID "B") for authentication-only implementation.
- **random**: The monotonically increasing sequence number.
- **timeStamp**: The timestamp.
- **generalID**: The identifier of the recipient (only in case of unicast messages). In case of hop-by-hop, this is the identifier of the next hop; in case of end-to-end, this is the far-end endpoint identifier.

- **sendersID**: The identifier of the sender.
- **dhkey**: The Diffie-Hellman parameters. This field and subfields are used during **Setup**-to-**Connect** messages.

The authenticator is computed over the **ClearToken** inside the **EncodedGeneralToken** (i.e., **ClearToken**) of the **token** of the **cryptoSignedToken**. The digital signature shall be computed over the ASN.1-encoded bitstring of **ClearToken**. Before computing the digital signature, the **tokenOID** in the **ClearToken** shall be set to {0 0}.

E.9 Authentication and integrity

For authentication and message integrity over all the ASN.1-coded message fields (using OID "A"), the procedure is the following.

The sender of a message shall compute the signature as follows:

1) Set the signature value to a specific default pattern with a fixed length (e.g., 1024 bits). This step shall reserve space for the maximum length of a digital signature, which is possible due to a given certificate. The exact bit pattern here does not matter, but a good choice is a unique bit pattern that does not occur in the remaining message.
2) ASN.1 encodes the entire message; for RAS this shall include the entire H.225.0 RAS message; for call signalling this shall include the entire H.225.0 call signalling message.
3) Locate[11] the default pattern in the encoded message; overwrite the found bit pattern all with zero bits.
4) Compute the digital signature upon the ASN.1-encoded message using the method indicated by the **algorithmOID** "V" or "W" (see clause E.10).
5) Substitute the default pattern in the encoded message with the computed digital signature value. In case the digital signature is shorter than the reserved space, leading zeros shall be put in front of the most significant bits of the signature value.

The recipient receives the message and then proceeds as follows:

1) ASN.1 decodes the message.
2) Extract the received digital signature value and keep it in a local variable SV.
3) Search and locate the signature value SV in the received encoded message.

NOTE – In rare circumstances where the signature value substring might occur several times in the entire message, steps 3-6 have to be iterated successively with a different starting search position.

4) Overwrite the bit pattern in the encoded message all with zeros.
5) Compute the digital signature upon the encoded message using the method indicated by the **algorithmOID** "V" or "W" (see clause E.10).
6) Compare SV with the computed signature value. The message is considered uncorrupted and authentic only if both signature values are equal; in this case, the authentication is successful and the procedure stops.
7) Otherwise, repeat steps 3-7 by restoring SV to the previous location and search for another match. If none of the matches yield a correct signature value comparison, then the authentication has failed and the message has been altered (accidentally or intentionally) during transit, or for some other reason.

---

[11] This may involve some trial-and-error steps in the rare case when the default pattern occurs more than once in the message.

E.10 Computation of the digital signature

The input to the digital signature generation process is an ASN.1-encoded bit string and includes the result of the message digest calculation process and the signer's private key. The details of the digital signature generation depend on the signature algorithm employed; the certificate determines the signature algorithm to be applied; when the key usage extension in the certificate is present, the **digitalSignature** bit must be set for the key to be eligible for signing. The signature value generated by the signer is encoded as a bit string and carried in the **signature** field.

The method described in [PKCS #1, section E.8.1.1] for computing an RSA-based digital signature with appendix (RSASSA-PKCS1-v1_5-SIGN) along with the procedures OS2IP, RSASP1, I2OSP and the EMSA-PKCS1-v1_5 encoding method shall be used.

E.11 Verification of the digital signature

The input to the signature verification process includes the result of the message digest calculation process and the signer's public key. The recipient may obtain the correct public key for the signer by any means, but the preferred method is from a certificate obtained from the **certificate** field and then validated using the hash of the signer's certificate. The validation of the signer's public key may be based on the certification path processing (RFC 3280). The details of the signature verification depend on the signature algorithm employed.

The method described in [PKCS #1, section E.8.1.2] for verifying an RSA-based digital signature with appendix (RSASSA-PKCS1-v1_5-VERIFY) along with the procedures OS2IP, RSAVP1, I2OSP and the EMSA-PKCS1-v1_5-ENCODE method shall be used.

E.12 Handling of certificates

For verification of digital signatures, the receiving entity must have access to the sender's certificate that is signed by a recognized certification authority (CA). There are several possibilities as to how the recipient can access the sender's certificate:

- The certificate is included in the message exchange as described by Procedures II and III; in this case, **certificate** holds the actual certificate and **type** holds OID "V" or OID "W".
- The recipient knows the certificate, possibly stored locally from an earlier exchange.
- Instead of including the certificate itself, the sender provides a URL where the certificate can be found. For this, **certificate** contains the URL and **type** is set to OID "P".
- The recipient obtains the certificate through some other means outside the scope of this Recommendation (e.g., LDAP directory lookup).

Whenever a digital certificate is conveyed in a message, the receiving entity (gatekeeper, endpoint) shall check the identity of the sender (gatekeeper, endpoint) against the identity of the certificate in order to prevent man-in-the-middle attacks.

For digitally-signed messages sent from the gatekeeper to the endpoint, different possibilities exist for an endpoint to check the gatekeeper identity:

– If the hostname is available, for example, in the common name attribute of the subject field or of the subjectAltName field in the certificate, the endpoint may check this hostname against the gatekeeper identifier. Additionally, the endpoint may use DNS to query the associated IP address and check it against the gatekeeper's IP address as presented in the gatekeeper's signed response message.

– For example, the gatekeeper identifier may be constructed by the IP address (represented as a 4-byte value in network byte order) concatenated with other identifying information of the gatekeeper identifier, truncated to the maximum length of senders ID field, which carries the gatekeeper’s identity. The endpoint may additionally check the IP address belonging to

the hostname against the IP address presented in the IP header of the response of the gatekeeper.

NOTE – This method would not work as expected when Network address translation (NAT) devices are involved.

– If the hostname is not available in the certificate, the IP address, which would be part of the certificate (*iPAddress subjectAltName*), shall be taken directly to perform the checks stated above.

Users should carefully examine the certificate presented by the gatekeeper to determine if it meets their expectations. If the endpoint has external information as to the expected identity of the gatekeeper, the hostname check may be omitted. For instance, an endpoint may be connecting to a gatekeeper whose address and hostname are dynamic but the endpoint knows the certificate that the gatekeeper will present. In such cases, it is important to narrow the scope of acceptable certificates as much as possible in order to prevent man-in-the-middle attacks. In special cases, it may be appropriate for the endpoint to simply ignore the gatekeeper's identity, but it must be understood that this leaves the connection open to active attacks.

If the hostname does not match the identity in the certificate, user oriented endpoints shall either notify the user (endpoints may give the user the opportunity to continue with the connection in any case) or terminate the connection with a bad certificate error. Automated endpoints shall log the error to an appropriate audit log (if available) and should terminate the connection (with a bad certificate error).

Automated endpoints may provide a configuration setting that disables this check, but shall provide a setting, which enables it.

Likewise, it is recommended that the gatekeeper perform an identity check for any digitally signed messages sent from the endpoint to the gatekeeper. How exactly the gatekeeper would implement such a checking is considered as a local matter and should be subject to implementation of the gatekeeper's security policy. As an example, one may imagine that the user name conveyed within the certificate may also be part of the H.323 identifier. Further on, the gatekeeper may crosscheck such identity information against locally administered/configured user data if available and may base a policy decision upon that.

If the gatekeeper has external information as to the expected identity of the endpoint, the hostname check may be omitted. For instance, a gatekeeper may be connecting to an endpoint whose address and hostname are dynamic, but the gatekeeper knows the certificate that the endpoint will present. In such cases, it is important to narrow the scope of acceptable certificates as much as possible in order to prevent man-in-the-middle attacks. In special cases, it may be appropriate for the gatekeeper to simply ignore the endpoint identity, but it must be understood that this leaves the connection open to active attacks.

If the hostname does not match the identity in the certificate, the gatekeeper shall log the error to an appropriate audit log (if available) and should terminate the connection (with a bad certificate error).

If a subjectAltName extension of type dNSName is present, that shall be used as the identity. Otherwise, the (most specific) Common Name field in the Subject field of the certificate shall be used. Although the use of the Common Name is existing practice, it is deprecated and Certification Authorities are encouraged to use the dNSName instead.

Matching shall be performed using the matching rules specified by RFC 3280. If more than one identity of a given type is present in the certificate (e.g., more than one dNSName name), a match in any one of the set is considered acceptable. Names may contain the wildcard character * which is considered to match any single domain name component or component fragment. For example, *.a.com matches foo.a.com but not bar.foo.a.com. f*.com matches foo.com but not bar.com.

Procedures II and III provide means to carry a digital certificate. For efficiency, the digital certificates of the entities need to be transmitted at most only once if they are not already available in the entities through other means outside of this Recommendation. The certificate exchange thus should occur only at the beginning of a communication establishment: for RAS, this occurs either during gatekeeper discovery or, if this phase is omitted, then during gatekeeper registration. Similarly, for fast connect, where the certificate may be included in the initial call signalling messages but can safely be omitted in later call signalling messages.

For this security profile, X.509v3 (1997) certificate shall be used. Other certificate formats are for further study.

E.13 Usage illustration for Procedure II

Consider the case in Figure E.2 where each entity has its own private-public key pair/certificate. An entity may also possess multiple key pairs. In the figure, an H.323 proxy separates EP1 from GK1.

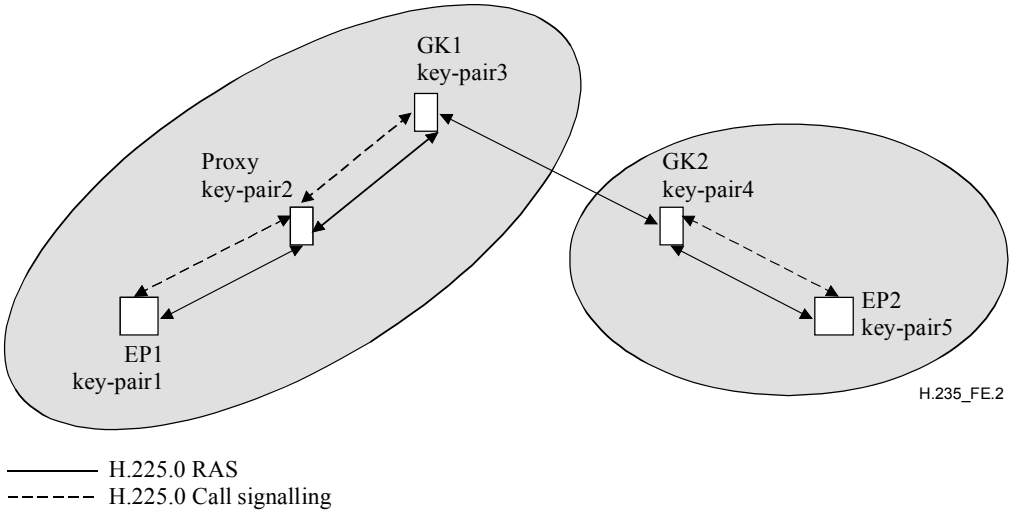


Figure E.2/H.235 – Illustrating public-key usage in a GK-GK routed model

The H.323 proxy acts in a dual behaviour. On the one hand, the proxy terminates the authentication and integrity on each of its legs. The proxy actively includes the freshly computed authentication/integrity information in the outgoing RAS messages in a similar manner as described in Procedure I of Annex D. On the other hand, the proxy lets the end-to-end security information pass unmodified. The proxy may, however, verify received certificates and/or digital signatures in transit.

Below, we illustrate the procedure details for RAS, H.225.0 call signalling and H.245 message authentication, integrity and non-repudiation.

E.13.1 RAS message authentication, integrity and non-repudiation

Consider the case for a hop-to-hop communication where EP1 wishes to send a RAS message, say an **ARQ** message, to GK1. EP1 generates a timestamp and a sequence number and includes it in the **timeStamp** and **random** fields respectively, along with the proxy's alias in the **generalID** field and the **sendersID** of EP1. These fields are present in the **ClearToken** field of the **EncodedGeneralTokens** present in the **token** of the **cryptoSignedToken** of the **CryptoToken** field of the **cryptoH323Token** of the **ARQ** message. This **cryptoH323Token** is one of at least several tokens in the **cryptoTokens** sequence. The **tokenOID** within the **cryptoSignedToken** is set to "A", indicating that all the fields in the **ARQ** message are signed. The **token** in **cryptoSignedToken** has **algorithmOID** set to "V", indicating the use of MD5-RSA or

algorithmOID set to "W", indicating the use of SHA1-RSA and **paramS** set to NULL. EP1 then computes the signature based on the given signature algorithm using its private key. The signature is computed over all the fields of the **ARQ** message when **tokenOID** is set to "A". EP1 includes the computed signature within **signature** in the **token** field of the **cryptoSignedToken** field of the **CryptoToken** present in the **cryptoH323Token** of the **ARQ** message and includes its certificate in the **certificate** field.

Similarly, for the end-to-end communication through a proxy, EP1 generates another **CryptoToken** containing a digital signature that covers certain critical fields (see clause E.7) in the **ClearToken** of the **ARQ** message. The **tokenOID** in the **CryptoSignedToken** is set to "B", indicating authentication-only of that **ClearToken**; sets **tokenOID** in the **ClearToken** to "R", indicating end-to-end authentication, also **timeStamp**, **random**, **sendersID**, **generalID** and in case it is a **SETUP/CONNECT** also **dhkey**, sets in **token** the following fields: **algorithmOID** to "V" or "W", indicating the signature algorithm, **paramS** to NULL, and **signature** to the computed digital signature over the **ClearToken** fields. The **certificate** carries the digital certificate of EP1. The **ARQ** message is then sent to the proxy.

Upon receiving the **ARQ** message, the proxy verifies the signature of those tokens that are addressed to it (in this case, say, that with **tokenOID** "A"). This is based on several criteria that include:

- liveness of the timestamp, uniqueness of the **random**;
- identity of the **generalID** and own identifier;
- access permissions for the **sendersID**;
- matching of signature in **ARQ** message with that computed by GK1;
- verification of Diffie-Hellman parameters, e.g., testing whether the 1024-bit prime and generator are correct. Testing of whether the DH-parameters are secure is a time-consuming process and may be done only when local policy requires it;
- verification of the received certificate.

If the signature is successfully verified, the proxy computes a new signature to insert (replace) in the **ARQ** message before forwarding it to GK1 as follows. The proxy replaces the **timeStamp**, **random**, **sendersID** and **generalID** fields in the **ClearToken** (**toBeSigned**) field using values relevant to the proxy-GK1 leg. The **timestamp** field contains the current timestamp, the **random** field contains the next monotonically increasing sequence number for the proxy-GK1 leg, the **sendersID** of the proxy and the **generalID** field contains the alias of GK1. The proxy then computes a new signature for this **ARQ** message using its private key and signature algorithm, inserts it in **signature** within **token** and adds its **certificate**. The proxy also includes the received end-to-end **CryptoToken** with its **ClearToken** in the new outgoing message and passes the **ARQ** message on to GK1. The signature, computed by EP1 based on selected fields of the **ARQ** message (**tokenOID** of "B") and which was not meant for the proxy, is also passed untouched in the **ARQ** message to GK1.

Upon receiving the **ARQ** message, GK1 verifies the signatures, computes a new signature after modifying the **ClearToken** fields in **toBeSigned** suitably, inserts it in the **signature** field, adds its **certificate** and passes the **Setup** message on to EP2. Again, GK1 should forward any end-to-end information received in the separate **CryptoTokens** to the peer GK2 by including that information into a separate **CryptoToken** unmodified.

E.13.2 RAS authentication only

Consider the case for a hop-to-hop communication where EP1 wishes to send a RAS message, say an **ARQ** message, to GK1. EP1 generates a timestamp and a sequence number and includes it in the **timeStamp** and **random** fields respectively, along with the proxy's alias in the **generalID** field and the EP's id in the **sendersID**. These fields are present in the **ClearToken** field of **toBeSigned**

present in the **token** in **cryptoSignedToken** of the **CryptoToken** field of the **cryptoH323Token** of the **ARQ** message. The **tokenOID** within the **cryptoSignedToken** is set to "B" indicating that only the specified subset fields in the **ClearToken** are signed. The **token** in **cryptoSignedToken** has **algorithmOID** set to "V" indicating use of MD5-RSA or "W" indicating use of the SHA1-RSA signature algorithm and **paramS** set to NULL. EP1 then computes the signature based on the signature algorithm using its private key. The signature is computed over the specified **ClearToken** fields of the **ARQ** message. EP1 includes the computed signature within **signature** in the **token** field of the **cryptoSignedToken** field of the **CryptoToken** present in the **cryptoH323Token** of the **ARQ** message and adds its **certificate**.

Similarly, EP1 generates another digital signature for end-to-end authentication that covers certain **ClearToken** fields in a separate **CryptoToken** in the **ARQ** message. This digital signature (identified by **tokenOID** of "V" or "W") is included. The **ARQ** message is then sent to the proxy.

Upon receiving the **ARQ** message, the proxy verifies the signature of those tokens that are addressed to it (in this case, say, that with **tokenOID** "B"). This is based on several criteria that include:

- liveness of the timestamp, uniqueness of the **random**;
- identity of the **generalID** and own identifier;
- access permissions for the **sendersID**;
- matching of signature in **ARQ** message with that computed by GK1;
- verification of the received certificate.

If the signature is successfully verified, the proxy computes a new signature to insert (replace) in the **ARQ** message before forwarding it to GK1 as follows. The proxy replaces the **timeStamp**, **random**, **sendersID** and **generalID** fields in the **ClearToken** field of **toBeSigned** using values relevant to the proxy-GK1 leg. The **timestamp** field contains the current timestamp, the **random** field contains the next monotonically increasing sequence number for the proxy-GK1 leg, and the **generalID** field contains the alias of GK1. The proxy then computes a new signature for this **ClearToken** using its private key and signature algorithm MD5-RSA or SHA1-RSA (**algorithmOID** ="V" or "W"), inserts it in **signature** within **token** of **cryptoSignedToken**, adds its **certificate** and passes the **ARQ** message on to GK1. The signature computed by EP1 based on selected **ClearToken** fields of the **ARQ** message (**tokenOID** of "B") and which was not meant for the proxy is also passed untouched in the **ARQ** message to GK1.

Upon receiving the **ARQ** message, GK1 verifies the signature, computes a new signature after modifying the **ClearToken** fields in **toBeSigned** suitably, inserts it in the **signature** field and passes the **Setup** message on to EP2. The end-to-end signature information from EP1 is included untouched in the **Setup** message.

E.13.3 H.225.0 message authentication, integrity and non-repudiation

The procedure for H.225.0 messages is identical to that for RAS messages. The only difference is that the set of fields that need to be signed has to be identified for each H.225.0 call signalling message when the **tokenOID** is set to "B".

E.13.4 H.245 message authentication and integrity

Consider the case where EP1 wishes to send an H.245 message, say a **TerminalCapabilitySet** message, to EP2. EP1 checks to see if an H.225.0 message needs to be sent to the proxy. If so, then the H.245 message is tunnelled within that H.225.0 message. The fields within the H.225.0 message are set as described earlier for the transmission of a H.225.0 message. Since the H.245 message is tunnelled, the **h323-uu-pdu** in the **h323-UserInformation** message has its fields set as follows:

- **h323-message-body** field is set to the H.225.0 message type that is being transmitted.

- **h245Tunnelling** set to TRUE.
- **h245Control** contains the H.245 PDU octet string.

However, if no H.225.0 message transmission is pending, then the H.245 message is tunnelled within an ad hoc H.225.0 **facility** message. The **h323-uu-pdu** in the **h323-UserInformation** message has its fields set as follows:

- **h323-message-body** field is set to **facility** which contains:
  - **reason** set to **undefinedReason**;
  - **tokens** and **cryptoTokens** set as for any H.225.0 message.
- **h245Tunnelling** set to TRUE.
- **h245Control** contains the H.245 PDU octet string.

The **facility** message is then transmitted by EP1 to the proxy.

In either case (whether a H.225.0 message transmission is pending or an ad hoc H.225.0 **facility** message is used), the proxy verifies the signature which is meant for it (in this case, depicted by **tokenOID** of "A") upon receiving the message. Then, if a H.225.0 message transmission is pending for the proxy-GK1 leg, the H.245 message is tunnelled within that message; otherwise, it is tunnelled within an ad hoc H.225.0 **facility** message. As in the case of transmission of any H.225.0 call signalling message, a new signature is computed for the H.225.0 message prior to its transmission from the proxy to GK1. The signature that was sent from EP1 to the proxy and that was not meant for the proxy is passed untouched by the proxy onto GK1.

This clause provides a summary of how, and by which means, the signature profile secures the various H.323 signalling messages.

E.14 H.235 version 1 compatibility

While these security profiles are developed with H.235 version 2 [H.235v2] in mind, it is also possible to apply the security profiles for H.235 version 1 [H.235v1] with some minor modifications. A recipient is able to detect presence of the sender's H.235 protocol version by evaluating the security profile object identifiers (see clause E.18).

H.235 version 1 [H.235v1] implementations:

- do not set or evaluate the **sendersID** in the **ClearToken**.

E.15 Multicast behaviour

H.225.0 multicast messages such as **GRQ** or **LRQ** shall include a **CryptoToken** according to Procedures II and III where the **generalID** is not set. When such messages are sent unicast, then the message shall include a **CryptoToken**.

E.16 List of secure signalling messages

E.16.1 H.225.0 RAS

H.225.0 RAS message	H.235 signalling fields	Authentication-only	Authentication and integrity	Non-repudiation
Any	cryptoTokens	Procedure II/III	Procedure II/III	Procedure II/III

NOTE – For unicast messages, Procedures II or III shall be applied with the security fields in the **CryptoToken** used.

E.16.2 H.225.0 call signalling

H.225.0 call signalling message	H.235 signalling fields	Authentication-only	Authentication and integrity	Non-repudiation
Alerting-UUIE, CallProceeding-UUIE, Connect-UUIE, Setup-UUIE, Facility-UUIE, Progress-UUIE, Information-UUIE, ReleaseComplete-UUIE, Status-UUIE, StatusInquiry-UUIE, SetupAcknowledge-UUIE, Notify-UUIE	cryptoTokens	Procedure II/III	Procedure II/III	Procedure II/III

E.17 Usage of sendersID and generalID

The **ClearToken** holds **sendersID** and **generalID** fields. When identification information is available, the **sendersID** shall be set to the gatekeeper identifier (GKID) for the gatekeeper-initiated message and to the endpoint identifier (EPID) for the endpoint-initiated messages. When identification information is available, the **generalID** shall be set to the GKID for endpoint-initiated messages and to EPID for the gatekeeper-initiated messages. When the identification information is not available, or in case of broadcast/multicast is ambiguous, the field is missing or shall contain a null string. Table E.2 summarizes the situation:

Table E.2/H.235 – Object identifiers used by Annex E

Message	sendersID	generalID
Unicast **GRQ**	**EPID** if available, otherwise **NULL**	**GKID**
Multicast **GRQ**	**EPID** if available, otherwise **NULL**	
GCF, GRJ	**GKID**	**EPID** if available, otherwise **NULL**
Initial **RRQ**		**GKID**
RCF	**GKID**	**EPID**
RRJ	**GKID**	
URQ, UCF, URJ, BRQ, BCF, BRJ, DRQ, DCF, DRJ, NSM, RIP, SCI, SCR, XRS (EP-to-GK)	**EPID**	**GKID**
URQ, UCF, URJ, BRQ, BCF, BRJ, DRQ, DCF, DRJ, NSM, RIP, SCI, SCR, XRS (GK-to-EP)	**GKID**	**EPID**
ARQ, IRQ, RAI	**EPID**	**GKID**
ACF, ARJ, BCF, LCF, LRJ, IRR, IRQ, RAC, LCF, LRJ, IACK, INAK	**GKID**	**EPID**
Unicast **LRQ** (EP-to-GK)	**EPID**	**GKID**
Unicast **LRQ** (GK-to-GK)	**GKID**	**GKID**
Multicast **LRQ**	**EPID**	
NOTE – GKID stands for gatekeeper identifier, EPID stands for endpoint identifier. Blank indicates a missing or null identification string.		

E.18 List of object identifiers

Table E.3 lists all the referenced OIDs (see also [OIW] and [WEBOIDs]). There are object identifiers for H.235v1 [H.235v1] and for H.235v2 [H.235v2].

Table E.3/H.235 – Object identifiers used by Annex E

Object identifier reference	Object identifier value(s)	Description
"A"	{itu-t (0) recommendation (0) h (8) 235 version (0) 2 1} {itu-t (0) recommendation (0) h (8) 235 version (0) 1 1}	Used in Procedure II for the CryptoToken-tokenOID indicating that the signature includes **all** fields in the H.225.0 RAS or call signalling message (authentication and integrity).
"B"	{itu-t (0) recommendation (0) h (8) 235 version (0) 3 2} {itu-t (0) recommendation (0) h (8) 235 version (0) 2 2} {itu-t (0) recommendation (0) h (8) 235 version (0) 1 2}	Used in Procedure II for the CryptoToken-tokenOID indicating that the signature includes a **subset** of fields in the RAS/H.225.0 message (ClearToken) for authentication-only terminals without integrity. Used in Annex D Procedure IA for the CryptoToken-tokenOID indicating that the hash includes a subset of fields in the RAS/H.225.0 message (ClearToken) for authentication-only terminals without integrity
"P"	{itu-t (0) recommendation (0) h (8) 235 version (0) 2 4} {itu-t (0) recommendation (0) h (8) 235 version (0) 1 4}	Used in Procedures II or III to indicate that **certificate** carries a URL.
"R"	{itu-t (0) recommendation (0) h (8) 235 version (0) 2 3} {itu-t (0) recommendation (0) h (8) 235 version (0) 1 3}	Used in Procedure II for the ClearToken-tokenOID indicating that the ClearToken is being used for end-to-end authentication/ integrity.
"S"	{itu-t (0) recommendation (0) h (8) 235 version (0) 2 7} {itu-t (0) recommendation (0) h (8) 235 version (0) 1 7}	Used in Procedure II this token OID indicates message authentication, integrity and non-repudiation.
"V"	{iso(1) member-body(2) us(840) rsadsi(113549) pkcs(1) pkcs-1(1) 4}	Used in Procedure II or in Procedure III as algorithm OID indicating use of MD5-RSA digital signature.
"W"	{iso(1) member-body(2) us(840) rsadsi(113549) pkcs(1) pkcs-1(1) 5}	Used in Procedure II or in Procedure III as algorithm OID indicating use of SHA1-RSA digital signature.

Annex F

Hybrid security profile

Summary

The purpose of this annex is to describe an efficient and scalable, PKI-based hybrid security profile for version 2 of ITU-T Rec. H.235. The hybrid security profile contained herein takes advantage of the security profiles in Annex D and Annex E by deploying digital signatures from Annex E and deploying the baseline security profile from Annex D.

F.1 Overview

This annex describes an efficient and scalable, PKI-based hybrid security profile deploying digital signatures from Annex E and deploying the baseline security profile from Annex D. This annex is suggested as an option. H.323 security entities (terminals, gatekeepers, gateways, MCUs, etc.) may implement this hybrid security profile for improved security or whenever required.

The notion of "hybrid" in this text shall mean that security procedures from the signature profile in Annex E are actually applied in a lightweight sense and the digital signatures still conform to the RSA procedures. However, digital signatures are deployed only where absolutely necessary while highly efficient symmetric security techniques from the baseline security profile in Annex D are used otherwise.

The hybrid security profile is applicable for scaleable "global" IP telephony. This security profile overcomes the limitations of the simple, baseline security profile of Annex D when strictly applying it. Furthermore, this security profile overcomes certain drawbacks of Annex E, such as the need for higher bandwidth and increased performance needs for processing, when strictly applying it. For example, the hybrid security profile does not depend on the (static) administration of mutual shared secrets of the hops in different domains. Thus, users can more easily choose their VoIP provider. This security profile thus supports a certain kind of user mobility as well. It applies asymmetric cryptography with signatures and certificates only where necessary and otherwise uses simpler and more efficient symmetric techniques. It provides tunnelling of H.245 messages for H.245 message integrity and also implements some provisions for non-repudiation of messages.

The hybrid security profile mandates the GK-routed model and is based upon the H.245 tunnelling techniques. Support for non GK-routed models is for further study.

The features provided by this profile include:

For RAS, H.225.0 and H.245 messages:

- User authentication to a desired entity irrespective of the number of application level hops[12] that the message traverses.
- Integrity of all or critical portions (fields) of messages arriving at an entity irrespective of the number of application-level hops that the message traverses. Integrity of the message itself using a strongly generated random number is also optional.

[12] Hop is understood here in the sense of a trusted H.235 network element (e.g., GK, GW, MCU, proxy, or firewall). Thus, application level hop-by-hop security when used with symmetric techniques does not provide true end-to-end security between terminals.

- Application-level hop-by-hop message authentication, integrity and (some) non-repudiation provide these security services for the entire message.
- Using the available public-key infrastructure, users can choose their service provider. Key-management for session key distribution is well integrated in the hybrid security profile.

Suitable provision of the above-described security services thwarts several types of attacks, including:

- *Man-in-the-middle attacks*: Application-level hop-by-hop message authentication and integrity prevents against such attacks when the man-in-the-middle is in an application-level hop, say, a hostile router.
- *Replay attacks*: Use of timestamps and sequence numbers prevent such attacks.
- *Spoofing*: User authentication prevents such attacks.
- *Connection hijacking*: Use of authentication/integrity for each signalling message prevents such attacks.

F.2 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; 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. The reference to a document within this Recommendation does not give it, as a stand-alone document, the status of a Recommendation.

– ITU-T Recommendation H.225.0, version 4 (2000), *Call signalling protocols and media stream packetization for packet-based multimedia communication systems*.

– ITU-T Recommendation H.235, version 2 (2000), *Security and encryption for H-series (H.323 and other H.245-Based) multimedia terminals*.

– ITU-T Recommendation H.245, version 8 (2001), *Control protocol for multimedia communication*.

– ITU-T Recommendation H.323, version 4 (2000), *Packet-based multimedia communications systems*.

– IETF RFC 3280 (2002), *Internet X.509 Public Key Infrastructure Certificate and Revocation List (CRL) Profile*.

F.3 Acronyms

This annex defines the following acronyms:

GCF Gatekeeper Confirm

GK Gatekeeper

GRQ Gatekeeper Request

ICV Integrity Check Value

LRQ Location Request

OID Object Identifier

RAS Registration, Admission and Status

RCF Registration Confirm

RRQ Registration Request

RSA Rivest, Shamir and Adleman encryption algorithm

SHA Secure Hash Algorithm

URQ Unregistration Request

F.4 Specification conventions

The hybrid security profile uses terms and definitions from Annexes D and E.

While the message integrity service always provides message authentication, the reverse is not always true. For the authentication-only mode, the integrity assured only spans a certain subset of message fields. This applies to integrity services realized by asymmetric means (e.g., digital signatures). Thus, in practice, a combined authentication and integrity service uses the same key material without introducing a security weakness.

This security profile is applicable in environments with potentially many terminals, where static password/symmetric key assignment is not feasible, e.g., in large-scale or global-scale scenarios. Instead, this security profile assumes availability of a public-key infrastructure with assigned certificates and private/public-keys, directories, etc. In addition, this security profile deploys symmetric crypto techniques where applicable.

This security profile introduces the terms "first" message and "last" message sent. Security protection of the first message (and probably also for the last message) is different from security protection of the remaining other messages.

The "first message" sent is understood as a message that flows between two H.323 entities and establishes a security context. It makes symmetric key material available to both entities and, for example, marks the beginning of a call. For H.225.0 RAS, the first message is the RRQ and the related response message. For H.225.0 call signalling using fast start, the first message is SETUP and CONNECT.

The "last message" terminates the established security context. The established key material shall be destroyed. For H.225.0 RAS, the last message is the URQ and related response message, while for H.225.0 call signalling the last message is RELEASE-COMPLETE.

This security profile assumes the GK-routed call model, where the fast connect call signalling method is applied. H.245 call control messages are securely tunnelled in H.225.0 call signalling messages and inherit thereby the H.225.0 security protection scheme.

The signature security profile allows to securely tunnel H.245 call control PDUs within H.225.0 facility messages. The H.245 key update and synchronization mechanisms require tunnelling for key-update FACILITY message to be signalled and is useful, for example, for very long duration calls.

The diagonally shaded area in Table F.1 represents the security mechanisms that are used by the hybrid security profile.

NOTE – RSA certificates with MD5 hashing are not part of this security profile.

The voice encryption security profile of Annex D (see clause D.7) could be optionally used in conjunction with the hybrid security profile. Its use is negotiated as part of the call set-up signalling.

Table F.1/H.235 – Overview of the hybrid security profile

Security services	Call functions			
	RAS	**H.225.0**	**H.245 (Note 3)**	**RTP**
Authentication	RSA digital signature (SHA1) HMAC-SHA1-96	RSA digital signature (SHA1) HMAC-SHA1-96	RSA digital signature (SHA1) HMAC-SHA1-96	
Non-repudiation	(possible only on first message)	(possible only on first message)		
Integrity	RSA digital signature (SHA1) HMAC-SHA1-96	RSA digital signature (SHA1) HMAC-SHA1-96	RSA digital signature (SHA1) HMAC-SHA1-96	
Confidentiality				
Access control				
Key management	certificate allocation authenticated Diffie-Hellman key-exchange	certificate allocation authenticated Diffie-Hellman key-exchange		

NOTE 1 – The hybrid security profile has to be also supported by other H.235 entities (e.g., gatekeepers, gateways and H.235 proxies).

NOTE 2 – Available key usage bits in the certificate could also determine the security service provided by a terminal (e.g., non-repudiation asserted).

NOTE 3 – Tunnelled H.245 or embedded H.245 inside H.225.0 fast connect.

This annex may apply message integrity protection that spans the entire message. For H.225.0 RAS the integrity protection covers the entire RAS message; for call signalling this covers the entire H.225.0 call signalling message including the Q.931 headers.

For authentication, the user should use a public/private key signature scheme. Such a scheme usually provides for better integrity.

This Recommendation does not describe procedures for registration, certification and certificate allocation from a trust centre and private/public key assignment, directory services, specific CA parameters, certificate revocation, key pair update/recovery and other certificate operational or management procedures such as certificate or public/private key and certificate delivery and installation in terminals. Such procedures may happen by means that are not part of this annex.

The communication entities involved are able to implicitly determine usage of either the Annex D baseline security profiles, Annex E signature profile, or this hybrid security profile by evaluating the signalled security object identifiers in the messages (**tokenOID**, and **algorithmOID**; see also clause E.8).

F.5 H.323 requirements

H.323 entities that implement this hybrid security profile are assumed to support the following H.323 features:

- Fast connect;
- H.245 tunnelling; and
- GK-routed model.

F.6 Authentication and integrity

This annex uses the following terms for provisioning the security services.

- **Authentication and integrity**: This is a combined security service that supports message integrity in conjunction with user authentication. The user authenticates when either correctly digitally signing some piece of data with the private key or when correctly applying a related, shared secret. In addition to that, the message is protected against tampering. Both security services are provided by the same security mechanism. Combined authentication and integrity is possible only on a hop-to-hop basis.

NOTE – When digital signatures are applied, a non-repudiation security service may be supported. This also depends on the settings of the key usage bits of the signing key in the certificate (see also RFC 3280).

We describe the following procedures for use in this profile.

Procedure IV is based on digital signatures using a private/public key pair and deploying symmetric crypto techniques for providing authentication and integrity of RAS, Q.931 and H.245 messages. Terminals may use this method if efficient, scalable security is required.

Depending on the security policy, authentication may be unilateral or mutual (i.e., applying the authentication/integrity in the reverse direction as well, thereby providing higher security). The preferred security mode is to have mutual authentication.

Gatekeepers detecting failed authentication and/or failed integrity validation in a RAS/call signalling message received from a terminal/peer gatekeeper will respond with a corresponding reject message indicating security failure. This is done by setting the reject reason to **securityDenial**, or other appropriate security error code according to clause B.2.2. Depending on the ability to recognize an attack, and the most appropriate way to react to it, a gatekeeper receiving a secured **xRQ** with undefined object identifiers (**tokenOID**, **algorithmOID**) should respond with an unsecured **xRJ** and reject with reason set to **securityDenial** or may discard that message. The endpoint shall discard the received unsecured message, time out and may retry once again by considering to choose different OIDs. Likewise, a gatekeeper receiving a secured H.225.0 call signalling SETUP message with undefined object identifiers (**tokenOID**, **algorithmOID**) should respond with an unsecured RELEASE COMPLETE and reject with reason set to **securityDenied**, or it may discard that message whereas a gatekeeper receiving a secured H.225.0 FACILITY with undefined object identifiers (**tokenOID**, **algorithmOID**) should respond with an unsecured FACILITY and reason set to **undefinedReason**, or it may discard that message. Similarly, the encountered security event should be logged. As part of the returned response, the sender may provide a list of acceptable certificates in separate tokens, in order to facilitate selection of an appropriate one by the recipient.

There is implicit H.235 signalling for indicating use of Procedure IV and the applied security mechanism based upon the value of the object identifiers (see also clause F.12) and the message fields filled-in. In this Recommendation, object identifiers are referenced symbolically through letters (e.g., "A").

This profile does not use the H.235 ICV fields. Rather, cryptographic integrity check values are put into the **signature** field of the **token** in the **cryptoSignedToken** when referring to Annex E, or the integrity check values are put in the hash fields of the **CryptoToken** when referring to Annex D.

F.7 Procedure IV

The following procedures shall be adhered to if Procedure IV is employed for hop-by-hop security. This procedure unites Procedure I of Annex D (see D.6.3.2) and Procedure II of Annex E (see clause E.5).

For the first message, including corresponding response sent in each direction, Annex E Procedure II (hop-by-hop authentication and integrity, see clause E.5) shall be used with the following settings:

- OID "A1" instead of OID "A" and OID "S1" instead of OID "S". Use of these OIDs allows identifying the hybrid security profile.
- **algorithmOID** in **tokenOID** shall be set to "W" indicating use of RSA-SHA1 signature.
- **signature** shall contain an ASN.1 encoded RSA signature (see clause E.10).
- **certificate** should contain the sender's user certificate if not available otherwise to the receiver; **type** shall hold OID "W" indicating an included RSA-SHA1 certificate or OID "P" (see clause E.18) indicating that **certificate** holds an URL.

In a single administrative domain scenario, "the first message/response" is defined to equal the intial H.225.0 RAS message/response; this is usually either GRQ/GCF or RRQ/RCF. In a multi-administrative domain scenario, the first message/response within each domain is defined as above; the first message between the domains is defined as SETUP.

Whenever a digital certificate is conveyed in a message, the receiving entity shall check the identity of the sender against the identity of the certificate according to the procedure in E.12 in order to prevent man-in-the-middle attacks.

Sender and recipient exchange and compute an authenticated Diffie-Hellman secret bit string. Table D.4 provides an example of Diffie-Hellman group parameters and recommends taking the 1024-bit prime whenever possible, for security reasons. The Diffie-Hellman secret shall be computed for each leg, regardless of whether the voice encryption profile is deployed or not.

From the common bit string that both parties compute, both parties derive a 160-bit secret by taking the least significant 160 bits. The resulting 160-bit secret acts as the password/shared secret that is used in Annex D.

In a scenario with gatekeepers in distinct administrative domains, sender and receiver shall use two tokens in each direction for H.225.0 call signalling:

- One **ClearToken** inside **CryptoToken**, which is used to compute the media key that is shared among the terminals (see D.7.1). This is only necessary if voice encryption is to be deployed.
- A separate **ClearToken** is used to compute a link key that is shared among the sender and receiver entities for protection of the signalling link. This link key replaces the shared password among the gatekeepers in Annex D. The **tokenOID** of that **ClearToken** shall be set to "Q", indicating use of Diffie-Hellman and hybrid security profile. Computation of the link key proceeds in the same manner as computation of the media key (see D.7.1).

NOTE 1 – For direct-routed environments, sender/receiver entities and terminals correspond. For GK-routed environments, the link key is shared hop-by-hop between each pair of peer gatekeepers, while the media key is shared on an end-to-end basis.

In GK-routed environments, the GK shall forward the received Diffie-Hellman token from the endpoint to the next hop.

For all but the very first message/response sent in each direction, Annex D Procedure I (see D.6.3.2) shall be used. This applies also in a scenario where multiple gatekeepers are located within an administrative domain. In this case, there is no need for asymmetric key management; instead, Annex D is sufficient.

This annex may be used with H.235 version 1 systems when taking care of restricted use of sendersID and generalID, as described in clause E.17.

It is anticipated that a gatekeeper should receive only a single **RRQ** including a DH-token with a digital signature from a particular fixed endpoint. However, lost or delayed **RCF**/**RRJ** messages may lead to retransmission using another signed **RRQ**.

In case the corresponding registration response does not arrive timely at the endpoint, the endpoint may attempt another try. For this, the endpoint shall use the most recent DH token but use a new sequence number and a new timestamp.

For a particular fixed endpoint, the gatekeeper shall use the most recently received signed **RRQ** message and derive the shared secret from that DH-token, regardless of whether or not the GK already has a shared secret available. Thus, the GK shall overwrite any existing shared secret with the newly derived secret. The GK shall respond with a signed **RCF** that holds the response DH-token. Preferably, the response DH-token should be generated anew.

NOTE 2 – The recommended and preferred method for key update is by using the FACILITY message as defined in clause F.9. However, it is recognized, that key update may be achieved using another additive signed **RRQ** with a new DH-token.

NOTE 3 – A gatekeeper in possession of a shared secret shall respond to an HMAC-protected **RRQ** (according to Annex D) with an HMAC-protected response message.

F.8 Security association for concurrent calls

An optimization is provided for the case that a fixed pair of entities would process several independent calls in parallel using a single call signalling channel. Instead of establishing several link keys with Diffie-Hellman for each call, a security association is defined which spans multiple concurrent calls.

More precisely, the security association spans all calls between a fixed pair of entities as long as the call signalling channel is alive. Entities use the **multipleCalls** flag within SETUP to indicate the capability of signalling multiple calls over a single call signalling connection (see 7.3/H.323).

If the single call signalling connection is used, then only one common link key needs to be established, see Figure F.1.

On the other hand, if the **multipleCalls** flag within SETUP is not set, then a link key shall be individually computed anew for each call.

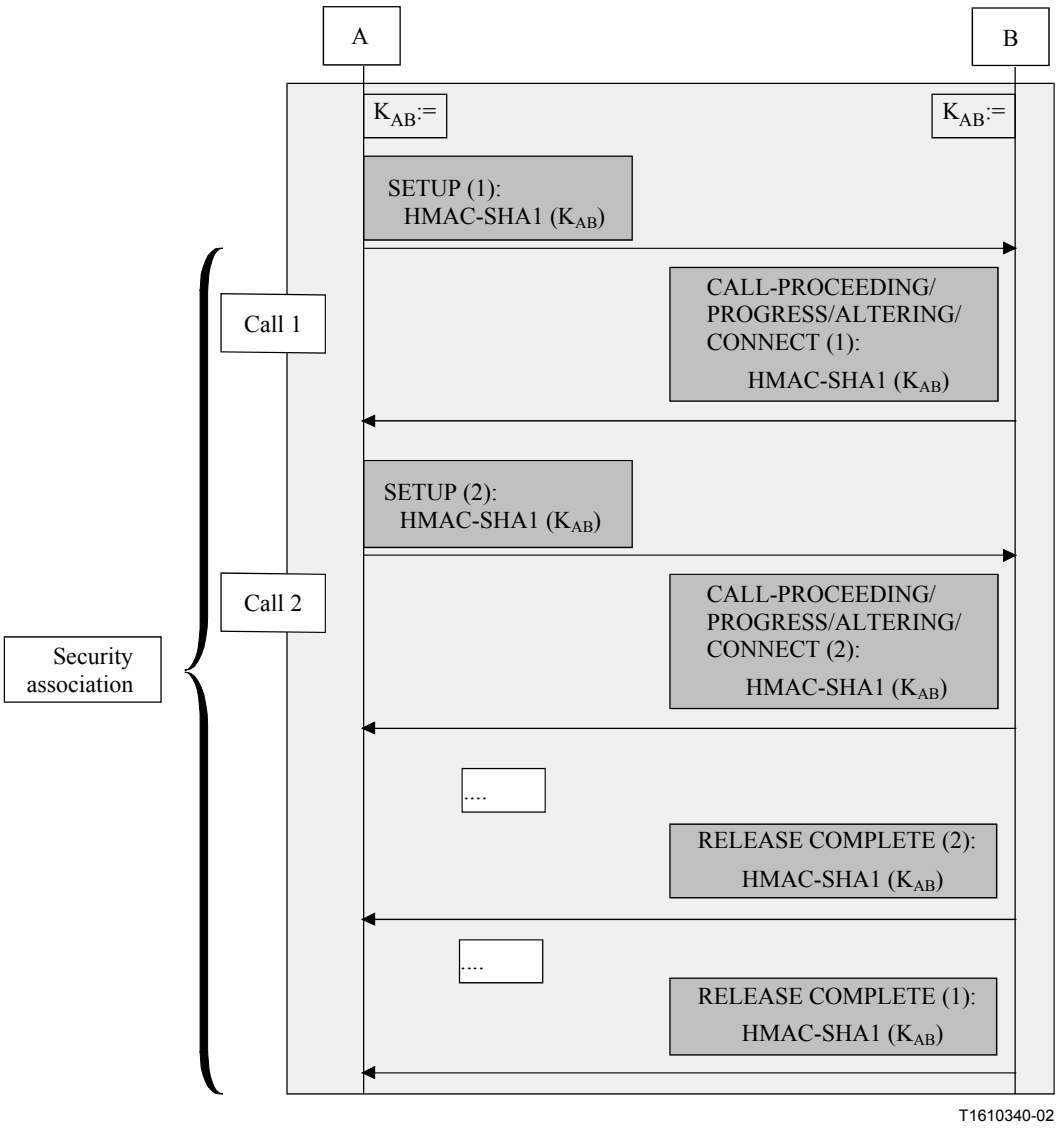


Figure F.1/H.235 – Security association for concurrent calls

F.9 Key update

An optional key update procedure allows either communication entity (GK or terminal) to refresh the currently-used session key with a new one. Such a key update should be initiated by whichever entity feels a need for it. A key update may be motivated by a compromised session key, the perception that the session key has or will become insecure, or other security policy criteria. These aspects are all outside the scope of this Recommendation.

The initiator invokes the key update using the FACILITY message. The FACILITY message for key update conveys a new Diffie-Hellman token, an optional digital certificate, and a digital signature of the initiator. Upon reception of the FACILITY message, the recipient replies with a similar FACILITY message conveying his Diffie-Hellman token, an optional digital certificate, and a digital signature of the recipient. Upon completion of the key update procedure, initiator and responder shall use the computed new link key.

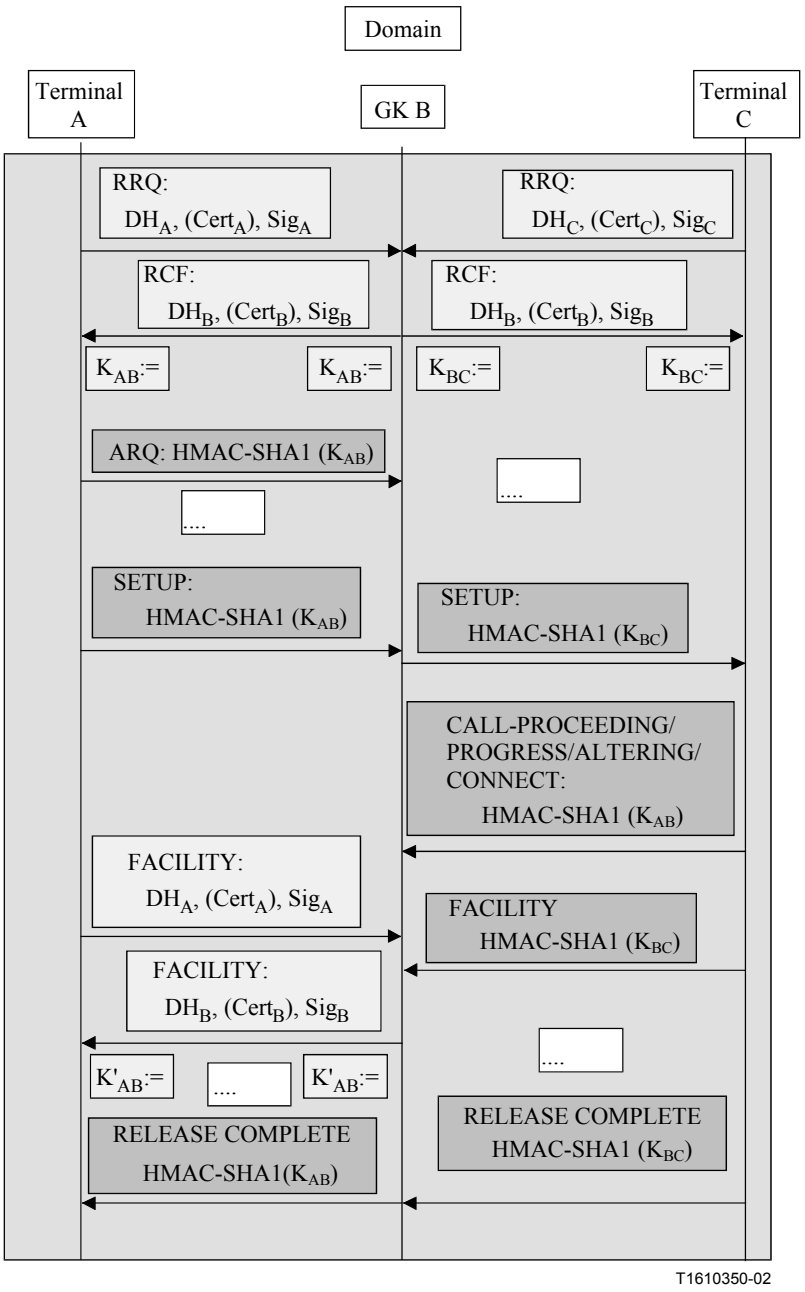
- **tokenOID** of the **ClearToken** within FACILITY shall be set to "Q" indicating use of Diffie-Hellman and hybrid security profile. Computation of the link key proceeds in the same manner as computation of the media session key (see D.7.1).

The FACILITY message for key update purposes shall be protected according to Annex E Procedure II. Any other FACILITY message without conveyed Diffie-Hellman token shall not be deployed for key update purposes and shall be protected according to Annex D Procedure I.

F.10 Illustration examples

The flow diagrams in Figures F.2 and F.3 illustrate usage of Annex F in a basic message flow. Note that the diagrams do not show the complete message flow and that several messages are omitted for simplicity. Messages highlighted in light gray relate to the signature profile Annex E, while dark gray messages relate to the baseline profile Annex D. The figures emphasize the (most important) security parts of each message (H.235 CryptoTokens, Tokens) while omitting details.

The flow diagram in Figure F.2 illustrates the basic message flow in a scenario with one gatekeeper within a single administrative domain. Assuming that the gatekeeper certificate is known to all the terminals involved, and that the terminals know the gatekeeper certificate likewise, there is no need to transmit the certificates in-band during the registration procedure.



Cert	User certificate	K, K'	symmetric link key
DH_A	Diffie-Hellman Token g^a mod p	Sig	digital signature
DH_B	Diffie-Hellman Token g^b mod p		
EP	Endpoint (Terminal)		
GK	Gatekeeper		

Figure F.2/H.235 – Flow diagram in a single administrative domain

NOTE 1 – Figures F.2 and F.3 also cover the fast start procedure when the call signalling messages SETUP and CALL PROCEEDING/PROGRESS/ALERTING/CONNECT include the faststart token (see 8.1.7/H.323). Otherwise, non-faststart mode is assumed according to 7.3.1/H.323. Figure F.2 shows also the key update procedure between Terminal A and Gatekeeper B using FACILITY.

Figure F.3 shows an example message flow in a scenario with different administrative domains. While the hybrid security profile is applied within each domain between terminal and gatekeeper as illustrated in Figure F.2, the hybrid security profile may be applied also between both domains during the call establishment phase.

NOTE 2 – Figure F.3 omits any communication among border elements (BE) and any communication between GK-to-BE. Figure F.3 also shows the key update procedure between both domains using FACILITY.

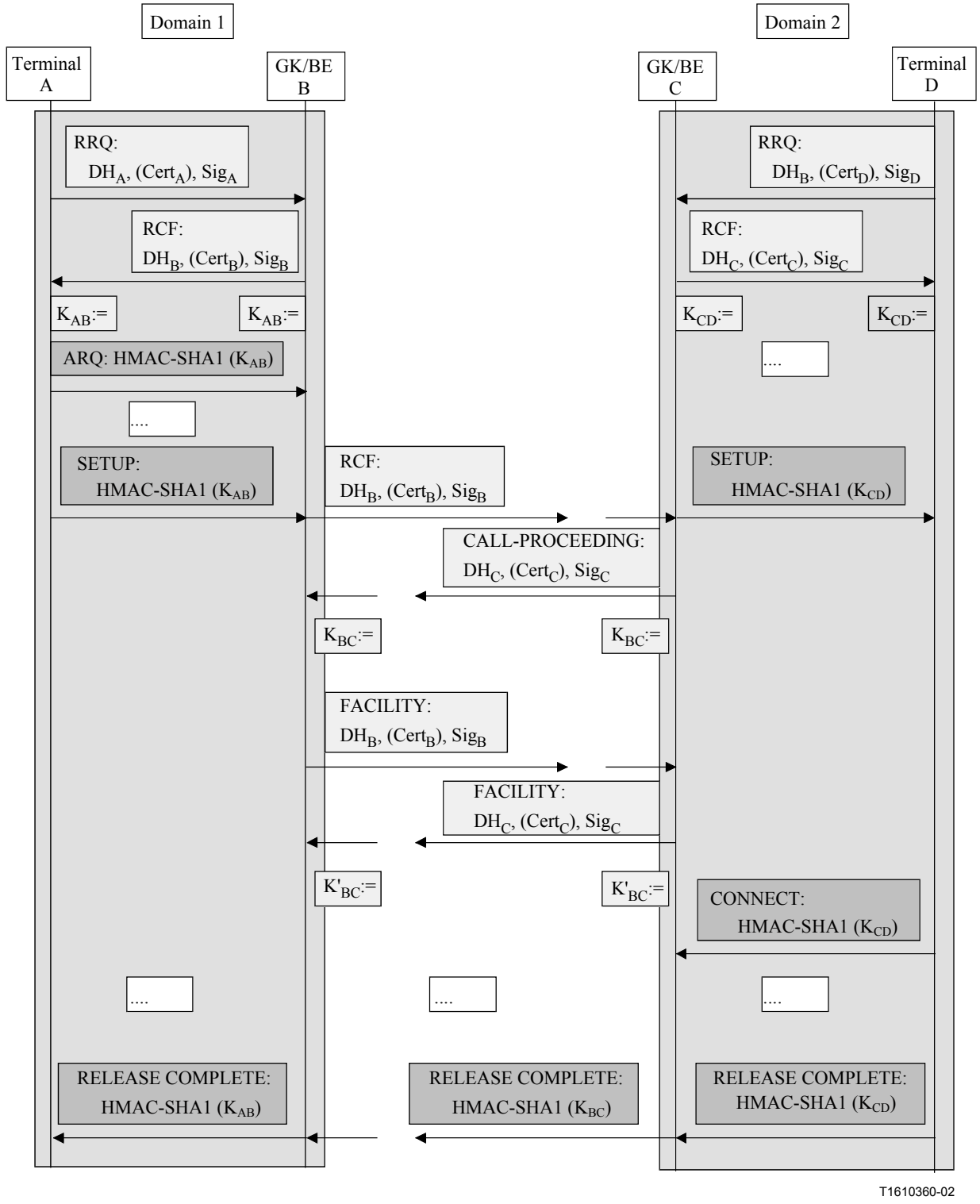


Figure F.3/H.235 – Flow diagram in a multi-administrative domain

F.11 Multicast behaviour

H.225.0 multicast messages such as **GRQ** or **LRQ** shall include a **CryptoToken** according to Procedure II where the **generalID** is not set. When such messages are sent unicast, then the message shall include a **CryptoToken** with the **generalID** set.

F.12 List of secure signalling messages

Procedure IV deploys Procedure I of Annex D or Procedure II of Annex E, depending on the scenario and on the actual message, as indicated below.

F.12.1 H.225.0 RAS

H.225.0 RAS message	H.235 signalling fields	Authentication and integrity	Non-repudiation
GatekeeperRequest, GatekeeperConfirm, GatekeeperReject if GK discovery is applied RegistrationRequest, RegistrationConfirm, RegistrationReject if GK discovery is not applied	CryptoToken, ClearToken	Procedure II	Procedure II
Any other RAS message (Note 2)	CryptoToken	Procedure I	
NOTE 1 – For unicast messages, Procedure II shall be applied with the security fields in the **CryptoToken** used. NOTE 2 – GK discovery and multicast messages are not sent.			

F.12.2 H.225.0 call signalling (single administrative domain)

H.225.0 call signalling message	H.235 signalling fields	Authentication and integrity	Non-repudiation
Setup-UUIE, Connect-UUIE (Note 1), Facility-UUIE (Note 2), Alerting-UUIE, CallProceeding-UUIE, Facility-UUIE, Progress-UUIE, Information-UUIE, ReleaseComplete-UUIE, Status-UUIE, StatusInquiry-UUIE, SetupAcknowledge-UUIE, Notify-UUIE	CryptoToken, ClearToken	Procedure I	
Facility-UUIE (Note 3)	CryptoToken	Procedure II	Procedure II
NOTE 1 – Assuming that either message is the first in each direction. NOTE 2 – Not used for key update. NOTE 3 – Used for key update.			

F.12.3 H.225.0 call signalling (multi-administrative domain)

H.225.0 call signalling message	H.235 signalling fields	Authentication and integrity	Non-repudiation
Setup-UUIE, Connect-UUIE (Note 1), Alerting-UUIE (Note 2), CallProceeding-UUIE, Facility-UUIE (Note 3), Progress-UUIE, Information-UUIE, ReleaseComplete-UUIE	CryptoToken, ClearToken	Procedure II	Procedure II
Alerting-UUIE (Note 4), CallProceeding-UUIE, Facility-UUIE (Note 5), Progress-UUIE, Information-UUIE, ReleaseComplete-UUIE, Status-UUIE, StatusInquiry-UUIE, SetupAcknowledge-UUIE, Notify-UUIE	CryptoToken, ClearToken	Procedure I	Procedure I
NOTE 1 – Assuming that either message is the first in each direction. NOTE 2 – Any of those messages occurs as first message in either direction. NOTE 3 – Used for key update. NOTE 4 – Any of those messages does not occur as the first message in either direction. NOTE 5 – Not used for key update.			

F.13 List of object identifiers

Table F.2 lists all the referenced OIDs.

Table F.2/H.235 – Object identifiers used by Annex F

Object identifier reference	Object identifier value(s)	Description
"A1"	{itu-t (0) recommendation (0) h (8) 235 version (0) 2 20}	Used as replacement for OID "A" in Procedure II of Annex E for the CryptoToken-tokenOID indicating that the RSA signature/hash includes all fields in the H.225.0 RAS or call signalling message (authentication and integrity).
"S1"	{itu-t (0) recommendation (0) h (8) 235 version (0) 2 21}	Used as replacement for OID "S" in Procedure II of Annex E for the ClearToken-tokenOID indicating that the ClearToken is being used for message authentication and integrity. This OID in the end-to-end CryptoToken implicitly indicates also use of DH during fast start.
"Q"	{itu-t (0) recommendation (0) h (8) 235 version (0) 2 22}	Used in Procedure IV indicating that the ClearToken on the hop-by-hop link carries a Diffie-Hellman token .
"W"	{itu-t (0) recommendation (0) h (8) 235 version (0) 2 23}	Used in Procedure IV as algorithm OID indicating use of an RSA-SHA1-based digital signature.

Annex G

Usage of the Secure Real-Time Transport Protocol (SRTP) in conjunction with the MIKEY key management protocol within H.235

This annex is left as for further study.

Annex H

RAS key management

This annex is left as for further study.

Annex I

Support of direct-routed calls

I.1 Scope

The purpose of this annex is to provide recommendations of security procedures for using direct-routed call signalling in conjunction with H.235 security profiles D and F.

This security profile is offered as an option and may complement the Annex D or Annex F security profiles.

This annex provides implementation details for clause B.6 using symmetric key management techniques.

NOTE – This annex currently features a security procedure for a limited scenario but may further develop more elaborate security procedures for the general case; this is left as for further study.

I.2 Introduction

H.323 is often deployed using the gatekeeper-routed model. For instance, using this model supports best billing and also other functionality. The widespread use of gatekeeper-routed call models is also the reason that within ITU-T Rec. H.235 different security profiles (such as Annex D, Annex E, Annex F) are defined with the focus on exactly this call model.

However, with the need to support an increasing number of parallel channels, the direct-routed call model with a gatekeeper could yield better performance and scalability properties. The advantage of this model is the utilization of a gatekeeper for registration, admission, address resolution, and bandwidth control, while performing the call establishment directly between the endpoints in an end-to-end fashion.

This annex describes the enhancements for Annex D baseline and for Annex F hybrid security profiles for supporting direct-routed calls with a gatekeeper.

I.3 Specification conventions

The object identifiers are referenced through a symbolic reference in the text (e.g., "I1"), clause I.12 lists the actual numeric values for the symbolic object identifiers, see also clause 5.

I.4 Terms and definitions

For the purposes of this Recommendation the definitions given in clause 3 of ITU-T Recs H.323, H.225.0, H.235 and X.800 apply along with those in this clause.

I.5 Symbols and abbreviations

This annex uses the following abbreviations:

$\{M\}_{K;\,S,\,IV}$ EOFB Encryption of *M* using secret key *K* and secret salting key *S* and initial vector *IV*

CT ClearToken

DRC Direct-Routed Call

EPID Endpoint Identifier

GKID Gatekeeper Identifier

K_{AG} Shared secret (Annex D, Annex F) between EP A and GK G

K_{BG} Shared secret (Annex D, Annex F) between EP B and GK G

KS_{AG} Secret, shared salting key between EP A and GK G

KS_{BG} Secret, shared salting key between EP B and GK G

K'_{AG} The encryption key shared between EP A and GK G.

K'_{BG} The encryption key shared between EP B and GK G.

K_{AB} The encryption key shared between EP A and EP B.

I.6 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; 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. The reference to a document within this Recommendation does not give it, as a stand-alone document, the status of a Recommendation

– ITU-T Recommendation H.225.0 (2003), *Call signalling protocols and media stream packetization for packet-based multimedia communications systems*.

– ITU-T Recommendation H.323 (2003), *Packet-based multimedia communications systems*.

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

– ISO/IEC 10118-3:2004, *Information technology – Security techniques – Hash functions – Part 3: Dedicated hash-functions*.

– IETF RFC 2104 (1997), HMAC: *Keyed-Hashing for Message Authentication*.

– IETF RFC 2246 (1999), *The TLS Protocol version 1.0*.

I.7 Overview

The Annex D baseline (see the main body of this Recommendation) as well as the Annex F hybrid security profiles (see Annex F) (after the first handshake) apply a shared secret to assure message authentication and/or integrity in a hop-by-hop fashion using the gatekeeper as a trusted intermediate host. Using the direct-routed call model, a shared secret between two endpoints cannot be assumed. It is also not practical to use a pre-established shared secret to secure the

communication since in this case all endpoints would have to know in advance which other endpoint will be called.

This annex addresses a scenario as shown in Figure I.1 where endpoints are attached to a single gatekeeper, and deploy direct-routed call signalling. The scenario assumes an unsecured IP network in the gatekeeper zone.

It is assumed that each endpoint has a communication relation and a security association with the gatekeeper, and that each endpoint has registered securely with the gatekeeper using either the baseline or the hybrid security profile.

Hence, the gatekeeper is able to provide a shared secret for the directly communicating endpoints using a Kerberos-like approach.

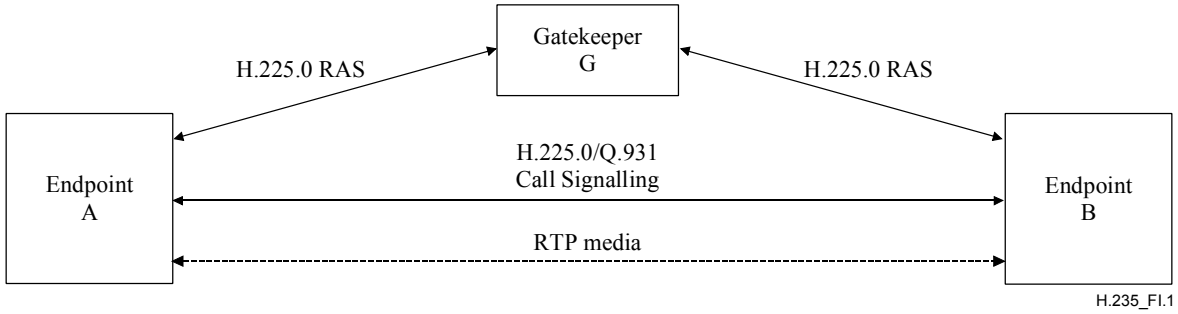


Figure I.1/H.235 – Direct-routed call scenario

I.8 Limitations

This annex currently does not address direct-routed scenarios where endpoints are attached to distinct gatekeepers. Further, this annex does not address direct-routed scenarios without any gatekeeper. This is all left as for further study.

I.9 Procedure DRC

Endpoints capable of supporting this security profile shall indicate this fact during GRQ and/or RRQ by including a separate ClearToken with **tokenOID** set to "I0"; any other fields in that ClearToken should not be used. The Annex I-capable gatekeeper that is willing to provide this functionality shall reply with GCF resp. RCF with a separate ClearToken included with **tokenOID** set to "I0" and all other fields in the ClearToken unused.

Before an endpoint A starts sending call signalling messages to another endpoint B directly, the endpoint A or B shall apply for admission at the gatekeeper G using ARQ. Endpoint A shall include within **ARQ** a separate ClearToken with **tokenOID** set to "I0" and all other fields in the ClearToken unused.

The gatekeeper, recognizing that endpoints A and B support this annex, shall generate key material and ClearTokens as specified below.

The gatekeeper is able to calculate a call-based shared secret K_{AB}, besides the normal ARQ operation. This call-based shared secret is then propagated to both endpoints using ClearTokens. Those ClearTokens are conveyed within the ACF message and are sent back to the caller.

Two ClearTokens shall be included, one CT_A for the caller A and another one CT_B for the callee B. Each **ClearToken** shall contain an OID ("I1" or "I2") within **tokenOID** that indicates whether the token is destined for the caller (OID "I1" for CT_A) or for the callee (OID "I2" for CT_B).

The **ClearToken** as defined in this annex may be used in conjunction with other security profiles such as with Annex D or with Annex F that deploy **ClearTokens** as well. In such a case, Annex I ClearToken shall use those other **ClearToken** fields too. For example, in order to use Annex I in conjunction with Annex D, the fields **timestamp**, **random**, **generalID**, **sendersID**, and **dhkey** shall be present and shall be used, as described by the Annex D security profiles.

The gatekeeper ID (GKID) shall be placed within **sendersID** whereas **generalID** shall hold either the endpoint identifier of endpoint A (CT_A) or of endpoint B (CT_B).

K' denotes the encryption key that is shared between and endpoint and the GK. The encryption keys K'_{AG} and K'_{BG} for the encrypted end-to-end key K_{AB} shall be derived from the shared secret between the gatekeeper and the endpoints (K_{AG} or K_{BG}) using the PRF-**based** key derivation procedure as defined in clause I.10 where **keyDerivationOID** in **V3KeySyncMaterial** shall hold "Annex I-HMAC-SHA1-PRF", see clause I.12.

The gatekeeper shall generate a common shared session secret K_{AB}, which is shared between endpoint A and endpoint B.

This session secret K_{AB} shall be encrypted by K'_{AG} (for CT destined to endpoint A) or by K'_{BG} (for the CT destined to endpoint B) using an encryption algorithm.

The enhanced OFB (EOFB) encryption mode (see B.2.5) shall be used with the secret, endpoint-specific salting key. Applicable encryption algorithms are (see clause D.11):

- DES (56 bits) in EOFB mode using OID "Y1": optional;
- 3DES (168 bits) in outer-EOFB mode using OID "Z1": optional;
- AES (128 bits) in EOFB mode using OID "Z2": default and recommended;
- RC2-compatible (56 bits) in EOFB mode using OID "X1": optional.

For the EOFB encryption mode, the GK shall generate a random initial value IV. For OID "X1", OID "Y1" and OID "Z1" the IV has 64 bits and shall be conveyed within **iv8** of **paramS** within **V3KeySyncMaterial;** whereas the IV has 128 bits for OID "Z2" and shall be conveyed within **iv16** of **paramS** within **V3KeySyncMaterial**.

The obtained ciphertext $\{K_{AB}\}_{K'_{AG}, KS_{AG}, IV}$ resp. $\{K_{AB}\}_{K'_{BG}, KS_{BG}, IV}$ shall then be conveyed in the **h235key** data structure as part of **secureShareSecret** where it shall be placed within the **encryptedSessionKey** of the **secureSharedSecret** data structure. The encryption algorithm shall be indicated in **algorithmOID** ("X1", "Y1", "Z1" or "Z2") within **V3KeySyncMaterial**.

For the ClearToken destined to endpoint A, the endpoint identifier of endpoint B ($EPID_B$) shall be placed within **generalID** of **V3KeySyncMaterial**. Likewise for the ClearToken destined to endpoint B, the endpoint identifier of endpoint A ($EPID_A$) shall be placed within **generalID** of **V3KeySyncMaterial**.

For the EOFB encryption algorithms, **encryptedSaltingKey** shall not be used.

The gatekeeper shall include both ClearTokens CT_A and CT_B in the **ACF** towards endpoint A.

Endpoint A shall identify CT_A by inspection of the **tokenOID** "I1" within ClearToken.

Endpoint A shall verify that the obtained CT_A is fresh by checking the **timestamp**. Further security checks shall verify the **generalID** and **sendersID** of the ClearToken and **generalID** within **V3KeySyncMaterial**. If the received CT_A was verified as being fresh, endpoint A shall retrieve the IV and compute K'_{AG} and KS_{AG} as described above for the gatekeeper. Endpoint A shall decrypt the **encryptedSessionKey** information found within **V3KeySyncMaterial** of CT_A to obtain K'_{AB}.

If the received CT_A was verified as being fresh, endpoint A is able to send a SETUP message to endpoint B. This SETUP message includes CT_B. The SETUP message shall be secured (authenticated and/or integrity protected) according to Annex D or according to Annex F using K_{AB}

as the applied shared secret. For this, **generalID** in the Annex D hashed ClearToken (not CT_B!) shall be set to $EPID_B$.

Endpoint B shall identify CT_B by inspection of the **tokenOID** "I2" within ClearToken.

Endpoint B shall verify that the obtained CT_B is fresh by checking the **timestamp**. Further security checks shall verify the **generalID** and **sendersID** of the ClearToken and **generalID** within **V3KeySyncMaterial**. If the received CT_B was verified as being fresh, endpoint B shall retrieve the IV and compute K'_{BG} and KS_{BG} as described above for the gatekeeper. Endpoint B shall decrypt the **encryptedSessionKey** information found within **V3KeySyncMaterial** of CT_B to obtain K'_{AB}.

In case CT_B was verified as being fresh, endpoint B is able to proceed the call signalling by replying with CALL-PROCEEDING, ALTERING or CONNECT etc., as appropriate. In case CT_B was found not to be fresh or the security verification of the SETUP message failed, endpoint B shall reply with RELEASE-COMPLETE and the **ReleaseCompleteReason** set to a security error as defined by B.2.2.

When media security is to be deployed (see clause D.7), endpoint A and endpoint B shall exchange Diffie-Hellman half-keys according to D.7.1 and establish a dynamic session-based master key from which media-specific session keys can then be derived.

Figure I.2 shows the basic communication flow:

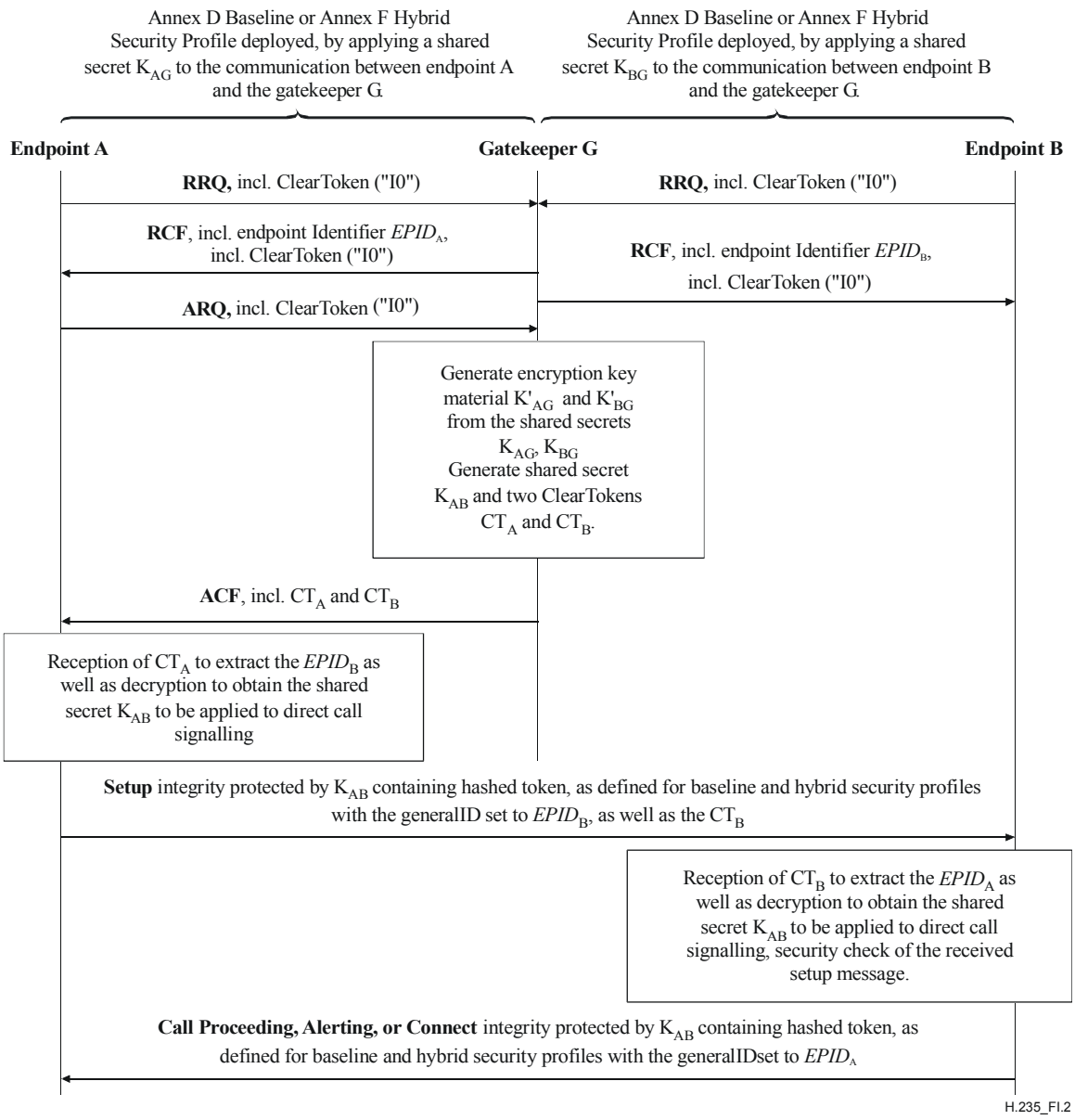


Figure I.2/H.235 – Basic communication flow

I.10 PRF-based key derivation procedure

This clause describes a procedure that defines how to derive key material from the shared secret and other parameters.

The encryption key K'_{AG} shall be computed using the PRF (see clause B.7) with the *inkey* parameter set to K_{AG} and *label* shall be set to the constant 0x2AD01C64 || **challenge**.

Likewise, the encryption key K'_{BG} shall be computed using that PRF with the *inkey* parameter set to K_{BG} and *label* shall be set to the constant 0x1B5C7973 || **challenge**. In both cases, *outkey_len* shall be set to the length of the required length of the encryption key for the chosen encryption algorithm.

Using that same PRF, a secret, shared salting key shall be generated by the gatekeeper and by each endpoint. The salting key, when being used in conjunction with the EOFB encryption mode, guards against known-plaintext attacks of the CT_B by EP A where EP A might otherwise attempt to discover K_{BG}.

KS_{AG} denotes the secret, shared salting key that is shared between EP A and the GK G. KS_{AG} shall be computed using the PRF with the *inkey* parameter set to K_{AG} and *label* shall be set to the constant 0x150533E1 || **challenge**. KS_{BG} shall be computed using PRF with the *inkey* parameter set to K_{BG} and *label* shall be set to the constant 0x39A2C14B || **challenge**.

NOTE – The 32-bit constant integers (i.e., 0x2AD01C64 etc.) are taken from the decimal digits of *e* (i.e., 2.7182...), and where each constant consists of nine decimal digits (e.g., the first nine decimal digits 718281828 = 0x2AD01C64). The strings of nine decimal digits are not chosen at random, but as consecutive "chunks" from the decimal digits of *e*.

I.11 FIPS-140-based key derivation procedure

This clause may describe a procedure that defines how to derive key material from a shared secret and other parameters using a FIPS-140 compliant crypto module. This is left as for further study.

I.12 List of object identifiers

Table I.1/H.235 – Object identifiers used by H.235 Annex I

Object identifier reference	Object identifier value	Description
"I0"	{itu-t (0) recommendation (0) h (8) 235 version (0) 3 48}	Used in procedure DRC during GRQ/RRQ and GCF/RCF and ARQ to let the EP/GK indicate support of Annex I.
"I1"	{itu-t (0) recommendation (0) h (8) 235 version (0) 3 49}	Used in procedure DRC for the ClearToken tokenOID indicating that the ClearToken holds an end-to-end key for the caller.
"I2"	{itu-t (0) recommendation (0) h (8) 235 version (0) 3 50}	Used in procedure DRC for the ClearToken tokenOID indicating that the ClearToken holds an end-to-end key for the callee.
"Annex I-HMAC-SHA1-PRF"	{itu-t (0) recommendation (0) h (8) 235 version (0) 3 51}	Used in procedure DRC for keyDerivationOID within V3KeySyncMaterial to indicate the applied key derivation method in I.10 using the HMAC-SHA1 pseudo-random function.

Appendix I

H.323 implementation details

I.1 Ciphertext padding methods

There is a description of Ciphertext Stealing in [Schneier], pages 191 and 196. Figures I.1 to I.5 illustrate the technique.

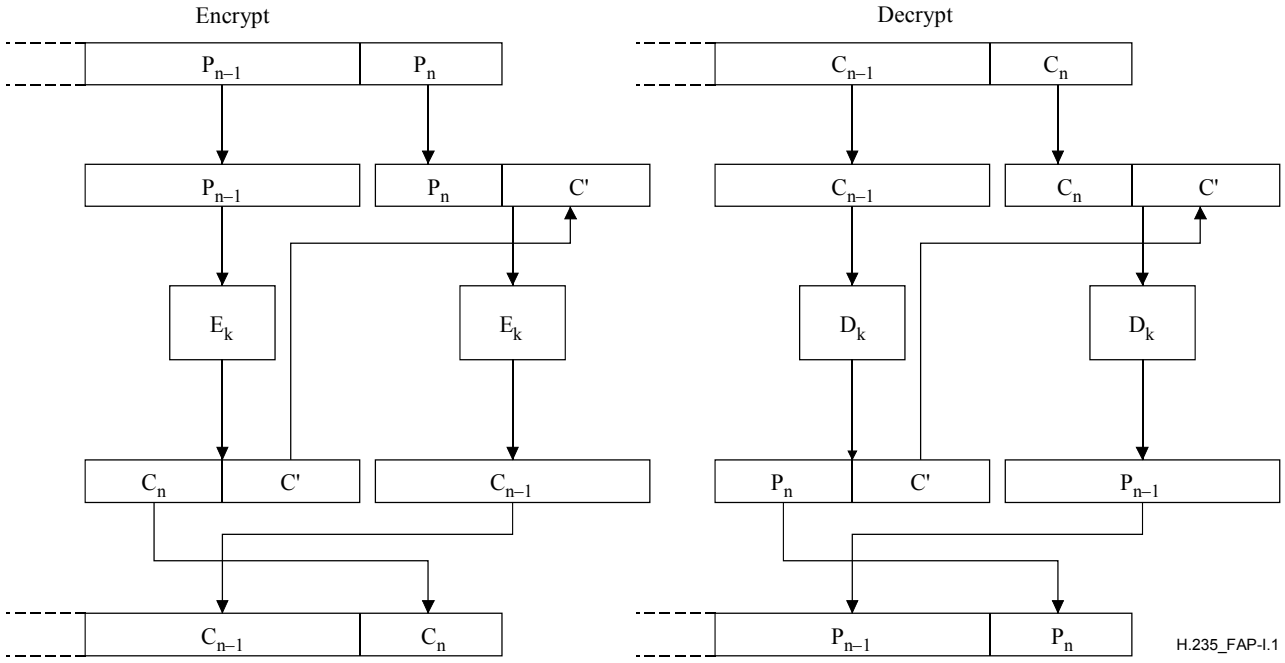


Figure I.1/H.235 – Ciphertext stealing in ECB mode

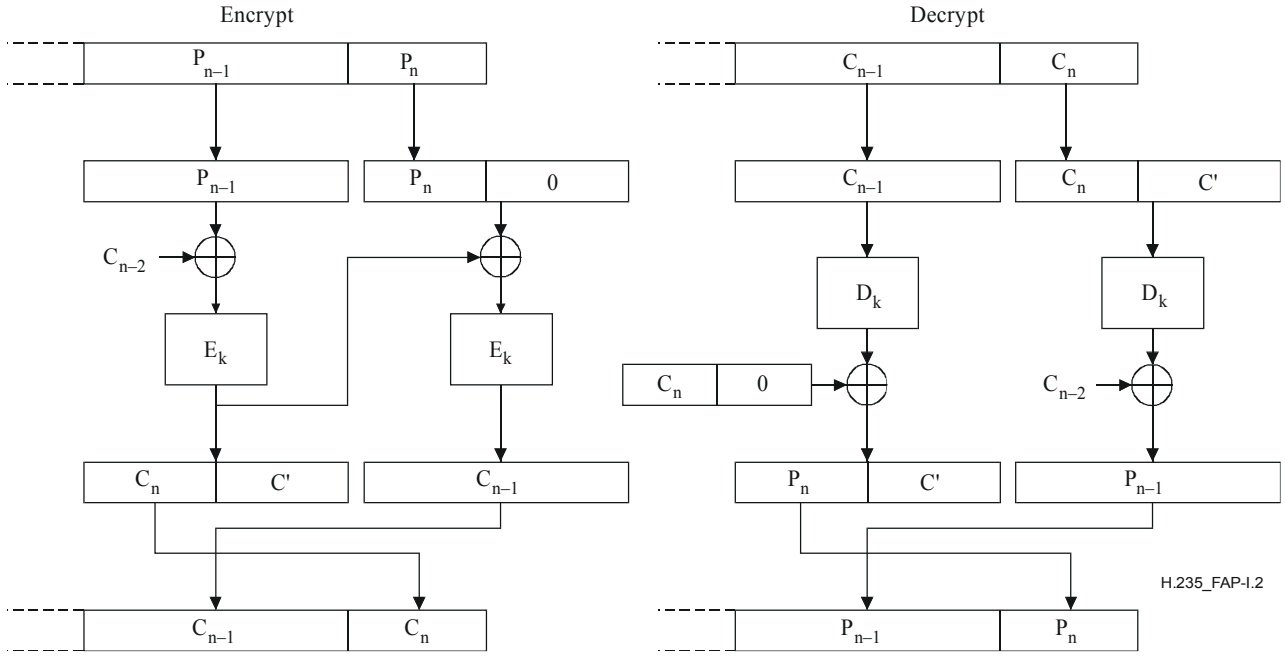


Figure I.2/H.235 – Ciphertext stealing in CBC mode

NOTE – Ciphertext stealing in ECB or CBC modes requires the payload to convey at least one complete block. Implementations deploying ciphertext stealing in ECB mode or CBC mode should ascertain that the payload conveys always at least one crypto block; e.g., by proper choice of the sampling/packetization rate or selection of the encryption algorithm.

In case the payload spans less than one single block, the initial vector (IV) shall be used as the previous ciphertext block when ciphertext stealing mode is applied in CBC mode.

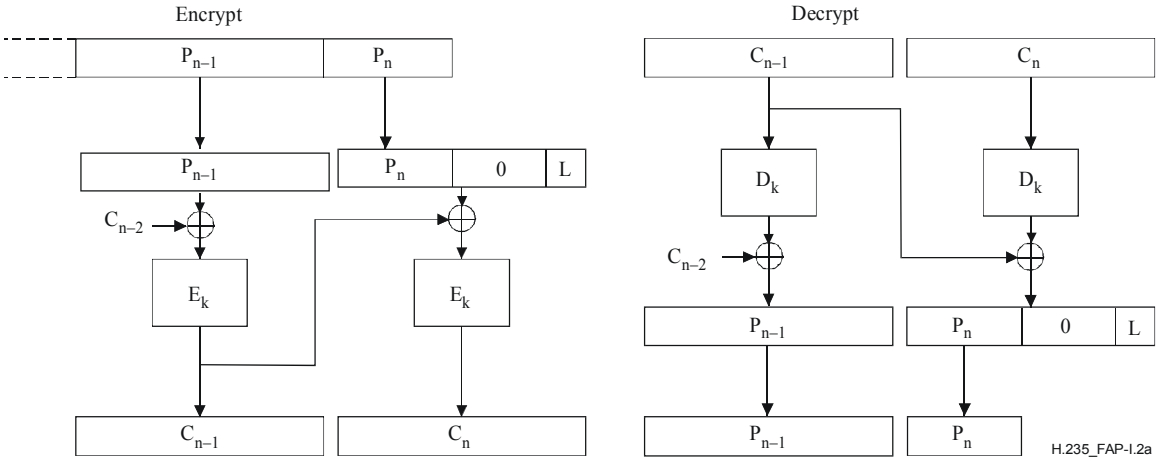


Figure I.2a/H.235 – Zero padding in CBC mode

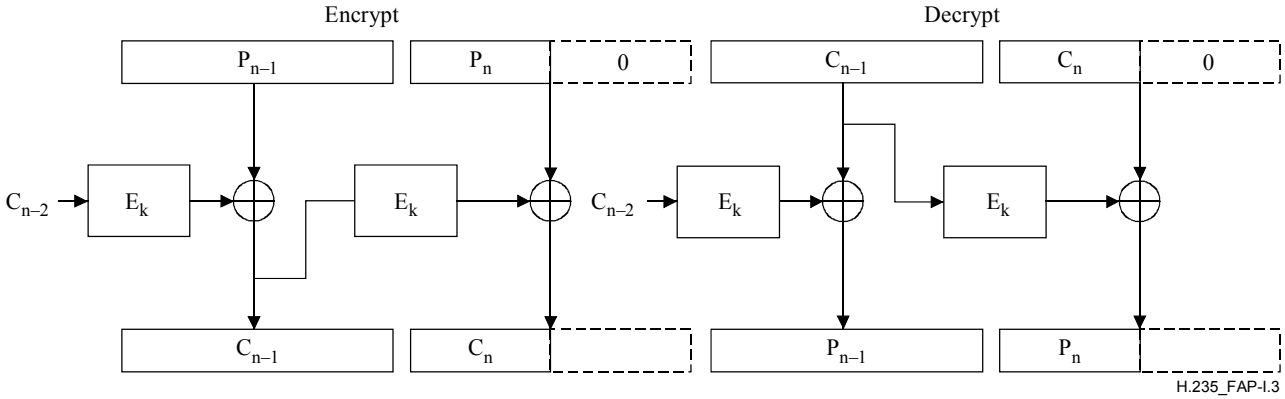
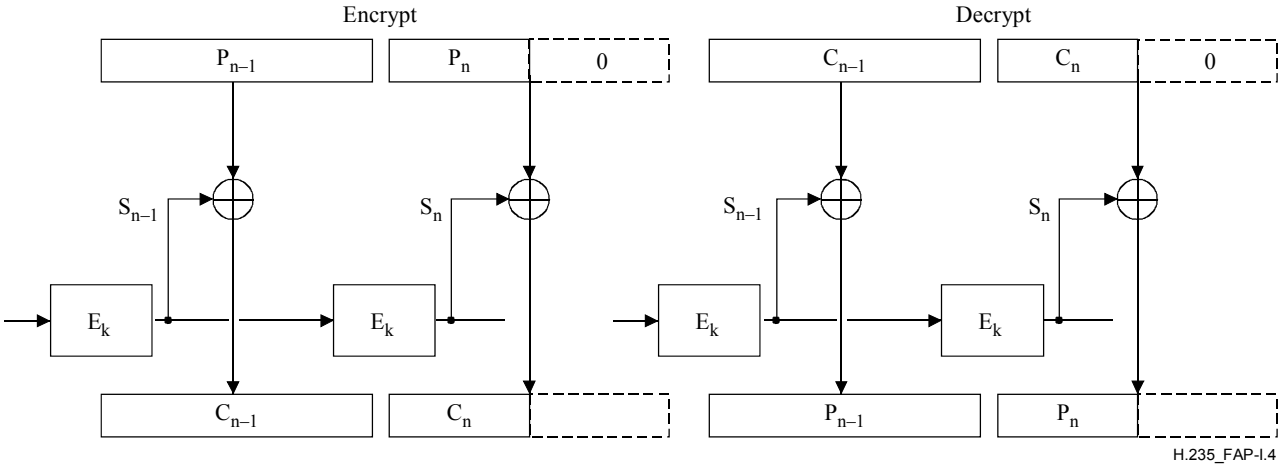


Figure I.3/H.235 – Zero padding in CFB mode



NOTE – S_i is the result of repetitive encryption (i.e. permutations) of the IV.

Figure I.4/H.235 – Zero padding in OFB mode

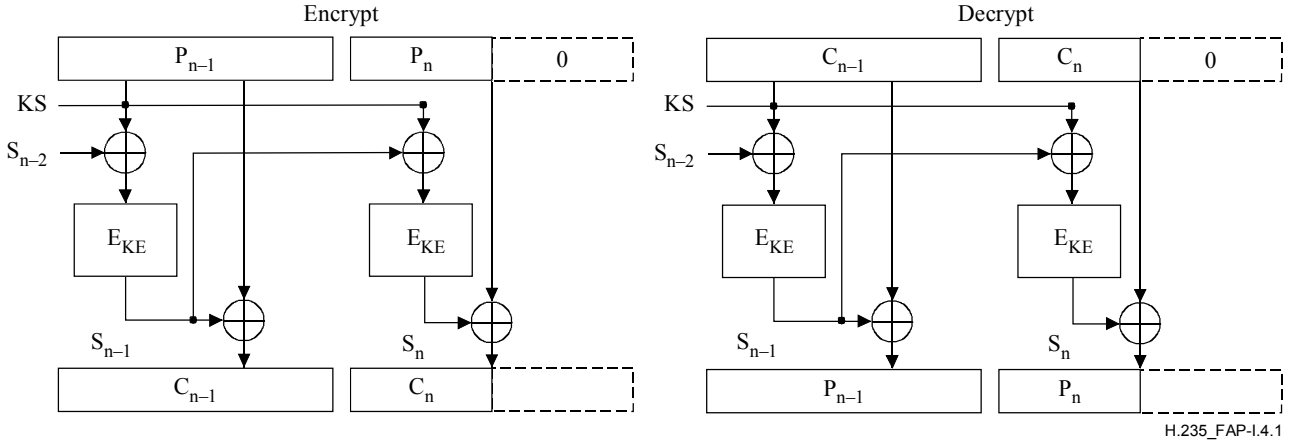


Figure I.4.1/H.235 – EOFB mode with zero padding

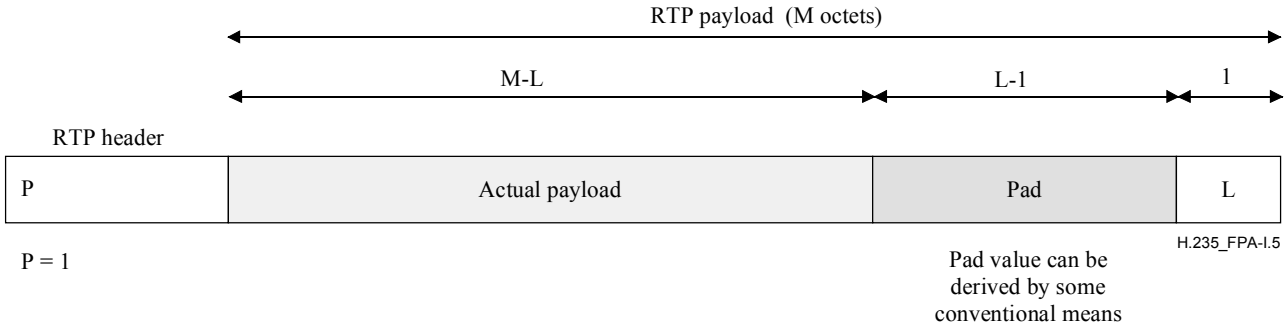


Figure I.5/H.235 – Padding as prescribed by RTP

I.2 New keys

The procedures outlined in 8.5/H.323 are completed by an MC to eject a participant from a conference. The master may generate new encryption keys for the logical channels (and not distribute them to the ejected party); this may be used to keep the ejected party from monitoring the media streams.

I.3 H.323 trusted elements

In general, MC(U)s, gateways, and gatekeepers (if implementing the gatekeeper-routed model) are trusted with respect to the privacy of the control channel. If the connections establishment channel (H.225.0) is secured *and* routed through the gatekeeper, it must also be trusted. If any of these H.323 components must operate on the media streams (i.e., mixing, transcoding) then, by definition, they shall also be trusted for the media privacy.

Firewall proxies (though not H.323-specific elements) may also be trusted, since they terminate connections, and may well have to manipulate the messages and media streams.

I.4 Implementation examples

These following subclauses describe example implementations that might be developed within the H.235 framework. These are not intended to constrain the many other possibilities available within this Recommendation, but rather to give more concrete examples of usage within ITU-T Rec. H.323.

I.4.1 Tokens

This clause will describe an example usage of security tokens to obscure or hide destination addressing information. The example scenario is an endpoint which wishes to make a call to another endpoint utilizing its well-known alias. More specifically, this involves an H.323 endpoint, gatekeeper, POTS-gateway, and telephone as illustrated in Figure I.6.

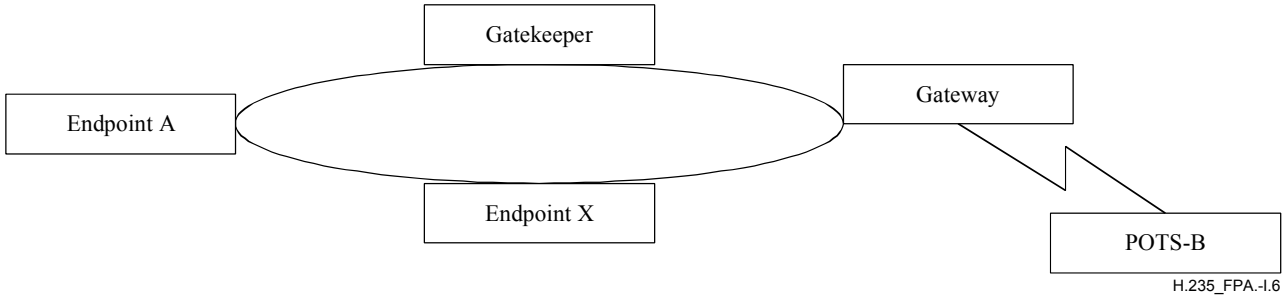


Figure I.6/H.235 – Tokens

Currently, H.323 may operate in a manner similar to a telephone network with caller-ID. This scenario will illustrate a situation in which the *caller* does not want to expose its physical address, while still allowing the call to complete. This may be important in POTS-H.323 gateways, where the target phone number may need to stay private.

Assume that EPA is trying to call POTS-B, and POTS-B does not want to expose its E.164 phone number to EPA. (How this policy is established is beyond the scope of this example.)

- EPA will send an ARQ to its gatekeeper to resolve the address of the POTS telephone as represented by its alias/GW. The gatekeeper would recognize this as a "private" alias, knowing that in order to complete the connection it must return the POTS-gateway address (similar to returning the address of an H.320 gateway if an H.320 endpoint is called by an H.323 endpoint).
- In the returned ACF, the gatekeeper returns the POTS-gateway's address as expected. The addressing information that is required to dial to the end telephone (i.e., the telephone number) is returned in an encrypted token included in the ACF. This encrypted token contains the actual E.164 (phone number) of the telephone which cannot be deciphered nor understood by the caller (i.e., EPA).

- The endpoint issues the SETUP message to the gateway device (whose call signalling address was returned in the ACF) including the opaque token(s) that it received with the ACF.
- The gateway, upon receiving the SETUP, issues its ARQ to its gatekeeper, including any token(s) that were received in the SETUP.
- The gatekeeper is able to decipher the token(s) and return the phone number in the ACF.

Partial ASN.1 of an example token structure is shown below, with the field contents described. Assume we utilize the **cryptoEncodedGeneralToken** to contain the encrypted telephone number.

An implementation might choose a **tokenOID** denoting this token as containing the E.164 phone number. The particular method that is used to encrypt this phone number (for example, 56-bit DES) would be included in the "ENCRYPT" definition **algorithmOID**.

```
CryptoToken::= CHOICE
{
     cryptoEncodedGeneralToken SEQUENCE    -- General purpose/application
                                           --  specific token
     {
          tokenOID  OBJECT IDENTIFIER,
          ENCRYPTED { EncodedGeneralToken }
     },
.
.
. [abbreviated text]
.

}
```

The **CryptoToken** would be passed in the SETUP (from EPA to GW) and the **ARQ** (from the GW to the gatekeeper) messages as outlined above. After the gatekeeper decrypted the token (the telephone number) it would pass the clear version of this in the **clearToken**.

I.4.2 Token usage in H.323 systems

There has been some confusion on the usage of individual **CryptoH323Tokens** as passed in RAS messages. There are two main categories of **CryptoH323Tokens**: those used for H.235 procedures and those used in an application-specific manner. The use of these tokens should be according to the following rules:

- All H.235-defined (e.g., **cryptoEPPwdHash**, **cryptoGKPwdHash**, **cryptoEPPwdEncr**, **cryptoGKPwdEncr**, **cryptoGKCert**, and **cryptoFastStart**) shall be utilized with the procedures and algorithms as described in this Recommendation.
- Application-specific or proprietary use of tokens shall utilize the **nestedcryptoToken** for their exchanges.
- Any **nestedcryptoToken** used should have a **tokenOID** (object identifier) which unambiguously identifies it.

I.4.3 H.235 random value usage in H.323 systems

The random value that is passed in xRQ/xCF sequence between endpoints and gatekeepers may be updated by the gatekeeper. As described in B.4.2, this random value may be refreshed in any xCF message to be utilized by a subsequent xRQ messages from the endpoint. Due to the fact that RAS messages may be lost (including xCF/xRJ), the updated random value may also be lost. The recovery from this situation may be the reinitializing of the security context but is left to local implementation.

Implementations that require the use of multiple outstanding RAS requests will be limited by the updating of the random values used in any authentication. If the updating of this value occurs on every response to a request, parallel requests are not possible. One possible solution is to have a logical "window" during which a random value remains constant. This issue is a local implementation matter.

I.4.4 Password

In this example, it is assumed that the user is a subscriber to the gatekeeper (i.e., the user will be in its zone) and has an associated subscription ID and password. The user would register with the gatekeeper using the subscription ID (as passed in an alias – H323ID) and encrypting a challenge string presented by the gatekeeper. This assumes that the gatekeeper also knows the password associated with the subscription ID. The gatekeeper will authenticate the user by verifying that the challenge string was correctly encrypted.

The example registration procedure with gatekeeper authentication is as follows:

1) If the endpoint uses **GRQ** to discover a gatekeeper, one of the aliases in the message would be the subscription ID (as an **H323ID**). The **authenticationcapability** would contain an **AuthenticationMechanism** of **pwdSymEnc** and the **algorithmOIDs** would be set to indicate the entire set of encryption algorithms supported by the endpoint. (For example, one of these would be 56-bit DES in EBC mode.)

2) The gatekeeper would respond with **GCF** (assuming it recognizes the alias) carrying a **tokens** element containing one **ClearToken**. This **ClearToken** would contain both a **challenge** and a **timeStamp** element. The **challenge** would contain 16 octets. (To prevent replay attacks, the **ClearToken** should contain a **timeStamp**.) The **authenticationmode** should be set to **pwdSymEnc** and the **algorithmOID** should be set to indicate the encryption algorithm required by the gatekeeper (for example, 56-bit DES in EBC mode).

   If the gatekeeper does not support any of the **algorithmOIDs** indicated in the **GRQ**, then it would respond with a **GRJ** containing a **GatekeeperRejectReason** of **resourceUnavailable**.

3) The endpoint application should then attempt to register with (one of) the GK(s) that responded with a **GCF** by sending an **RRQ** containing a **cryptoEPPwdEncr** in the **cryptoTokens**. The **cryptoEPPwdEncr** would have the **algorithmOID** of the encryption algorithm agreed to in the **GRQ**/**GCF** exchange, and the encrypted challenge.

   The encryption key is constructed from the user's password using the procedure described in 10.3.2. The resulting octet "string" is then used as the DES key to encrypt the **challenge**.

4) When the gatekeeper receives the encrypted challenge in the **RRQ**, it would compare it to an identically generated encrypted challenge to authenticate the registering user. If the two encrypted strings do not match, the gatekeeper should respond with an **RRJ** with the **RegistrationRejectReason** set to **securityDenial** or other appropriate security error code according to B.2.2. If they match, the gatekeeper sends an **RCF** to the endpoint.

5) If the gatekeeper receives an **RRQ** which does not contain an acceptable **cryptoTokens** element, then it should respond with an **RRJ** with a **GatekeeperRejectReason** of **discoveryRequired**. The endpoint, upon receiving such an **RRJ**, may perform discovery which will allow the gatekeeper/endpoint to exchange a new challenge.

   NOTE – The **GRQ** message may be unicast to the gatekeeper.

I.4.5 IPSEC

In general, IPSEC [IPSEC] can be used to provide authentication and, optionally, confidentiality (i.e., encryption) at the IP layer transparent to whatever (application) protocol runs above. The application protocol does not have to be updated to allow this; only security policy at each end.

For example, to make maximum use of IPSEC for a simple point-to-point call, the following scenario could be followed:

1) The calling endpoint and its gatekeeper would set policy to require the use of IPSEC (authentication and, optionally, confidentiality) on the RAS protocol. Thus, before the first RAS message is sent from the endpoint to the gatekeeper, the ISAKMP/Oakley daemon on the endpoint will negotiate security services to be used on packets to and from the RAS channel's well-known port. Once negotiation is complete, the RAS channel will operate exactly as if it were not secured. Using this secure channel the gatekeeper will inform the endpoint of the address and port number of the call signalling channel in the called endpoint.

2) After obtaining the address and port number of the call signalling channel, the calling endpoint would dynamically update its security policy to require the desired IPSEC security on that address and protocol/port pair. Now, when the calling endpoint attempts to contact this address/port, the packets would be queued while an ISAKMP/Oakley negotiation is performed between the endpoints. Upon completion of this negotiation, an IPSEC Security Association (SA) for the address/port will exist and the Q.931 signalling can proceed.

3) On the Q.931 SETUP and CONNECT exchange, the endpoints can negotiate the use of IPSEC for the H.245 channel. This will allow the endpoints to again dynamically update their IPSEC policy databases to force the use of IPSEC on that connection.

4) As with the call signalling channel, a transparent ISAKMP/Oakley negotiation will take place before any H.245 packets are transmitted. The authentication performed by this ISAKMP/Oakley exchange will be the initial attempt at user-to-user authentication, and will set up a (probably) secure channel between the two users on which to negotiate the characteristics of the audio channel. If, after some person-to-person Q&A, either user is not satisfied with the authentication, different certificates can be chosen and the ISAKMP/Oakley exchange repeated.

5) After each H.245 ISAKMP/Oakley authentication, new keying material is exchanged for the RTP audio channel. This keying material is distributed by the master on the secure H.245 channel. Because the H.245 protocol is defined for the master to distribute the media keying material on the H.245 channel (to allow for multipoint communication), it is not recommended that IPSEC be used for the RTP channel.

An encrypted H.245 channel is a potential problem for proxy or NAT firewall, since the dynamically-assigned port numbers are carried in the H.245 protocol. Such firewalls would have to decipher, modify and re-encipher the protocol to operate correctly. For this reason, the "Security" Logical Channel was introduced into ITU-T Rec. H.245. If this channel is used, the H.245 channel can remain unsecured; authentication and key-generation would be done with the "Security" Logical Channel. Logical channel signalling would allow this channel to be protected with IPSEC, and the secret key used on the "Security" Logical Channel would be used to protect the **EncryptionSync** distributed by the master on the H.245 channel.

I.4.6 Back-end service support

Back-end servers are an important supplementary function in an overall H.323-based multimedia environment. For example, BES provides services for user authentication, for service authorization, also for accounting, charging and billing and other services. In a simple model, the gatekeeper could provide such services. In a decomposed architecture, the GK may not always provide such services; either because it may not have access to the BES databases, or it may be part of a different administrative domain. Likewise, the terminal or user usually does not know their BES.

Figure I.7 shows a scenario with a multimedia terminal (e.g., a SASET), a gatekeeper and linked BES. It is not within the scope of ITU-T Rec. H.323 as to how exactly the BES communicates with

the GK. Several methods and protocols could be applicable: RADIUS (see RFC 2865) is considered as one of the most important ones, which is widely deployed by service providers.

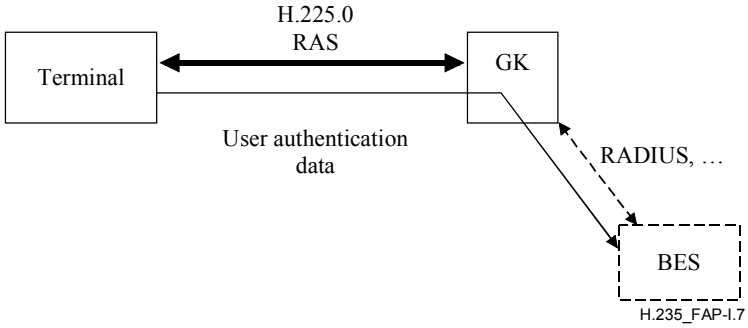


Figure I.7/H.235 – Scenario with Back-end Server

A GK offering BES support should support at least the following two modes:

1) **default mode**: in this mode, the terminal does not know the BES, and requires a trust relationship with the GK. The terminal sends the user authentication data in encrypted form (**cryptoEncryptedToken**) to the GK, which decrypts it, extracts the user authentication information and applies it towards the BES. The password-based encryption of the **ClearToken** is accomplished by applying a distinct secret that is shared between the terminal and the GK to the **CryptoToken**. The encryption key could be derived from the password with which the terminal securely registers at the GK.

   CryptoToken carries **cryptoEncryptedToken** where **tokenOID** is set to "M" indicating BES default mode; and **token** holding:

   - **algorithmOID** indicating the encryption algorithm; "Y" (DES56-CBC), "Z" (3DES-ocbc); see clause D.11;
   - **paramS** unused;
   - **encryptedData** set to the octet representation of the encrypted **ClearToken**.

   The **ClearToken** holds as **password** the user authentication data. Protected **ClearToken** information could be password/PIN, user identification, prepaid calling card number and credit card number. The **timestamp** is set to the current time of the terminal, **random** contains a monotonically increasing sequence number, **sendersID** is set to the terminal ID and **generalID** to the GK identifier. The initial value of the encryption algorithm shall be kept constant; it could be part of the terminal subscription secret.

   NOTE – The **ClearToken** is not transmitted.

2) **RADIUS mode**: in this mode, the BES and the terminal user share a common secret and the GK should not be trusted for the BES RADIUS authentication. The GK simply forwards a RADIUS challenge received from the BES within *Access-Challenge* towards the terminal and sends the user's response as a RADIUS response within *Access-Request* in the reverse direction. Terminal and GK negotiate this **radius** challenge/response capability in **AuthenticationBES** within **AuthenticationMechanism** during gatekeeper discovery.

   Upon receipt of a RADIUS *Access-Challenge* message conveying a challenge, the GK puts the 16-octet challenge in the **challenge** field of the **ClearToken** when querying the terminal with a **GCF** or any other RAS message. The **tokenOID** 'K' in the **ClearToken** indicates a RADIUS challenge.

   The terminal may then present the challenge to the user and wait for the response entered. The terminal shall reply with a RAS message where the response is put into the **challenge**

field of the **ClearToken**. The **tokenOID** 'L' in the **ClearToken** indicates a RADIUS response.

Table I.1 lists all the referenced OIDs.

Table I.1/H.235 – Object identifiers used by I.4.6

Object identifier reference	Object identifier value	Description
"K"	{itu-t (0) recommendation (0) h (8) 235 version (0) 2 31}	indicates a RADIUS challenge in the ClearToken
"L"	{itu-t (0) recommendation (0) h (8) 235 version (0) 2 32}	indicates a RADIUS response (conveyed in the challenge field) in the ClearToken
"M"	{itu-t (0) recommendation (0) h (8) 235 version (0) 2 33}	indicates BES default mode with a protected password in the ClearToken

Appendix II

H.324 implementation details

For further study.

Appendix III

Other H-series implementation details

For further study.

Appendix IV

Bibliography

[Daemon] DAEMON (J.), *Cipher and Hash function design*, Ph.D. Thesis, Katholieke Universiteit Leuven, March 1995.

[IPSEC] MAUGHAN (D.), SCHERTLER (M.), SCHNEIDER (M.), TURNER (J.), *Internet Security Association and Key Management Protocol (ISAKMP)*, draft-ietf-ipsec-isakmp-08.text, *Internet Engineering Task Force*, 1997.

[ISO | IEC 14888-3] *Information technology – Security techniques – Digital signatures with appendix; Part 3: Certificate-based mechanisms*, 1998.

[J.170] ITU-T J.170 (2002), *IPCablecom security specification*.

[MIKEY] ARKKO (J.), CARRARA (E.), LINDHOLM (F.), NASLUND (M.), NORRMAN (K.), "*MIKEY: Multimedia Internet KEYing*", Internet Draft <draft-ietf-msec-mikey-06.txt>, RFC xxxx, Work in Progress (MSEC WG), IETF, 02/2003.

{Editor's note: This RFC # will be included when available}

[PKCS] PKCS #1 v2.0: *RSA Cryptography Standard;* RSA Laboratories; October 1, 1998; http://www.rsa.com/rsalabs/pubs/PKCS/index.html.

[PKCS] PKCS #7: *Cryptographic Message Syntax Standard*, An RSA Laboratories Technical Note, version 1.5, Revised November 1, 1993; http://www.rsa.com/rsalabs/pubs/PKCS/index.html

[RTP] SCHULZRINNE (H.), CASNER (S.), FREDERICK (R.), JACOBSON (V.), *RTP: A transport Protocol for Real-Time Applications*, RFC 3550, *Internet Engineering Task Force*, 2003.

[Schneier] SCHNEIER (B.), *Applied Cryptography: Protocols, Algorithms, and Source Code in C*, 2nd Edition, John Wiley & Sons, Inc., 1995.

[SRTP] Baugher, McGrew, Oran *et al*: *The Secure Real-Time Transport Protocol*; draft-ietf-avt-srtp-07.txt, RFC xxxx; Internet Engineering Task Force, 2003.

{Editor's note: This RFC # will be included when available}

[TLS] DIEKS (T.), ALLEN (C.): *The TLS Protocol Version 1.0*, RFC 2246, *Internet Engineering Task Force*, 1999.

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	Cable networks and 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, Internet protocol aspects and Next Generation Networks
Series Z	Languages and general software aspects for telecommunication systems

Printed in Switzerland
Geneva, 2004