INTERNATIONAL TELECOMMUNICATION UNION

ITU-T

TELECOMMUNICATION
STANDARDIZATION SECTOR
OF ITU

H.235

(11/2000)

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

(Formerly CCITT Recommendation)

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

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 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 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 H.245 does not guarantee the interoperability between two entities' codecs.

This version of ITU-T H.235 supersedes 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.

Source

ITU-T Recommendation H.235 was revised by ITU-T Study Group 16 (2001-2004) and approved under the WTSA Resolution 1 procedure on 17 November 2000.

The first version of ITU-T H.235 was approved by the ITU-T Study Group 16 on 6 February 1998.

Keywords

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

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.

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 (2000) provides details on implementation with ITU-T H.323. This framework is expected to operate in conjunction with other H-series protocols that utilize ITU-T 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 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 H.323, ITU-T H.324 and ITU-T H.310 make use of the logical channel signalling procedures of ITU-T 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; all users of this Recommendation are therefore encouraged to investigate the possibility of applying the most recent edition of the Recommendations and other references listed below. A list of the currently valid ITU-T Recommendations is regularly published.

– ITU-T H.225.0 (2000), *Call signalling protocols and media stream packetization for packet-based multimedia communication systems*.

– ITU-T H.235 (1998), *Security and encryption for H-series (H.323 and other H.245-based) multimedia terminals*.

– ITU-T H.245 (2000), *Control protocol for multimedia communication*.

– ITU-T H.323 (2000), *Packet-based multimedia communications systems*.

– ITU-T H.323 Annex J (2000), *Security for H.323 Annex F*.

– ITU-T Q.931 (1998), *ISDN user-network interface layer 3 specification for basic call control*.

– ITU-T X.509 (2000) | ISO/IEC 9594-8:2001, *Information technology – Open Systems Interconnection – The Directory: Authentication framework.*

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

– ITU-T X.810 (1995) | ISO/IEC 10181-1:1996, *Information technology – Open Systems Interconnection – Security frameworks for open systems: Overview*.

– ITU-T 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 mechanisms – 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 FCD 15946-1, *Information technology – Security techniques – Cryptographic techniques based on elliptic curves – Part 1: General*.

– ISO/IEC FCD 15946-2, *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.*

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

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

– IETF RFC 2138 (1997), *Remote Authentication Dial In User Service (RADIUS)*.

– 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 Encryption Version 2.0*.

– IETF RFC 2459 (1999), *Internet X.509 Public Key Infrastructure Certificate and CRL Profile*.

3 Terms and definitions

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

3.2 authentication: The provision of assurance of the claimed identity of an entity (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 (X.810). In this Recommendation the term refers to "public key" certificates which are values that represent an owners 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 (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 nonrepudiation: 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 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 (X.810).

3.20 threat: A potential violation of security (X.800).

4 Symbols and abbreviations

This Recommendation uses the following abbreviations:

CRL	Certificate Revocation List
DSS	Digital Signature Standard
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
IPSEC	Internet Protocol Security
QOS	Quality of Service
RSA	Rivest, Shamir and Adleman (public key algorithm)
SDU	Service Data Unit
TLS	Transport Level Security

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 Recommendation 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**).

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 (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: **EncryptionUpdateRequest** and **EncryptionUpdate**.

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.

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 CONNECT ACKNOWLEDGE 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 logical 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 8.3/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 overhead 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.

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.

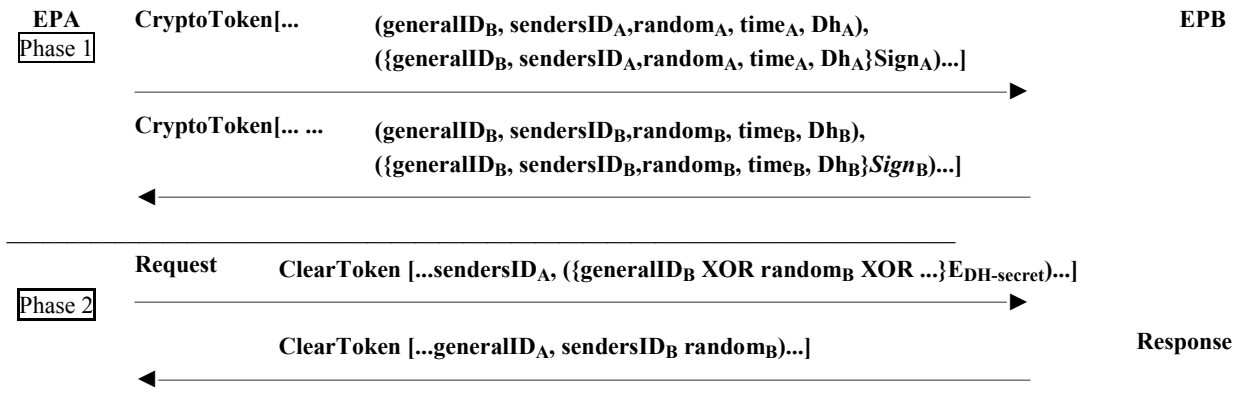
10.2 Diffie-Hellman with optional authentication

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, implementers 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.

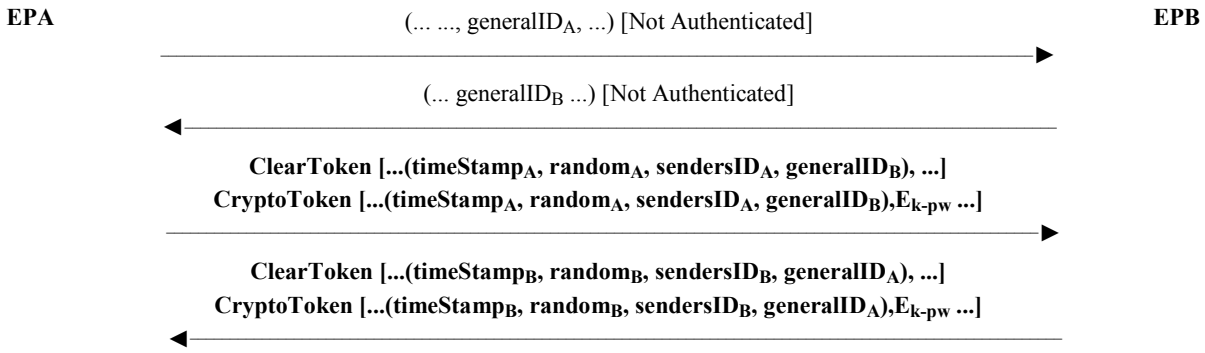
10.3.2 Password with symmetric encryption

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).



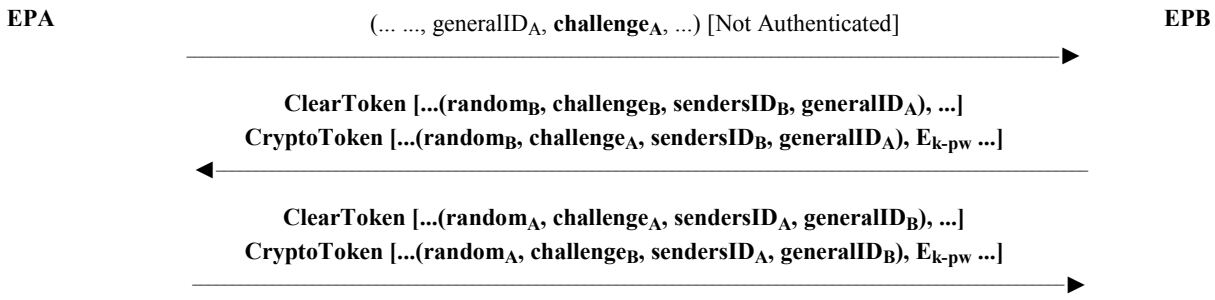
NOTE 1 – The return token from EPB is optional; if omitted, only one-way authentication is achieved.

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 and encryption function that are encrypted using the key "k" derived from the password "pw".

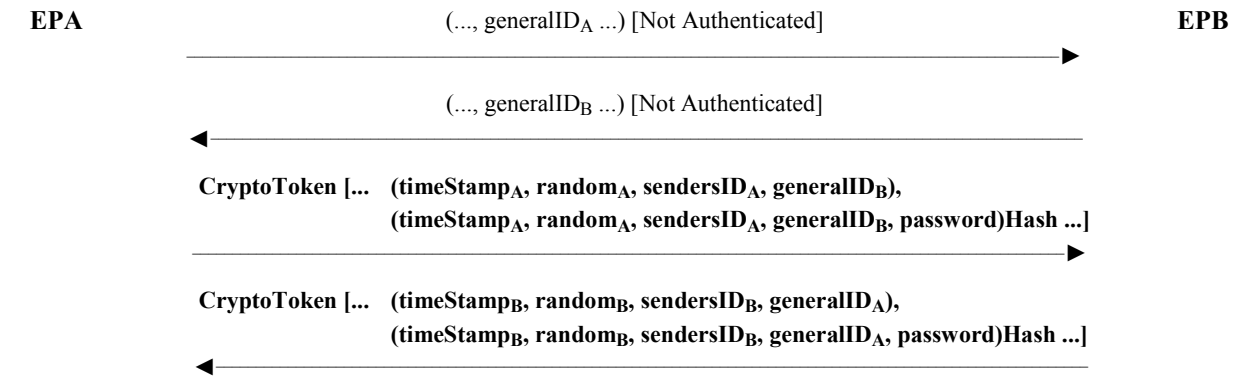
NOTE 3 – In the third message, EPA provides a a new **challenge$_A$** in plaintext in a separate **ClearToken**, that is identified through as same OID as the OID in the **CryptoToken**. EPA also returns the encrypted **challenge$_B$** as response; similarly for the second message and vice versa.

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.

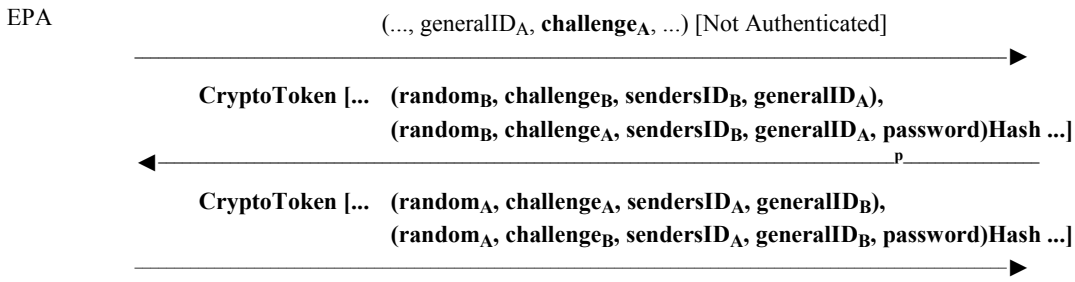


NOTE 1 – The return token from EPB is optional; if omitted, only one-way authentication is achieved.

NOTE 2 – **Hash** indicates a hashing function that operates on the contained values.

NOTE 3 – **random** is a monotonically increasing counter making multiple message 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. Implementers shall include the timestamp in the **hashedVals** and shall *not* include the password. (E.g. 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**.

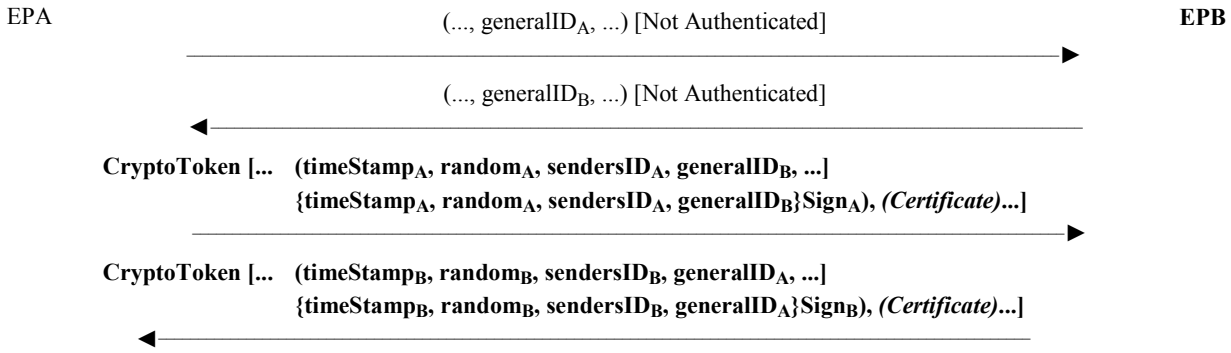
NOTE 3 – Implementations should ensure that user-entered passwords convey sufficient entropy. Passwords that are too short or that are susceptible against dictionary attacks should be rejected. Feeding the user-entered pass-phrase through a cryptographic hash function and using the output bits may be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

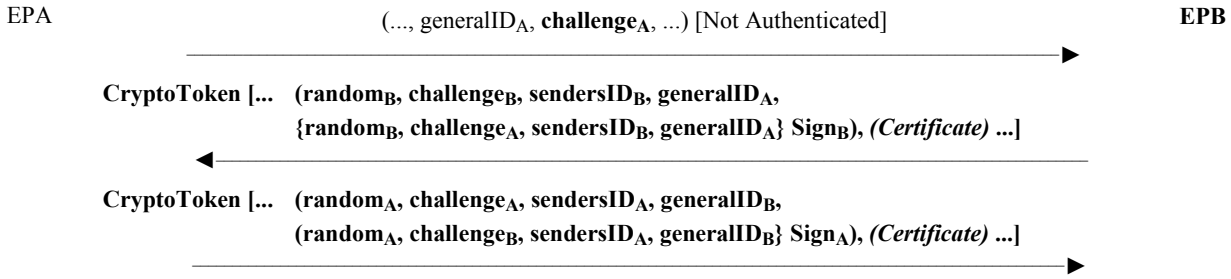
NOTE 1 – An optional certificate element may also be provided; these are illustrated in ***italics*** below.

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 message 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 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 can provide only sufficient security, when chosen randomly from a large space, convey sufficient entropy such that they are unpredictable and are changed periodically. Rules for setting up and maintaining passwords do not fall within the scope of this Recommendation.

A good practice 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 SHA-1 has function applied to the password string yields a 20-byte shared secret. As benefit, the hashed result does not only conceal the actual password but also defines a fixed length bit string format without really sacrifying 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 H.323 the RTP header (SDU) will change its payload type to indicate the switch to the new key.

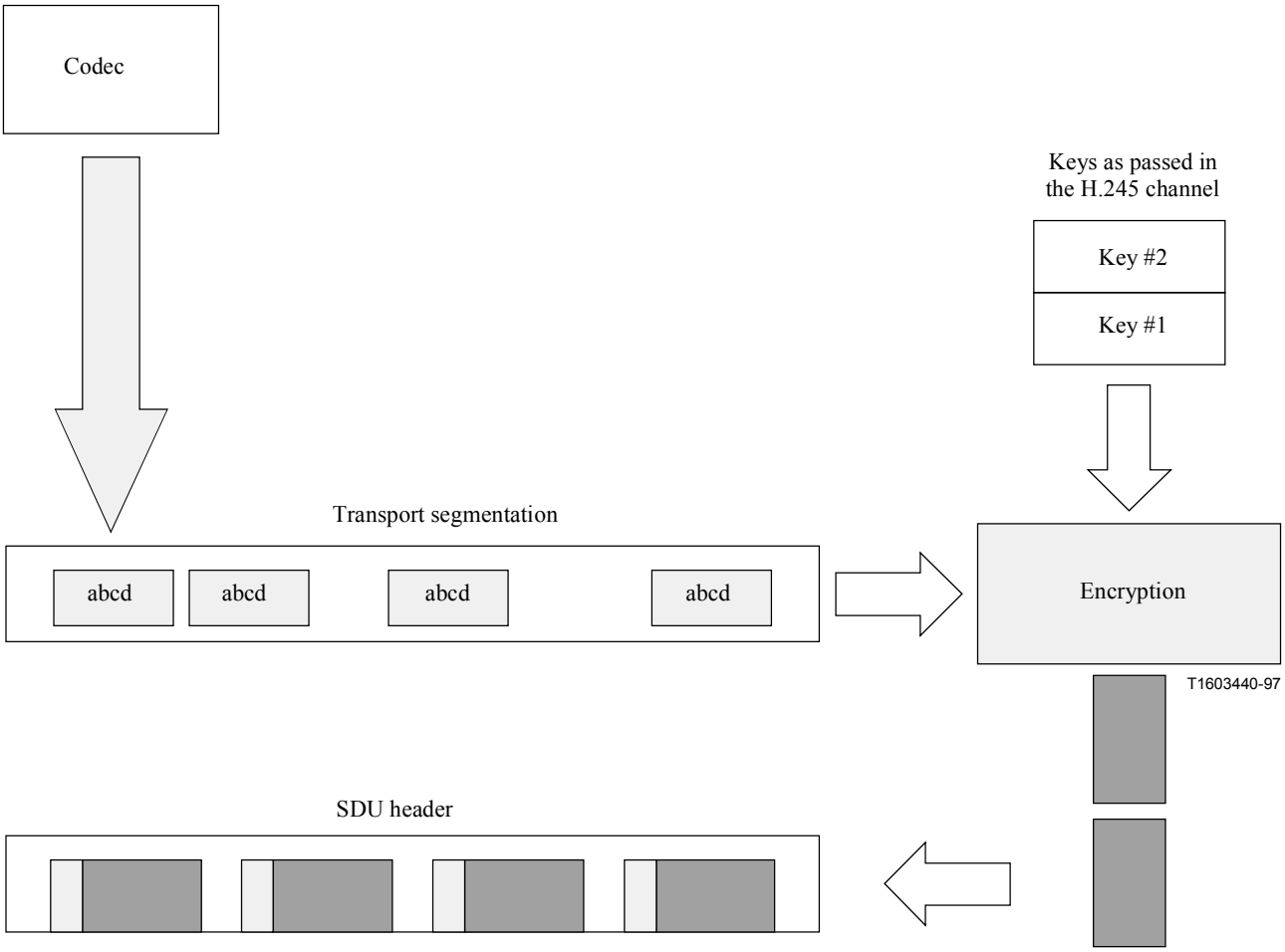


Figure 5/H.235 – Encryption of media

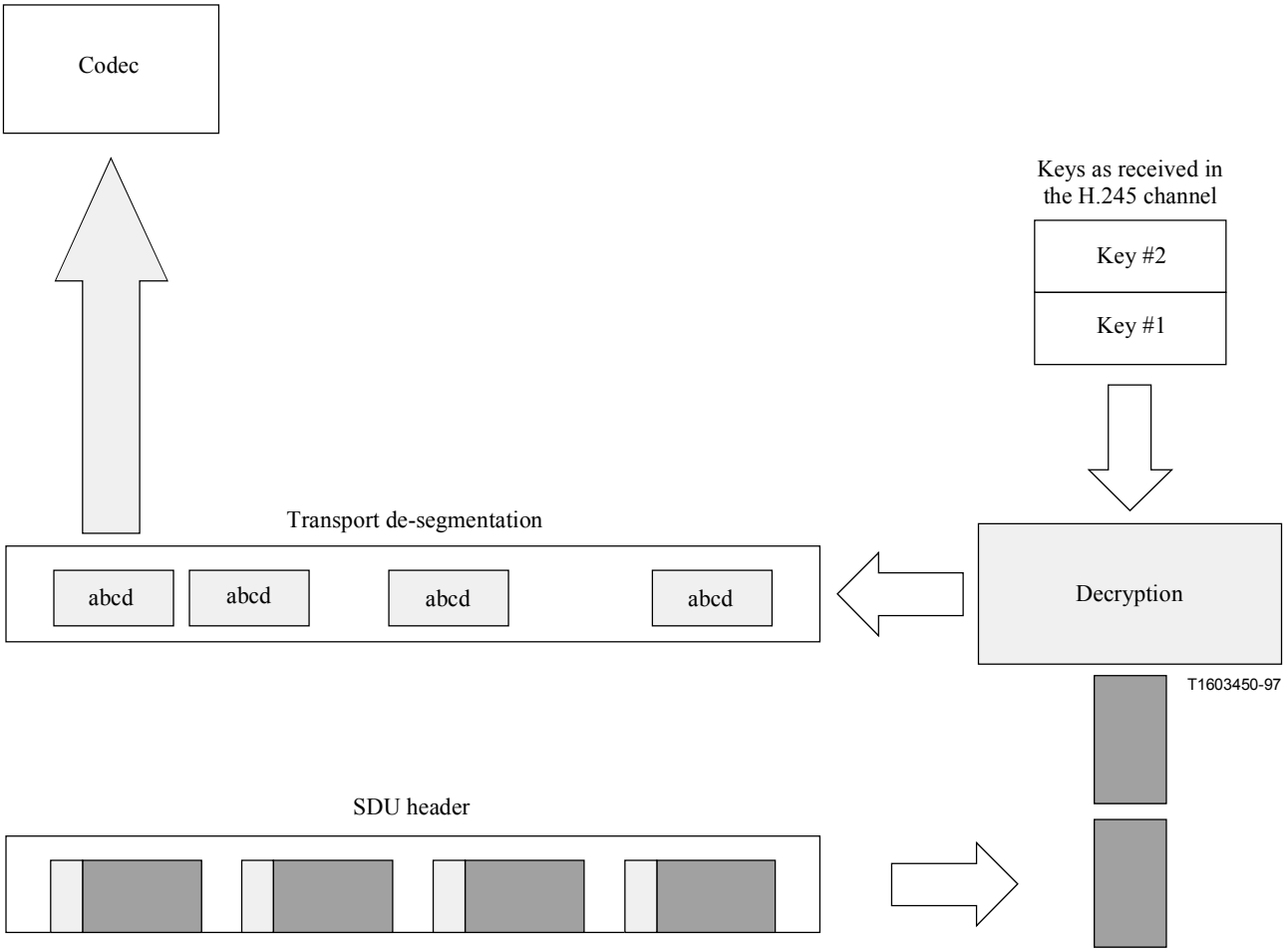


Figure 6/H.235 – Decryption of media

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 "S" 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 requires a different object identifier then. 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:

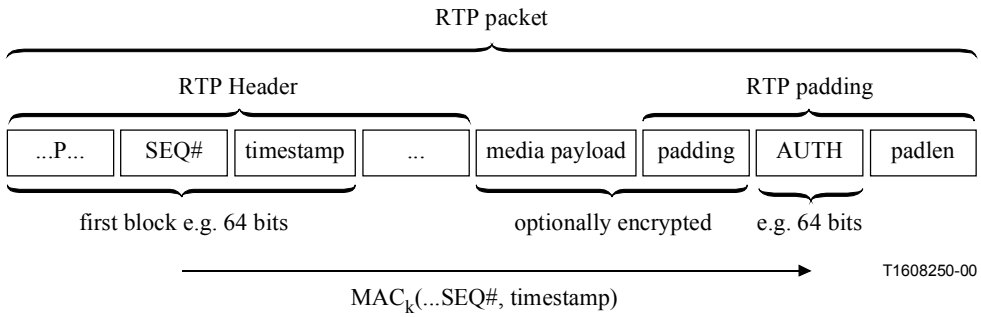


Figure 7/H.235 – RTP packet format for media anti-spamming

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 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.

   As key 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 mismatch, 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 likely indicates an attempted 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 verified the authenticity of the RTP packet, the payload is decrypted and then the RTP padding be discarded. The general procedure is according to case 1 above.

11.2.1 List of Object Identifiers

Table 1 lists all the referenced OIDs.

Table 1/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 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 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 CryptoSystems

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 curves 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 either one shall be present when Diffie-Hellman key exchange is applied.

Remark – Dnot 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
}
```

```
--      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 ECGDSASignature
} ( 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
}

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 "h235Key" field
{
        secureChannel               KeyMaterial,
        sharedSecret                ENCRYPTED {EncodedKeySyncMaterial},
        certProtectedKey            SIGNED { EncodedKeySignedMaterial },
        ...
}

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)


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 H.323.

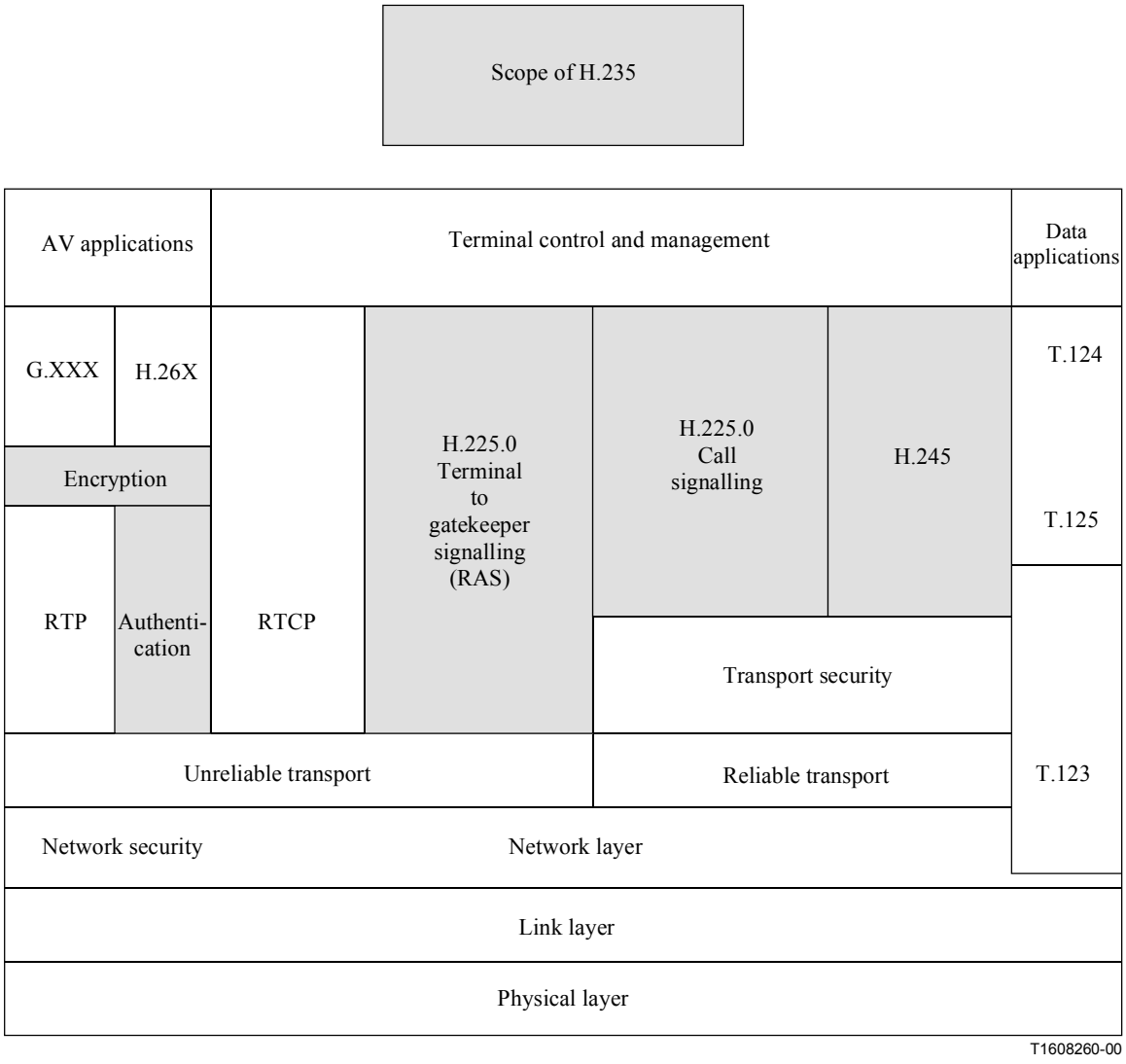


Figure B.1/H.235 – Overview

For ITU-T 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 **ReleaseComplete** 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 **ReleaseComplete** with a reason code of ***UndefinedReason***.

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 (including the payload header) shall not be encrypted. Synchronization of new keys and encrypted text is based upon dynamic payload type.

Initial encryption key is presented by the master in conjunction with the dynamic payload number (via **EncryptionSync** in ITU-T 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. 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 that the specific values do not matter, as long as they change for every new key that is distributed.

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

---

[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.

Recommendation. All modes must allow for lost (or out-of-sequence) packets, in addition to padding packets to an appropriate multiple of octets.

Deciphering the stream must be stateless due to the fact that packets may be lost; each packet decipherable on its own merits. Two requirements of block algorithm mode shall operate as follows:

a) *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. In all cases, an IV shall be constructed from the first B (where B is the block size) octets of: (Seq# + Timestamp). This pattern should be repeated until enough octets have been generated. 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) *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 ECB and CBC; Zero pad for CFB and OFB.

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

[RTP, section 5.1] describes a method of padding in which the payload is padded to a multiple of blocks, the last octet set with 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.

Integrity and replay protection of the RTP stream is for further study.

Application of cryptographic techniques to RTCP elements is 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, 10.2.2, 10.2.3 and 10.2.4. 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 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 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 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**.

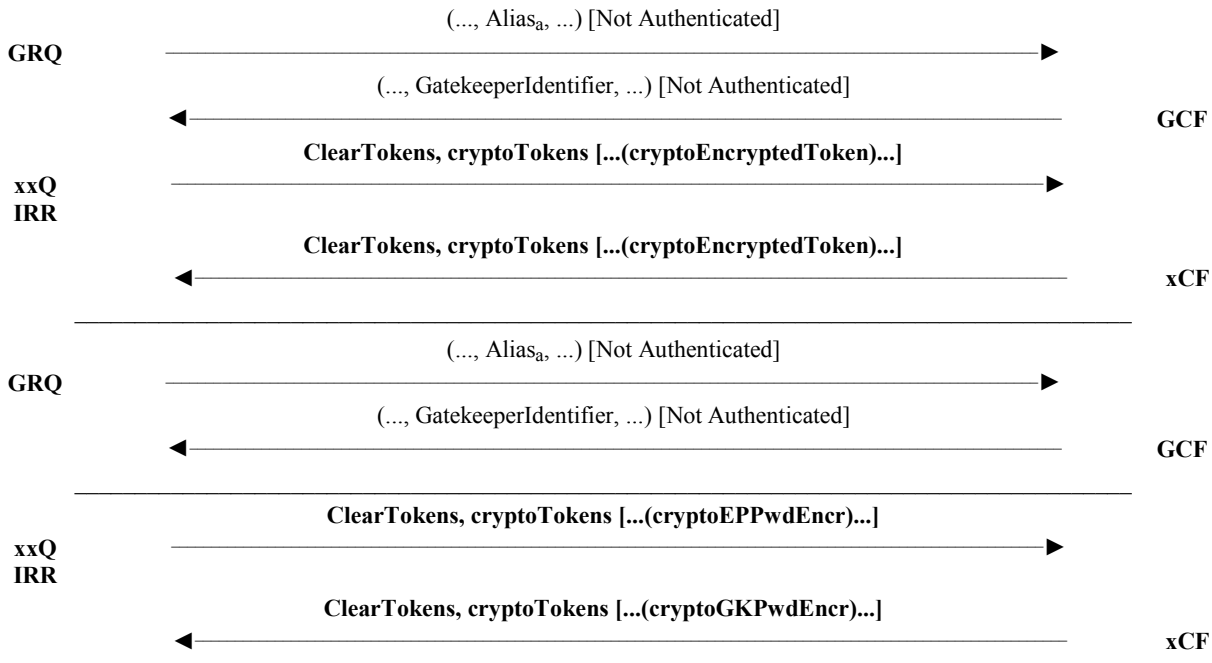


Figure B.2/H.235 – Password with symmetric encryption

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**.

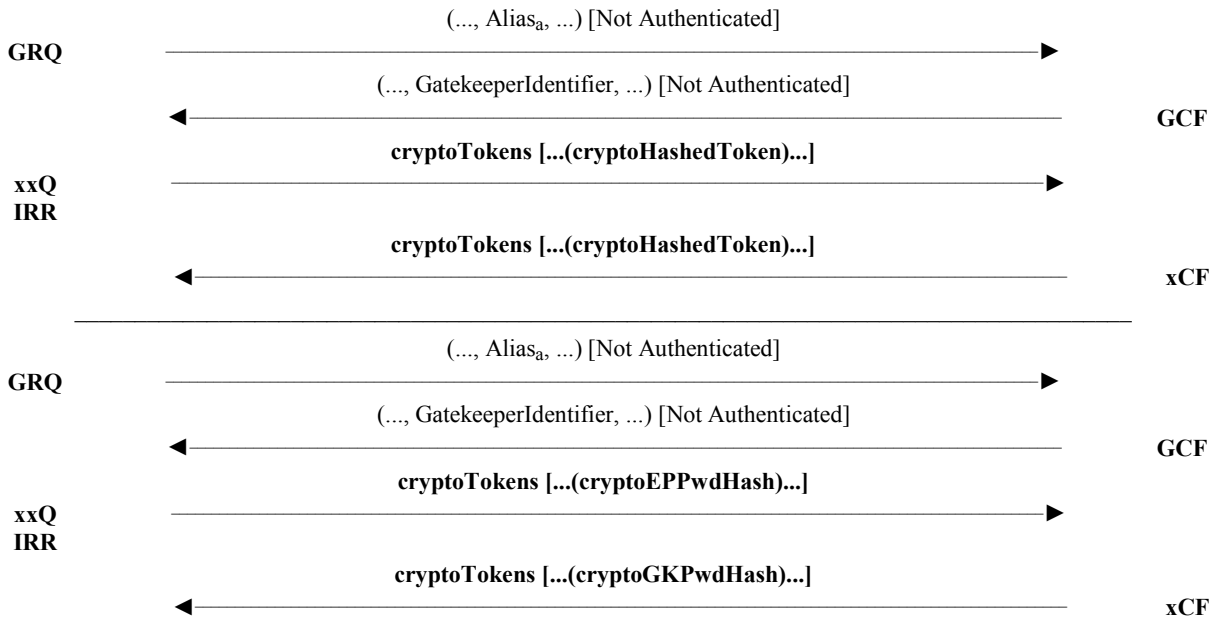


Figure B.3/H.235 – Password with hashing

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**.

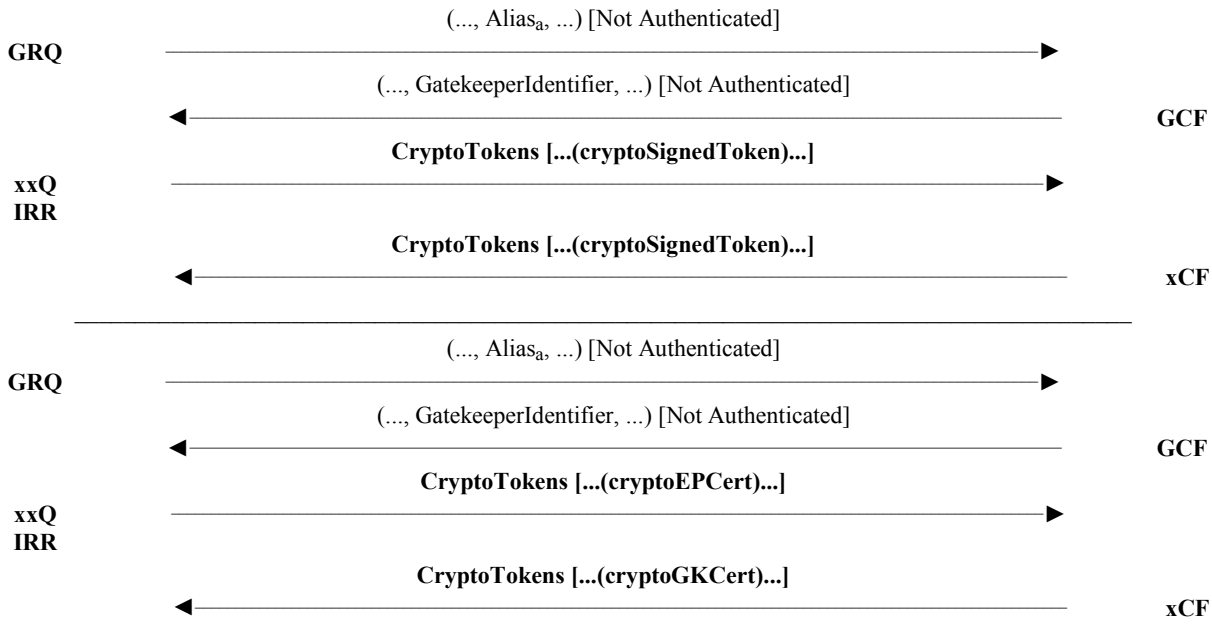


Figure B.4/H.235 – Certificate-based with signatures

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.

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 H.235, available ETSI and IMTC security profiles. The security profiles select appropriate security features from ITU-T 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").

While the message integrity service always provides also 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, password, session key and shared secret all have in common 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 unused in the following sense: **random** is actually a monotonically increasing number starting at any value and is being increased for 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 this is the GK or endpoint identifier[3], for H.225.0 call signalling messages this is the called endpoint identifier. The **sendersID** shall be set to the identification string of the sender.

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.

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 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

BES Back-end Service
CBC Cipher Block Chaining
DES Data Encryption Standard
DH Diffie-Hellman
ECB Electronic Code Book
EP Endpoint

---

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

[3] Which one depends on the direction EP to GK or vice versa.

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
MAC Message Authentication Code
MD5 Message Digest 5
OID Object Identifier
PFS Perfect Forward Secrecy
RAS Registration, Admission and Status
RSA Rivest, Shamir and Adleman
RTP Real-Time 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

DES [FIPS-46-2] US National Bureau 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.

[FIPS-74] US National Bureau 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.

[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] *Information Technology – Security techniques – Hash-functions – Part 3: Dedicated hash-functions*, 1998.

[H.225.0] ITU-T H.225.0 Version 2, *Call Signalling Protocols and Media Stream Packetization for Packet Based Multimedia Communications Systems*, 1998.

[H.235v1] ITU-T H.235 Version 1, *Security and encryption for H-series (H.323 and other H.245-based) multimedia terminals*, 1998.

[H.235v2] ITU-T H.235 Version 2, *Security and encryption for H-series (H.323 and other H.245-based) multimedia terminals*, 2000.

[H.245] ITU-T H.245 Version 7, *Control protocol for multimedia communication*, 2000.

[H.323] ITU-T H.323 Version 4, *Packet Based Multimedia Communication Systems*, 2000.

[H.323 Annex F] ITU-T H.323 Annex F, *Simple Endpoint Types*, 1999.

[RFC 2268] RIVEST (R.): *A Description of the RC2® Encryption Algorithm*, RFC 2268, March 1998.

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[4] 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.
- Applicable to various deployment scenarios such as in closed groups and for scaleable environments and in multipoint conferences.

---

[4] 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.

Table D.1 summarizes all the procedures defined in this annex by the security profiles to deal with different security requirements. The table includes the baseline security profile (vertical shading – blue in the electronic copy) and the voice encryption security profile (horizontal shading – green in the electronic copy).

Table D.1/H.235 – Summary of Annex D Security Profiles

Security services	Call functions						
	RAS	H.225.0		H.245[a)]	RTP		
Authentication	Password HMAC-SHA1-96	Password HMAC-SHA1-96		Password HMAC-SHA1-96			
Non-Repudiation							
Integrity	Password HMAC-SHA1-96	Password HMAC-SHA1-96		Password HMAC-SHA1-96			
Confidentiality					56-bit DES	56-bit RC2-compatible	168-bit triple-DES
Access Control							
Key Management	Subscription-based password assignment	Subscription-based password assignment	Authenti-cated Diffie-Hellman key-exchange	Integrated H.235 session key management (key distribution, key update using 56-bit DES/56-bit RC2-compatible/168-bit triple-DES)			

a) Tunnelled H.245 or embedded H.245 inside H.225.0 fast connect.

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 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 RC2-compatible, DES or triple-DES based on the business model and exportability requirement. Some environments that are offering already 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 168-bit Triple-DES while 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 happen by means that are not part 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 for RAS, 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	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. Three encryption algorithms are offered: the suggested scheme is encryption using RC2-compatible, DES or triple-DES based on the business model and exportability requirement. Some environments that are offering already 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 168-bit triple-DES while 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

Security services	Call functions					
	RAS	H.225.0	H.245	RTP		
Authentication						
Non-Repudiation						
Confidentiality				56-bit DES	56-bit RC2-compatible	168-bit triple-DES
Access Control						
Key Management		Authenticated Diffie-Hellman key-exchange	Integrated H.235 session key management (key distribution, key update)			

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 could authenticate when correctly applying a shared secret key. Both security services are provided by the same security mechanism.

When using symmetric key techniques, the security services authentication/integrity apply only 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

We describe the following procedure 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, Q931 and H.245 messages (see D.6.3.2).

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 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**.

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:

- A 12-byte (96-bit) hash value is used with the HMAC-SHA1-96 algorithms for generating the 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 RAS/H.225.0 message.
    - **hashedVals** containing the **ClearToken** field used with the following fields:
      - **tokenOID** set to "T", indicating that **ClearToken** is being used for message authentication/integrity.
      - **timeStamp** contains the timestamp.
      - **random** contains a monotonically increasing sequence number. This number allows to make two messages with the same timestamp (within the clock resolution) unique.
      - **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** and **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 need to be sent; instead **halfkey**, **modsize** and **generator** may be set to the binary representation of 0 for simplicity.
  - **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 RAS/H.225.0 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 PDU** 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 re-computed 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 10118-3] shall be used.

NOTE 5 – When the combined authentication and integrity is being used, then 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 **algorithmOID** within the hashed **EncodedGeneralToken** (detecting presence of "U").

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").

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

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 yielded a correct hash value comparison, then the authentication failed and the message has been altered (accidentally or intentionally) during transit.

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, that this scenario does not provide true end-to-end security between endpoints; all security depends on the trusted intermediate gatekeepers.

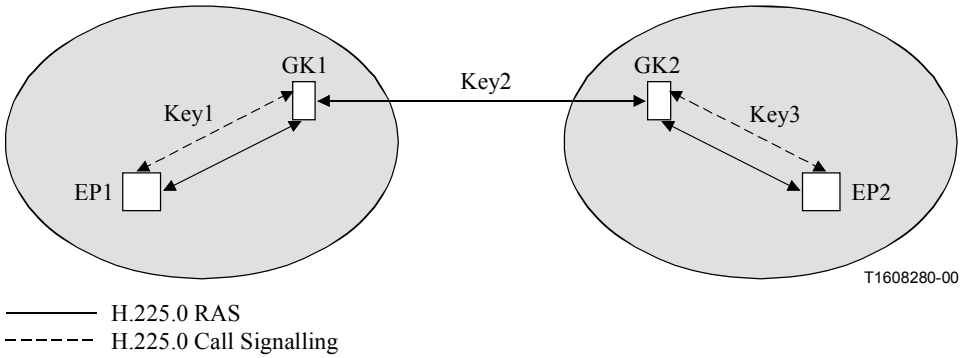


Figure D.1/H.235 – Illustrating Procedure I usage in a GK-GK scenario with both EPs in GK-routed zones

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.

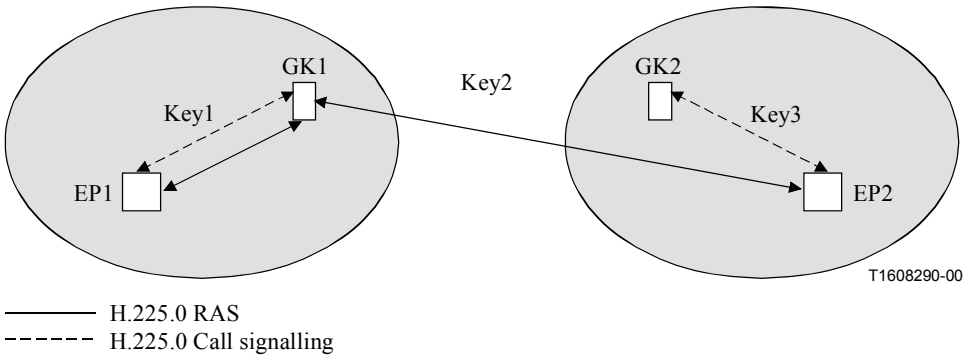


Figure D.2/H.235 – Illustrating Procedure I usage in a mixed scenario with EP1 in a GK-routed zone and EP2 in a direct-routed zone

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 that this scenario provides true end-to-end security among endpoints without relying on trusted intermediate nodes.

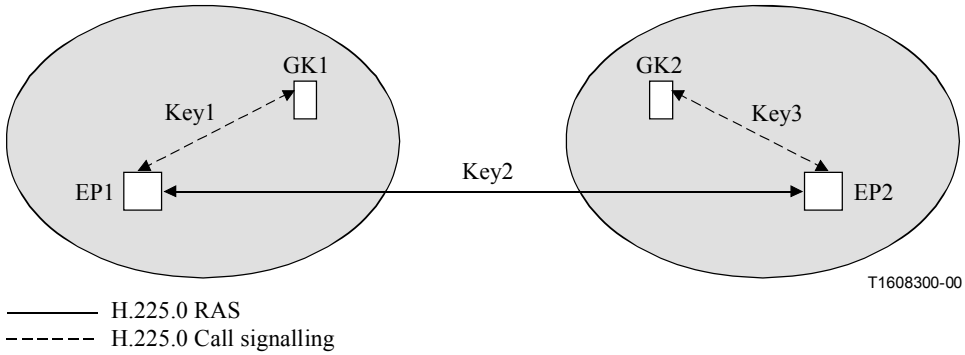


Figure D.3/H.235 – Illustrating Procedure I usage in a scenario with both EPs in zones using a direct-routed GK

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 10.3.2. 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 – The scenarios shown in the Figures 1 to 3 do not scale well in case 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.

---

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

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 12-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:

- 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 **Setup** 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 12-byte key *Key1*. The authenticator is computed according to the hash method chosen (A) taking into account the entire H.225.0 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 **Setup** 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 **H323-UU-PDU** 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 [H.235 (2000)] in mind, it is also possible to apply the security profiles for H.235 version 1 [H.235 (1998)] 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 D.11).

H.235 version 1 [H.235 (1998)] implementations

- do not set or evaluate the **sendersID** in the **ClearToken**.
- cannot use backend 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 the 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 RC2-compatible, DES and triple-DES. Note that 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 a).
- Padding if needed is to occur as described in Annex B.

The audio payload[7] is encrypted using the negotiated encryption algorithm ("X", "Y" or "Z") operating in CBC mode according to the procedures described in clause 11 and in Annex B and the ciphertext padding methods of I.1.

D.7.1 Key management

- During the **Setup**-to-**Connect** sequence, a Diffie-Hellman (DH) exchange is 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 4. For more details, please refer to [RFC 2412, Appendix E2]. Note that since the H.225.0 messages are authenticated (as described earlier by Procedure I), the DH exchange is an authenticated one.

---

[7] Without the payload header.

- During FastStart, the caller (source of the **Setup**) presents both its DH token, and the supported FastStart structures. Both H235Cap and nonH235Cap channels should be offered. The **mediaWaitForConnect** should be set to TRUE[8].
- During FastStart, the caller (source of the **Connect**) presents its DH token and the accepted FastStart structures. The session key is included in the **encyptionSync** field. The session key is itself encrypted with the DH shared secret in the same manner as the non-FastStart operation.
- During the H.245 Cap exchange, endpoints present **H235Capability** entries for the codecs that they support. Each codec is associated with a separate H.235 capability. These capabilities should indicate support for 56-bit RC2-compatible (OID – "X"), should indicate support for 56-bit DES (OID – "Y") and may indicate support for 168-bit triple-DES (OID – "Z").

  The negotiated encryption algorithms and their modes of operation for the media stream encryption will be used also for secure distribution of the session key. The encryption algorithm for encryption of the media key shall operate in the same chaining mode as the media encryption algorithm.
- **OpenLogicalChannel(Ack)** responses are issued with the (master) created session key included in the **encryptionSync** field. The session key is itself encrypted with the DH shared secret in a manner described below[9].
- **OpenLogicalChannel** conveys both **forwardLogicalChannelParameters** and **reverseLogicalChannelParameters** with **dataType** providing **h235Media** with **encryptionAuthenticationAndIntegrity** where in the **encryptionCapability** at most one **MediaEncryptionAlgorithm** shall be present.

NOTE – 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 of the security policy.

- 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**. The **KeySyncMaterial** is encrypted using:
  - 56 bits of the shared secret, starting with the least significant bits from the Diffie-Hellman secret for OID "X" or OID "Y".
  - all the bits of the shared secret for OID "Z" starting with the least significant bits from the DH secret.

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**.

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" or OID "Y" and the 168 least significant bits of the Diffie-Hellman secret for OID "Z".

---

[8] Note that 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. (For the security relationship's purpose, the callee is the *a priori* master.)

[9] Note that 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.

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

OID	D-H group description
"X" (RC2-compatible), "Y" (DES)	Mod-P, any suitable 512-bit prime
"Z" (triple-DES)	Mod-P, 1024-bit prime Prime = $2^{1024} - 2^{960} - 1 + 2^{64} \times \{ [2^{894}$ pi] + 129093 } = (179769313486231590770839156793787453197860296048756011706444 423684197180216158519368947833795864925541502180565485980503 646440548199239100050792877003355816639229553136239076508735 759914822574862575007425302077447712589550957937778424442426 617334727629299387668709205606050270810842907692932019128194 467627007)$_{10}$ Generator[a] = 2
[a] The generator is used to generate the DH token.	

D.7.2 Key update and synchronization

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 10.3). 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** of the **miscellaneousCommand** message. On the other hand the slave can request a new session key from the master to change by using the **encryptionUpdateRequest** of the **miscellaneousCommand** message.

The **MiscellaneousCommand** message contains the **encryptionUpdate** 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" for the 56-bit RC2-compatible, set to "Y" for 56-bit DES or set to "Z" for 168-bit Triple-DES. This is the encryption algorithm by which the media session key is being encrypted.

NOTE 1 – The session key encryption algorithm is the same as the negotiated media encryption algorithm.

- **paramS**: set to the initial value. **iv8** holds a random 64-bit block bit pattern that the initiator generates. This field is not used for the CBC mode and is set to NULL.
- **encryptedData**: set to the result of the encrypted **KeySynchMaterial**.

As part of the **KeySyncMaterial**:

- **generalID**: identifier of the source distributing the key.
- **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. The master shall generate a new session key that meets at least the following security criteria: 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 2 – 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. Thus, key update and synchronization can only be used in the signature security profile.

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].

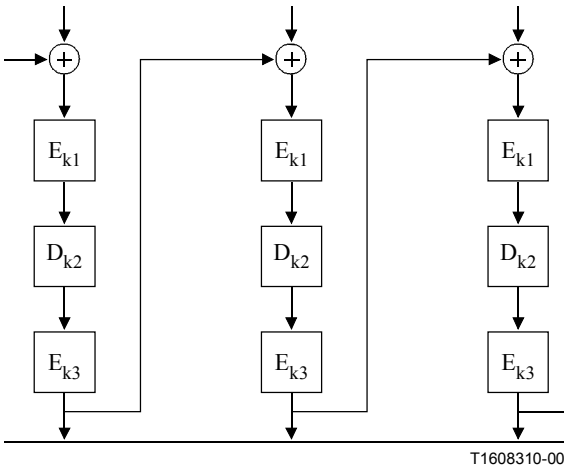


Figure D.4/H.235 – Triple-DES encryption in outer CBC mode

D.8 Lawful Interception

For further study (see [LI]).

D.9 List of secured signalling messages

This clause provides 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	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.235 (1998)] and for H.235v2 [H.235 (2000)].

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 RAS/H.225.0 message (authentication and integrity).
"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 Procedure I for the ClearToken-tokenOID, indicating that the ClearToken is being used for message authentication and integrity.
"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.
"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.
"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.
"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 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[10] 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.

---

[10] "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.

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 also 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, 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 **CryptoSignedToken** element. This information is re-computed 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.

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[11].

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 (orange in the electronic copy), the authentication-only security profile results. An option within the signature security profile is to choose between RSA-SHA-1 or RSA-MD5 digital signatures. The voice encryption security profile of Annex D (see 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**[a)]	**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		
a) 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).

[11] 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.

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 center 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 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[12] 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 2459).

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:

---

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

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 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 message indicating security failure by setting the reject reason to **securityDenial**.

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 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 RAS/H.225.0 message (see E.9);
      – "B", indicating that the authentication/ integrity computation includes only a subset of fields (see 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 time stamp;
        - **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** and **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 need to be sent; instead **halfkey**, **modsize** and **generator** may be set to the binary representation of 0 for simplicity.

  - **certificate** containing the sender's digital certificate (see 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 E.9) or certain critical fields (if **tokenOID** is "B", see E.8) of the RAS/H.225.0 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 **H323-UU-PDU** message with all fields according to the procedure described in E.9. In case, **tokenOID** "B" is used, authentication-only of the **CryptoToken** is achieved when applying the Procedure III (see 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 to associate 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 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 re-computed 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.

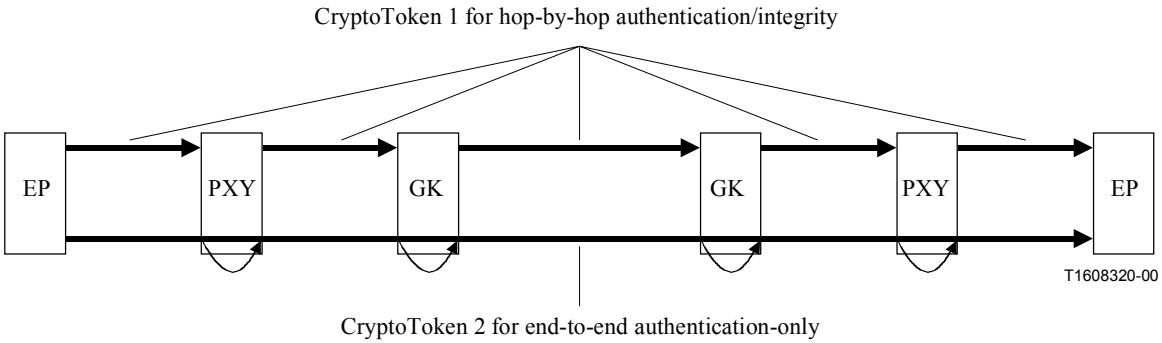


Figure E.1/H.235 – Simultaneous use of hop-by-hop security and end-to-end authentication

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 E.9);
    - "B", indicating that the authentication computation includes only a subset of fields (see E.8) in the RAS/H.225.0 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[13] on an end-to-end basis;
    - **random** contains a monotonically increasing sequence number;
    - **timeStamp** optionally for enhanced security only when the terminating end entities are time synchronized;

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

- **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 E.12);
- **dhkey**, used to pass the Diffie-Hellman parameters as specified in this Recommendation during **Setup** and **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 need to be sent; instead **halfkey**, **modsize** and **generator** may be set to the binary representation of 0 for simplicity.

- **token** with the fields:
  - **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 RAS/H.225.0 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 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 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 time stamp.
- **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 sub-fields are only used during **Setup** and **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.
3) Locate[14] 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 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 sub-string 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 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 yielded a correct signature value comparison, then the authentication failed and the message has been altered (accidentally or intentionally) during transit or for some other reason.

[14] 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 2459). 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 how the recipient can access the sender's certificate:

- The certificate is included in the message exchange as described by Procedures II and III.
- 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 of this recommendation (e.g. LDAP directory lookup).

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.

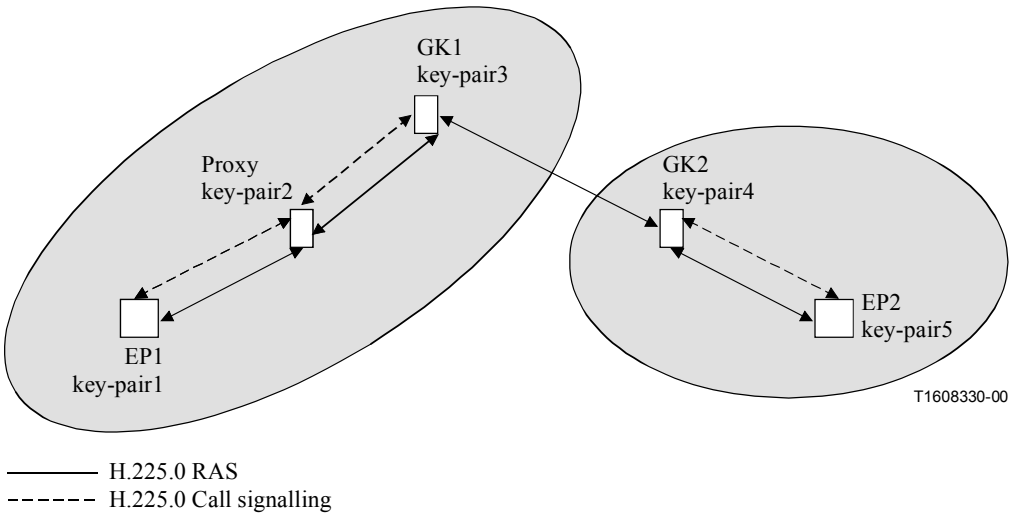


Figure E.2 /H.235 – Illustrating public-key usage in a GK-GK routed model

The H.323 proxy acts in a dual behaviour: On 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 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 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 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 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 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.235 (2000)] in mind, it is also possible to apply the security profiles for H.235 version 1 [H.235 (1998)] 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 E.18).

H.235 version 1 [H.235 (1998)] 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 the 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	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.235 (1998)] and for H.235v2 [H.235 (2000)].

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 RAS/H.225.0 message (authentication and integrity).
"B"	{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.
"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 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 as algorithm OID indicating use of SHA1 RSA digital signature.

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.

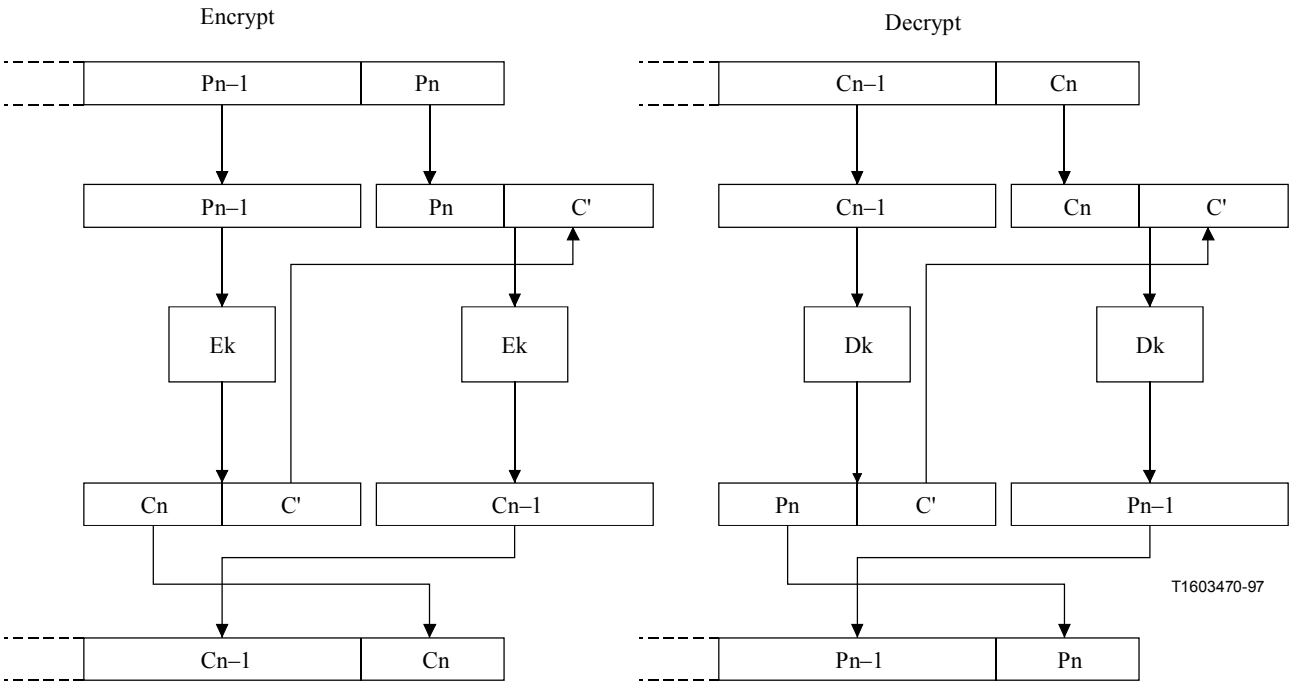


Figure I.1/H.235 – Ciphertext stealing in ECB mode

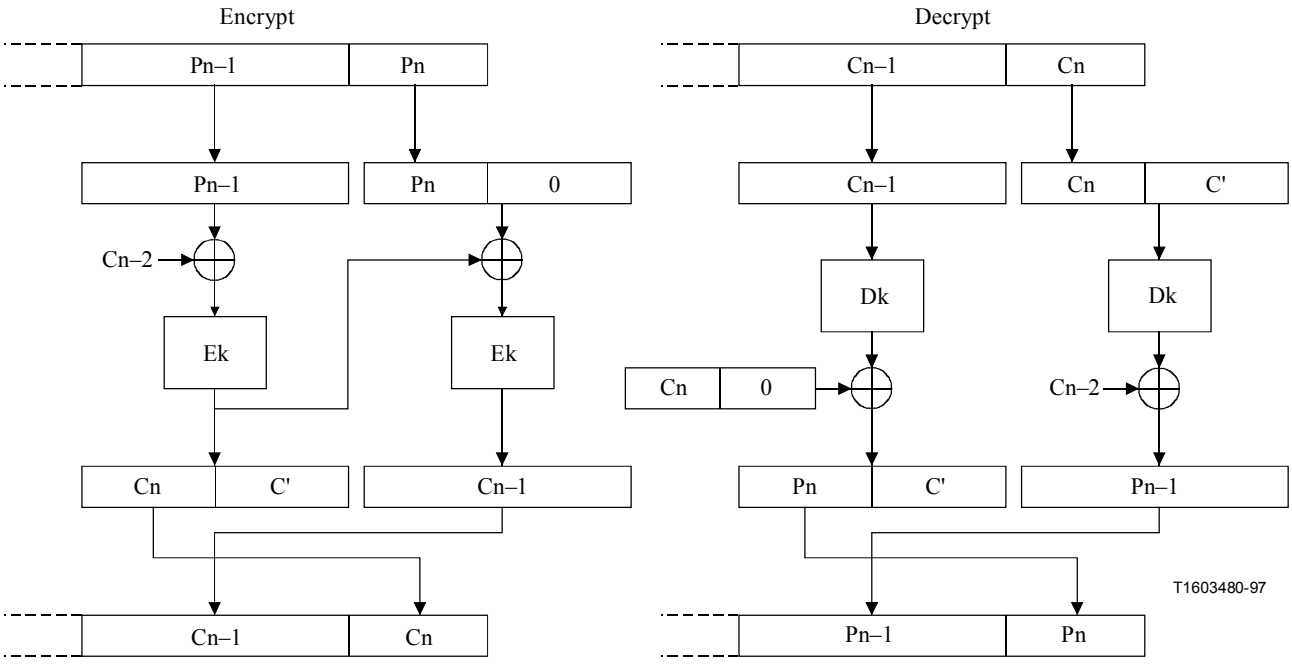


Figure I.2/H.235 – Ciphertext stealing in CBC mode

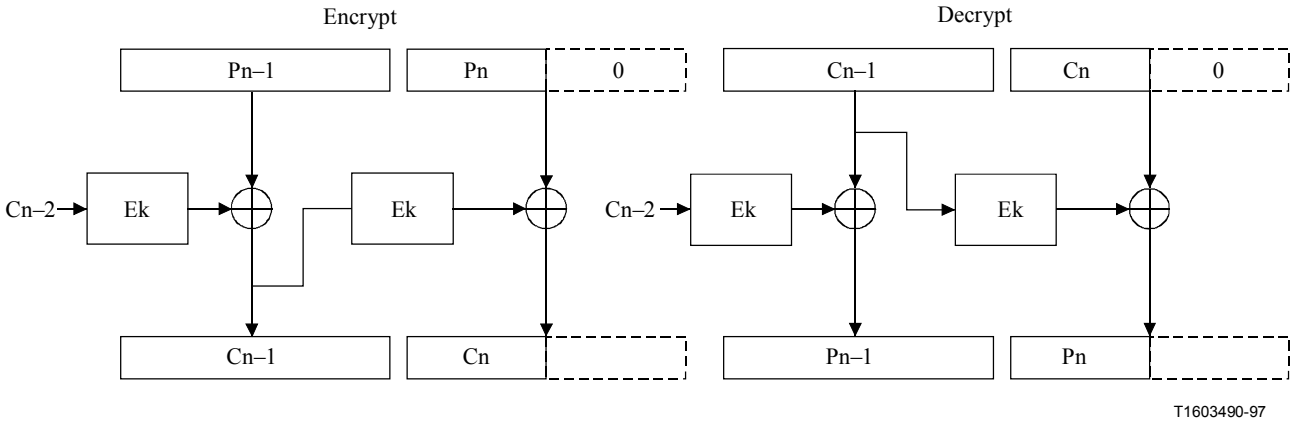
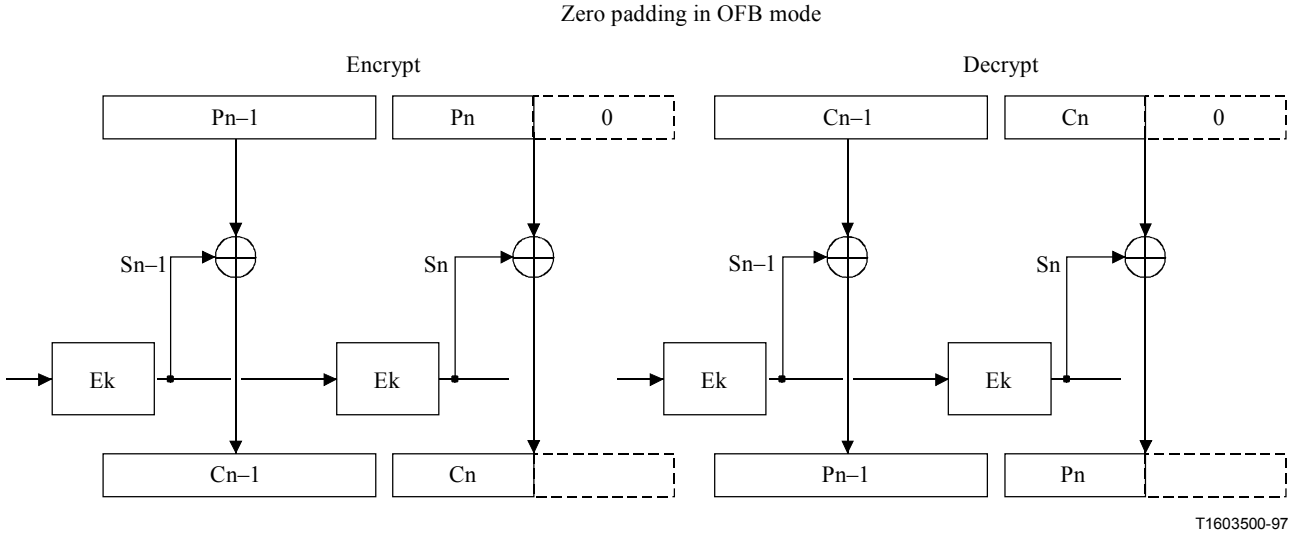


Figure I.3/H.235 – Zero padding in CFB mode



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

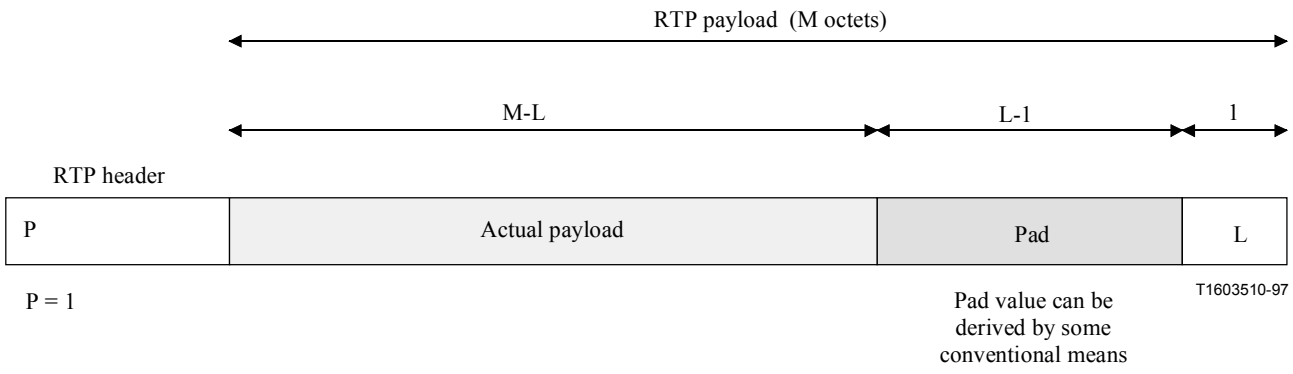


Figure I.5/H.235 – Padding as prescribed by RTP

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 next 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 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.

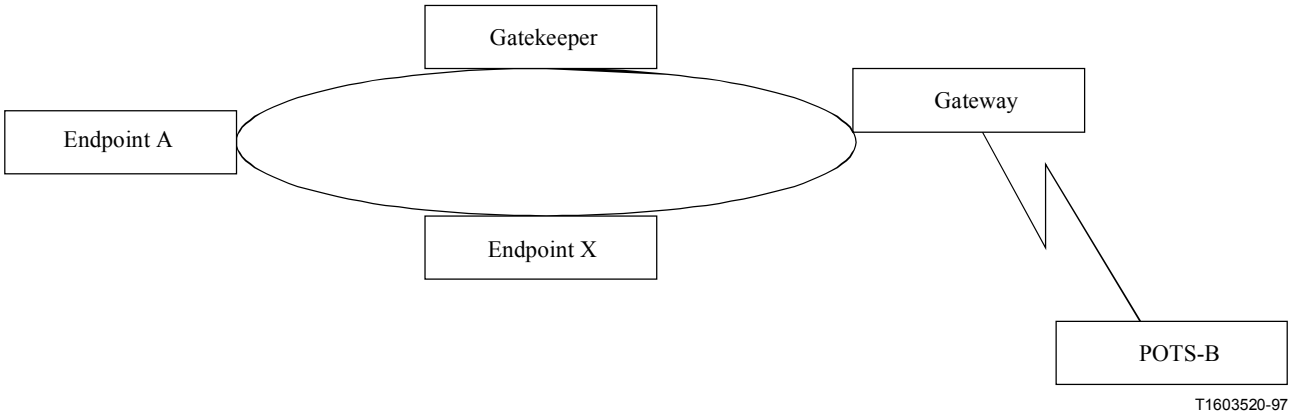


Figure I.6/H.235 – Tokens

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**. 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 that the **GRQ** message may be unicast to the gatekeeper.

I.4.5 IPSEC

In general, IPSEC **[13/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 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 in the scope of H.323 how exactly the BES communicates with the GK. Several methods and protocols could be applicable: RADIUS (see RFC 2138) is considered as one of the most important ones, which is widely deployed by service providers.

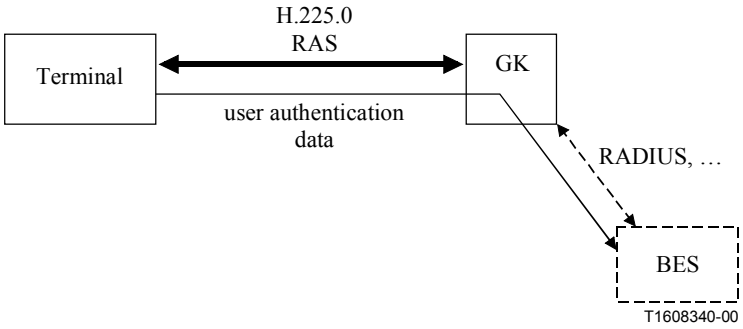


Figure I.7/H.235 – Scenario with Back-end Server

A GK offering BES support should support at least the following two modes:

1) **default mode**, where 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 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, pre-paid 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**, where BES and the terminal user share a common secret and the GK has not to 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 1}	indicates a RADIUS challenge in the ClearToken
"L"	{itu-t (0) recommendation (0) h (8) 235 version (0) 2 2}	indicates a RADIUS response (conveyed in the challenge field) in the ClearToken
"M"	{itu-t (0) recommendation (0) h (8) 235 version (0) 2 3}	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.

[PKCS]

– PKCS #1 v2.0: RSA Cryptography Standard; RSA Laboratories; October 1, 1998; http://www.rsa.com/rsalabs/pubs/PKCS/index.html.

– 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 1889, *Internet Engineering Task Force*, 1996.

[Schneier]

– SCHNEIER (B.): Applied Cryptography: Protocols, Algorithms, and Source Code in C, 2nd Edition, John Wiley & Sons, Inc., 1995.

[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	Transmission of television, sound programme and other multimedia signals
Series K	Protection against interference
Series L	Construction, installation and protection of cables and other elements of outside plant
Series M	TMN and network maintenance: international transmission systems, telephone circuits, telegraphy, facsimile and leased circuits
Series N	Maintenance: international sound programme and television transmission circuits
Series O	Specifications of measuring equipment
Series P	Telephone transmission quality, telephone installations, local line networks
Series Q	Switching and signalling
Series R	Telegraph transmission
Series S	Telegraph services terminal equipment
Series T	Terminals for telematic services
Series U	Telegraph switching
Series V	Data communication over the telephone network
Series X	Data networks and open system communications
Series Y	Global information infrastructure and Internet protocol aspects
Series Z	Languages and general software aspects for telecommunication systems

Geneva, 2001