International Telecommunication Union

ITU-T

TELECOMMUNICATION
STANDARDIZATION SECTOR
OF ITU

H.235.6

(01/2014)

SERIES H: AUDIOVISUAL AND MULTIMEDIA SYSTEMS

Infrastructure of audiovisual services – Systems aspects

H.323 security: Encryption profile with native ITU-T H.235/H.245 key management

Recommendation ITU-T H.235.6

International Telecommunication Union

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.349
Directory services architecture for audiovisual and multimedia services	H.350–H.359
Quality of service architecture for audiovisual and multimedia services	H.360–H.369
Supplementary services for multimedia	H.450–H.499
MOBILITY AND COLLABORATION PROCEDURES	
Overview of Mobility and Collaboration, definitions, protocols and procedures	H.500–H.509
Mobility for H-Series multimedia systems and services	H.510–H.519
Mobile multimedia collaboration applications and services	H.520–H.529
Security for mobile multimedia systems and services	H.530–H.539
Security for mobile multimedia collaboration applications and services	H.540–H.549
Mobility interworking procedures	H.550–H.559
Mobile multimedia collaboration inter-working procedures	H.560–H.569
BROADBAND, TRIPLE-PLAY AND ADVANCED MULTIMEDIA SERVICES	
Broadband multimedia services over VDSL	H.610–H.619
Advanced multimedia services and applications	H.620–H.629
Ubiquitous sensor network applications and Internet of Things	H.640–H.649
IPTV MULTIMEDIA SERVICES AND APPLICATIONS FOR IPTV	
General aspects	H.700–H.719
IPTV terminal devices	H.720–H.729
IPTV middleware	H.730–H.739
IPTV application event handling	H.740–H.749
IPTV metadata	H.750–H.759
IPTV multimedia application frameworks	H.760–H.769
IPTV service discovery up to consumption	H.770–H.779
Digital Signage	H.780–H.789
E-HEALTH MULTIMEDIA SERVICES AND APPLICATIONS	
Interoperability compliance testing of personal health systems (HRN, PAN, LAN and WAN)	H.820–H.849
Multimedia e-health data exchange services	H.860–H.869

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

Recommendation ITU-T H.235.6

H.323 security: Encryption profile with native ITU-T H.235/H.245 key management

Summary

Recommendation ITU-T H.235.6 holds the security procedures for the encryption profile (formerly in Annex D of Recommendation ITU-T H.235 version 1) including the accompanying native ITU-T H.235/H.245 key management.

This revision introduces support for key lengths larger than 2048 bits. In previous versions, bit sizes of 3072 and 4096 were defined. However, the field size defined in Recommendation ITU-T H.235.0 was limited to 2048 bits.

Version 4 of Recommendation ITU-T H.235 broke up Recommendation ITU-T H.235 version 3 into a suite of ITU-T H.235.x sub-series Recommendations, and it restructured the sub-series. In earlier versions, prior to version 4 of the ITU-T H.235 sub-series, this profile was contained in the main body and Annex D of Recommendation ITU-T H.235 version 1. Appendices IV, V and VI to Recommendation ITU-T H.235.0 show the complete clause, figure and table mapping between Recommendation ITU-T H.235 versions 3 and 4.

History

Edition	Recommendation	Approval	Study Group	Unique ID*
1.0	ITU-T H.235.6	2005-09-13	16	11.1002/1000/8598-en
1.1	ITU-T H.235.6 (2005) Amd. 1	2008-06-13	16	11.1002/1000/9472-en
2.0	ITU-T H.235.6	2009-03-16	16	11.1002/1000/9694-en
3.0	ITU-T H.235.6	2014-01-13	16	11.1002/1000/12059-en

* To access the Recommendation, type the URL http://handle.itu.int/ in the address field of your web browser, followed by the Recommendation's unique ID. For example, http://handle.itu.int/11.1002/1000/11830-en.

FOREWORD

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

The World Telecommunication Standardization Assembly (WTSA), which meets every four years, establishes the topics for study by the ITU-T study groups which, in turn, produce Recommendations on these topics.

The approval of ITU-T Recommendations is covered by the procedure laid down in WTSA Resolution 1.

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

NOTE

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

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

INTELLECTUAL PROPERTY RIGHTS

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

As of the date of approval of this Recommendation, ITU had not received notice of intellectual property, protected by patents, which may be required to implement this Recommendation. However, implementers are cautioned that this may not represent the latest information and are therefore strongly urged to consult the TSB patent database at http://www.itu.int/ITU-T/ipr/.

Table of Contents

Recommendation ITU-T H.235.6

H.323 security: Encryption profile with native ITU-T H.235/H.245 key management

1 Scope

This Recommendation specifies a security profile for encryption that uses the native ITU-T H.235/H.245 key management. Procedures for both encryption and for the related native ITU-T H.245 key management are specified within this Recommendation.

2 References

The following ITU-T Recommendations and other references contain provisions which, through reference in this text, constitute provisions of this Recommendation. At the time of publication, the editions indicated were valid. All Recommendations and other references are subject to revision; users of this Recommendation are therefore encouraged to investigate the possibility of applying the most recent edition of the Recommendations and other references listed below. A list of the currently valid ITU-T Recommendations is regularly published. The reference to a document within this Recommendation does not give it, as a stand-alone document, the status of a Recommendation.

[ITU-T H.225.0] Recommendation ITU-T H.225.0 v7 (2009), *Call signalling protocols and media stream packetization for packet-based multimedia communication systems*.

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

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

[ITU-T H.235v3] Recommendation ITU-T H.235 v3 (2003), *Security and encryption for H-series (H.323 and other H.245-based) multimedia terminals* plus Corrigendum 1 (2005).

[ITU-T H.235.0] Recommendation ITU-T H.235.0 (2014), *H.323 security: Framework for security in ITU-T H-series (ITU-T H.323 and other ITU-T H.245-based) multimedia systems*.

[ITU-T H.235.1] Recommendation ITU-T H.235.1 (2005), *H.323 security: Baseline security profile*.

[ITU-T H.235.2] Recommendation ITU-T H.235.2 (2005), *H.323 security: Signature security profile*.

[ITU-T H.235.3] Recommendation ITU-T H.235.3 (2005), *H.323 security: Hybrid security profile*.

[ITU-T H.245] Recommendation ITU-T H.245 (2005), *Control protocol for multimedia communication*.

[ITU-T H.323] Recommendation ITU-T H.323 (2009), *Packet-based multimedia communications systems*.

[ITU-T H.530] Recommendation ITU-T H.530 (2002), *Symmetric security procedures for H.323 mobility in H.510*.

[ITU-T X.800]	Recommendation ITU-T X.800 (1991) | ISO/IEC 7498-2:1989, *Security architecture for Open Systems Interconnection for CCITT applications*.
[ITU-T X.803]	Recommendation ITU-T X.803 (1994) | ISO/IEC 10745:1995, *Information technology – Open Systems Interconnection – Upper layers security model*.
[ITU-T X.810]	Recommendation 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]	Recommendation ITU-T X.811 (1995) | ISO/IEC 10181-2:1996, *Information technology – Open Systems Interconnection – Security frameworks for open systems: Authentication framework*.
[IETF RFC 2198]	IETF RFC 2198 (1997), *RTP Payload for Redundant Audio Data*.
[IETF RFC 2246]	IETF RFC 2246 (1999), *The TLS Protocol Version 1.0*.
[IETF RFC 2401]	IETF RFC 2401 (1998), *Security Architecture for the Internet Protocol*.
[IETF RFC 2833]	IETF RFC 2833 (2000), *RTP Payload for DTMF Digits, Telephony Tones and Telephony Signals*.
[IETF RFC 3546]	IETF RFC 3546 (2003), *Transport Layer Security (TLS) Extensions*.
[ISO/IEC 9797-1]	ISO/IEC 9797-1:2011, *Information technology – Security techniques – Message Authentication Codes (MACs) – Part 1: Mechanisms using a block cipher*.
[ISO/IEC 9797-2]	ISO/IEC 9797-2:2011, *Information technology – Security techniques – Message Authentication Codes (MACs) – Part 2: Mechanisms using a dedicated hash-function*.
[ISO/IEC 10116]	ISO/IEC 10116:2006, *Information technology – Security techniques – Modes of operation for an n-bit block cipher*.
[ISO/IEC 10118-3]	ISO/IEC 10118-3:2004, *Information Technology – Security techniques – Hash-functions – Part 3: Dedicated hash-functions*.
[NIST FIPS 197]	NIST FIPS 197 (2001), *Advanced Encryption Algorithm (AES)*. <http://csrc.nist.gov/publications/fips/fips197/fips-197.pdf>

3 Terms and definitions

For the purposes of this Recommendation, the definitions given in clause 3 of [ITU-T H.323], [ITU-T H.225.0] and [ITU-T H.245] apply. Some of the terms used in this Recommendation are also defined in [ITU T X.800], [ITU-T X.803], [ITU-T X.810] and [ITU-T X.811].

The **session key** for encrypting media streams is generated by the master for each RTP session of a call and exchanged via an OLC. The generated session key is encrypted with a key that is derived from the agreed Diffie-Hellman **shared secret** that both end points have computed. In this case, the DH-shared secret acts as Master Key for protection of the session key(s).

4 Abbreviations and acronyms

This Recommendation uses the following abbreviations and acronyms:

3DES	Triple Data Encryption Standard algorithm
AES	Advanced Encryption Standard algorithm
ASN.1	Abstract Syntax Notation One
CBC	Cipher Block Chaining
CFB	Cipher Feedback
DES	Data Encryption Standard
DH	Diffie-Hellman
DTMF	Dual Tone Multi-Frequency
ECB	Electronic Code Book
EOFB	Enhanced Output Feedback mode
EP	End Point
FEC	Forward Error Correction
GK	Gatekeeper
HMAC	Hashed Message Authentication Code
IPsec	Internet Protocol Security
IV	Initialization Vector
KS	Salting Key in EOFB mode
MAC	Message Authentication Code
MC	Multipoint Controller
MCU	Multipoint Control Unit
MPS	Multiple Payload Stream
OFB	Output Feedback Mode
OID	Object Identifier
OLC	Open Logical Channel
RAS	Registration, Admission and Status
RC	Rivest Cipher
ROC	Rollover Counter
RSA	Rivest, Shamir and Adleman
RTCP	Real-time Transport Control Protocol
RTP	Real-time Transport Protocol
SDU	Service Data Unit
SEQ	Sequence number
SHA	Secure Hash Algorithm
TCP	Transmission Control Protocol
TLS	Transport Layer Security

TSAP Transport Service Access Point

UDP User Datagram Protocol

XOR Exclusive OR

5 Conventions

In this Recommendation the following conventions are used:

– **"shall"** indicates a mandatory requirement.

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

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

When deploying media encryption in conjunction with payload padding, the text sometimes says "the value of the pad should be determined by the normal convention of the cipher algorithm", for examples refer to clauses 7.6.1, 8.3 and Figure I.7. This means that some cipher algorithms (e.g., DES) provide further implementation advice as to how the sender may choose the value of the padding byte(s). Examples could be random fill-in values, static values or other generated patterns. Whatever method is deployed does not impact interoperability, though the security quality may well be different. This is considered as an implementation matter and is not specified any further in this Recommendation.

6 System overview

6.1 Encryption security profile

The 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 set-up procedures; e.g., the Diffie-Hellman key agreement and other key management functions.

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

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

This Recommendation further profiles the list of candidate encryption algorithms that Annex D of [ITU-T H.235v2] or [ITU-T H.235v3] had offered.

NOTE 1 – This further encryption algorithm profile takes into account the known cryptanalysis and security findings on the strengths of encryption algorithms and the change in crypto export policies. In particular, the profiling of encryption algorithms of this Recommendation takes into account interoperability requirements with systems complying with [ITU-T H.235v2] or [ITU-T H.235v3].

ITU-T H.323 entities that implement this Recommendation with version 4 of Recommendation ITU-T H.235 or higher versions shall offer 128-bit AES as the preferred encryption algorithm in their offered security capabilities. Those ITU-T H.323 entities may additionally and optionally also offer 168-bit Triple-DES as an encryption algorithm to achieve higher interoperability with ITU-T

H.323 systems that have implemented the encryption features in Annex D of [ITU-T H.235v2] and [ITU-T H.235v3]. Since 56-bit DES and 56-bit RC2-compatible (exportable) encryption algorithms are no longer considered sufficiently secure, ITU-T H.323 entities should not offer these particular weak encryption algorithms unless there is a specific need, such as achieving interoperability with encryption systems in Annex D of [ITU-T H.235v2] and [ITU-T H.235v3]. ITU-T H.323 entities having higher security requirements may offer 192-bit or 256-bit AES as an optional algorithm.

ITU-T H.323 entities that implement this Recommendation with version 4 of Recommendation ITU-T H.235 should accept AES with the highest key length offered if allowed by their security policy. These ITU-T H.323 entities should additionally accept 168-bit Triple-DES if AES was not offered or is otherwise not allowed by their security policy. These ITU-T H.323 entities should not accept 56-bit DES or 56-bit RC2-compatible for security reasons, unless their security policy explicitly allows such insecure encryption algorithms, or exportability needs require such algorithms and other more secure alternatives like AES or 168-bit Triple-DES are not offered.

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

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

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 clause 11).

Table 1 summarizes the security features of the encryption profile. The encryption profile is specified in clauses 7, 8 and 9.

Table 1 – Encryption profile

<table>
<tr><th rowspan="2">Security services</th><th colspan="4">Call functions</th></tr>
<tr><th>RAS</th><th>ITU-T H.225.0</th><th>ITU-T H.245</th><th>RTP</th></tr>
<tr><td>Authentication and integrity</td><td></td><td></td><td></td><td></td></tr>
<tr><td>Non-repudiation</td><td></td><td></td><td></td><td></td></tr>
<tr><td>Confidentiality</td><td></td><td></td><td></td><td>56-bit DES | 56-bit RC2-compatible | 168-bit triple-DES | 128-bit AES | 192-bit AES | 256-bit AES</td></tr>
<tr><td></td><td></td><td></td><td></td><td>CBC-mode or EOFB-mode</td></tr>
<tr><td>Access control</td><td></td><td></td><td></td><td></td></tr>
<tr><td>Key management</td><td></td><td>Authenticated Diffie-Hellman key-exchange</td><td>Integrated ITU-T H.235 session key management (Authenticated Diffie-Hellman key-exchange, key update)</td><td></td></tr>
</table>

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 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 confidentiality is assured by other means, there is no need to implement the media encryption and the related key management procedures (Diffie-Hellman key agreement, key update and synchronization).

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

NOTE 2 – 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 clause 9.3.1.
- Padding, if needed, is to occur as described in clause 9.3.2.

The payload shall be encrypted using the negotiated encryption algorithm ("X", "Y", "Z", "Z3", "Z4", "Z5") according to the procedures described in clauses 9 and 9.3 and the ciphertext padding methods of clause I.1. The audio payload may be encrypted using the negotiated encryption algorithm ("X1", "Y1", "Z1" or "Z2") operating in a stream cipher mode (EOFB).

7 ITU-T 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.

7.1 Secure ITU-T H.245 channel operation

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

7.2 Unsecured ITU-T H.245 channel operation

Alternatively, the ITU-T 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 ([IETF RFC 2246], [IETF RFC 3546]) or IPsec ([IETF RFC 2401]) 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**.

7.3 Capability exchange

Following the procedures in clause 5.2 of [ITU-T H.245] (Capability exchange procedures) and the appropriate ITU-T H-series system Recommendation, end points exchange capabilities using ITU-T H.245 messages. These capability sets may now contain definitions which indicate security

and encryption parameters. For example, an end point might provide capabilities to send and receive ITU-T H.261 video. It may also signal the ability to send and receive encrypted ITU-T 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, end points may supply both independent and dependent encrypted codecs in their exchange. This will allow end points to scale their security capabilities based upon the overheads and resources available.

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

7.4 Master role

The ITU-T 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 end point 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.

7.5 Logical channel signalling

End points 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 end point (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.

7.6 Fast connect security

End points may deploy the fast connect procedure (see clauses 8.1.7 and 8.1.7.1 of [ITU-T H.323]) using the fast start element for securely exchanging key material (master key and session encryption keys). The procedures given in clause 7.6.1 describe "plain" fast start that does not use multiple offered encryption algorithms whereas clause 7.6.1.1 describes the particular case of fast start with multiple offered encryption algorithms that enables more compact message encoding.

7.6.1 Unidirectional fast start security

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

Procedures of the caller

The caller (source of the SETUP) presents both its DH token, and the supported FastStart structures. The DH token shall be conveyed within an embedded ClearToken as part of a CryptoToken, or as a separate ClearToken; see also clause 7.8. During the SETUP-to-CONNECT sequence, a Diffie-Hellman (DH) exchange shall be performed, this seeds both end points with a shared secret. The **ClearToken** field of the **CryptoToken** fields shall contain a **dhkey** for a key length up to 2048 bits or a **dhkeyext** for a key length greater than 2048, 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 Appendix E.2 of [b-IETF RFC 2412].

NOTE 1 – Since the ITU-T H.225.0 messages are authenticated (as described earlier by procedure I), the DH exchange is an authenticated one.

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

The FastStart structures hold the offered open logical channels with the proposed security capabilities. Both H235Cap and nonH235Cap channels should be offered. During the ITU-T H.245 Cap exchange, end points present **H235SecurityCapability** entries for the codecs that they support. Each codec is associated with a separate ITU-T H.235 security capability. According to Table 6, these capabilities should indicate support for 128-bit AES-CBC (OID – "Z3"), 192-bit AES-CBC (OID – "Z4"), 256-bit AES-CBC (OID – "Z5"), 56-bit RC2-compatible-CBC (OID – "X"), should indicate support for 56-bit DES-CBC (OID – "Y") and may indicate support for 168-bit Triple-DES-CBC (OID – "Z"), or 168-bit Triple-DES-EOFB (OID – "Z1"), RC2-compatible-EOFB (OID – "X1"), DES-EOFB (OID – "Y1") or AES-EOFB (OID – "Z2").

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

For the security relationship's purpose, the callee is the *a priori* master; see also clause 7.4.

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

NOTE 2 – In this case, if the callee sends encrypted media to the caller (which theoretically it may do, because it has the caller's RTP/RTCP addresses), the caller will not be able to decipher it without the shared secret provided in the (Alerting, Call Proceeding) CONNECT message.

Procedures of the callee

During FastStart, the callee presents its DH token (see also clause 7.8) and the accepted FastStart structures. In case the Diffie-Hellman procedure is applied, it is recommended that the callee returns its DH token as part of the response message at the earliest opportunity; i.e., in the response message immediately following the SETUP. This allows the caller to compute the master key from the DH shared secret and to be prepared for receiving the session key and encrypted media.

NOTE 3 – In case there is no encryption algorithm available at both sides, the media stream may be left unencrypted or the connection may be aborted, depending on the security policy.

Each entity shall take the appropriate least significant bits from the common shared Diffie-Hellman secret for the key encryption key (master key); i.e., the 56 least significant bits of the

Diffie-Hellman secret for OID "X", OID "X1", OID "Y1" or OID "Y" and the 168 least significant bits of the Diffie-Hellman secret for OID "Z", OID "Z1" or OID "Z2" and the 128 least significant bits of the Diffie-Hellman secret for OID "Z3" or OID "Z2"; see also Table 6.

OpenLogicalChannel(Ack) responses are issued with the (master) created session key included in the **encryptionSync** field. This **encryptionSync** holds the session key for the directed logical channel from caller to callee. Key transport shall proceed according to the procedure described in clause 8.3, using either **KeySyncMaterial** or **V3KeySyncMaterial** (see clause 8.3.1). The session key shall be encrypted with the DH shared secret in a manner described below.

NOTE 4 – 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 the generation of weak keys.

Using the procedure of clause 8.3, the encrypted session key shall be carried in the **H.235Key**/**sharedSecret** within the **encryptionSync** field. The session key shall be carried in the **keyMaterial** field of the **KeySyncMaterial**; if it is not a multiple of the block size, it shall be padded to a multiple of blocks before encryption. The value of the pad should be determined by the normal convention of the cipher algorithm. The (padded) **KeySyncMaterial** shall be encrypted using:

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

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

Alternatively and preferably, the improved key transport according to clause 8.3.1 should be used when possible, due to the outcome of the version 3 indicating procedure (see clause 8.2).

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

7.6.1.1 Using multiple encryption algorithms in fast connect

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

The encryption algorithm to be applied to a media stream is specified through inclusion of the **dataType.h235Media.encryptionAuthenticationAndIntegrity.encryptionCapability dataType** in the **OLC**. The ITU-T H.235 version 2 practice is to include only a single **MediaEncryptionAlgorithm** in the **encryptionCapability**, although the latter element is defined as a sequence of the former elements. This procedure permits the inclusion of a preference-ordered sequence of encryption capabilities in each offered **OLC**. The receiver of the **OLC** shall then select a single algorithm from among those offered, and shall return the **OLC** with only the selected algorithm present (along with the appropriate transport addresses and encryption key information.)

In order to provide the maximum efficiency, the Object ID "NULL-ENCR" (see Table 2) represents the "null" encryption algorithm which means that no encryption operation is to take place. Using this particular method requires only one **OLC** per offered codec per direction.

Table 2 – Object identifier for NULL encryption

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

Procedure for the caller (see clause 8.1.7.1 of [ITU-T H.323])

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

Procedure for the callee (see clause 8.1.7.1 of [ITU-T H.323])

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

7.6.2 Bidirectional fast start security

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

7.7 Encrypted ITU-T H.245 DTMF

End points may choose to send encrypted DTMF [IETF RFC 2833] signals to achieve confidentiality. Using the session encryption key, end points may encrypt the DTMF [IETF RFC 2833] signals in **UserInputIndication** as:

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

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

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

The key management as specified by clause 6.1 should be applied to yield a session encryption key. That session encryption key shall be used to encrypt the ITU-T H.245 DTMF ([IETF RFC 2833]) signals.

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

However, when also using the DTMF ([IETF RFC 2833]) via RTP by setting the **rtpPayloadIndication** flag, it is highly recommended that the RTP payload be secured using the encryption profile of clause 6.1.

Table 3 provides the available encryption algorithms (DES, 3DES or AES) which should deploy EOFB (incl. OFB as a special case, see clause 8.4). To avoid potential padding of the DTMF ([IETF RFC 2833]) characters, CBC, CFB or other block chaining modes that may make padding necessary, are not recommended for the encryption of DTMF ([IETF RFC 2833]) signals.

7.7.1 Encrypted basic string

If **encryptedBasicString** in **UserInputCapability** has been selected, then **encryptedAlphanumeric** shall indicate the applied encryption algorithm within **algorithmOID**,

paramS holds the initial value for the encryption operation. The encrypted alphanumeric string shall be placed in **encrypted**.

7.7.2 Encrypted iA5 string

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

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

7.7.3 Encrypted general string

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

7.7.4 List of object identifiers

Table 3 – Object identifiers for ITU-T H.245 DTMF encryption

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

7.8 Diffie-Hellman operation

This Recommendation supports the Diffie-Hellman protocol for end-to-end key agreement. Depending on the situation, the negotiated Diffie-Hellman key may act as master key (see clause 6.1) or as a dynamic session key ([ITU-T H.235.3] and [ITU-T H.530]).

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

[ITU-T H.235.0] conveys a Diffie-Hellman instance (g, p, g^x) encoded within a **ClearToken** where **dhkey** or **dhkeyext** holds the **halfkey** g^x mod p (resp. g^y mod p) for some secret random x (resp. y), the prime p in **modsize** and the **generator** g. A special case is the triplet (0, 0, 0) or an empty **dhkey** that does not represent any DH-instance but shall be used in signalling that the encryption profile is not being used.

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

For the fixed DH system parameters, a shorthand characterization through an object identifier may yield more compact encoded messages than including literal values. A **ClearToken** that carries a DH-instance with fixed, standardized DH parameters, may reference the DH instance with a DH-OID in the **tokenOID** field; unless the **tokenOID** is used for other purposes (such as in clause 7 of [ITU-T H.235.1] for a distinguished **CryptoToken**). The sender may additionally include the literal DH values but need not do so.

In case several DH-instances are to be indicated each through a DH-OID, the DH-parameters in a distinguished **CryptoToken** (which is being occupied by ITU-T H.235.1) shall be omitted by leaving **dhkey** or **dhkeyext** absent, and all DH-instances shall then be carried within separate **ClearTokens** where the **tokenOID** holds the DH-OID, and **dhkey** or **dhkeyext** may be left absent; any other fields within that **ClearToken** shall not be used.

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

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

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

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

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

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

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

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

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

7.8.1 Requesting renegotiation of DH parameters in the middle of the call

An ITU-T H.323 gatekeeper may request renegotiation of the DH parameters in the middle of the call using the procedures defined in this clause. Such renegotiation procedures may be needed to

establish DH key agreement between an end point already connected to the gatekeeper and an end point to be connected (see Figure 1). The procedure for renegotiating the Diffie-Hellman parameters is needed in supporting several supplementary services. All the procedures defined in this clause shall be performed only when the ITU-T H.323 end points are in "transmitter side paused" state, defined in clause 8.4.6 of [ITU-T H.323].

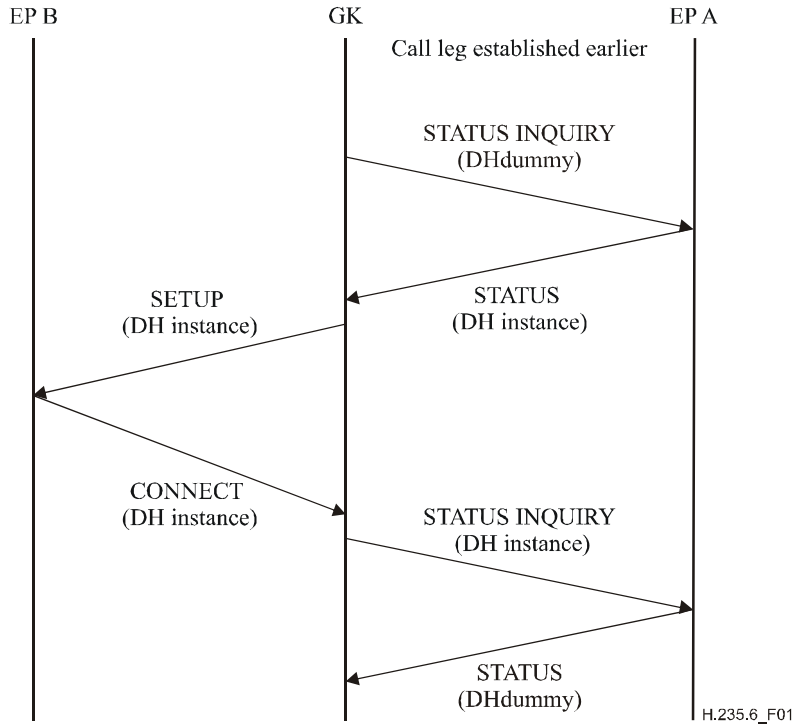


Figure 1 – Usage of "Requesting DH parameters in the middle of the call" for supplementary services

To request DH parameters in the middle of the call, the ITU-T H.323 entity shall send a STATUS INQUIRY message containing a ClearToken field with DH-OID "DHdummy" in the tokenOID field and the rest of the fields omitted.

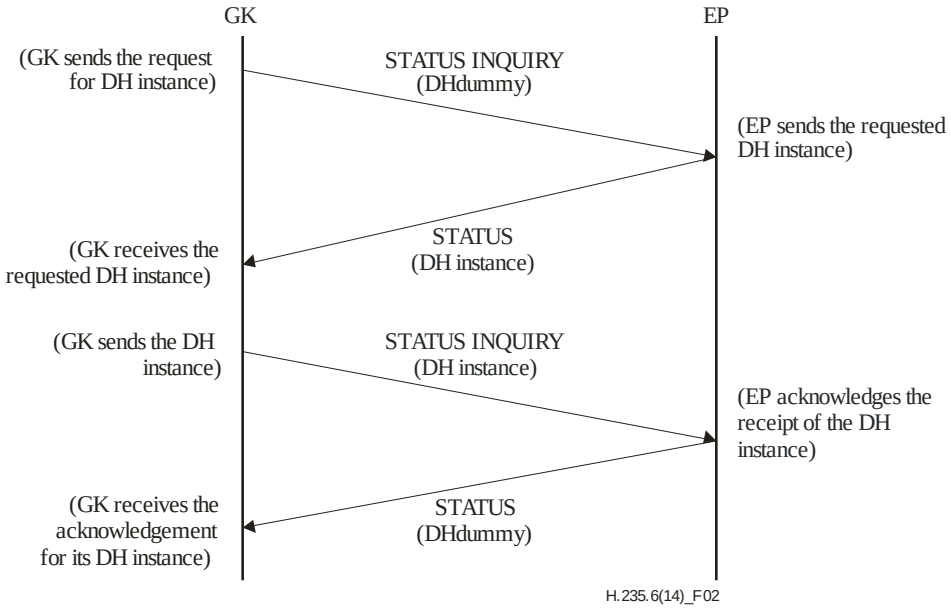


Figure 2 – Requesting DH parameters in the middle of the call

If an ITU-T H.323 entity receives the STATUS INQUIRY message containing **ClearToken** field with DH-OID "DHdummy" in the **tokenOID** field, the ITU-T H.323 end point shall respond with STATUS message containing the set of DH instances; see Figure 2. The DH instances shall be specified in this STATUS message according to the rules defined in clause 7.8 for the SETUP message.

NOTE 1 – The ITU-T H.323 entity which does not support this procedure is supposed to respond to STATUS INQUIRY with STATUS message without DH instances.

To convey the accepted DH instance in the middle of the call, the ITU-T H.323 entity shall send STATUS INQUIRY containing the accepted DH instance; see Figure 2. The DH instances shall be specified in this STATUS INQUIRY message according to the rules defined above in clause 7.8 for the response to the SETUP message.

If an ITU-T H.323 end point receives such a STATUS INQUIRY message containing **ClearToken** field with a DH instance, the ITU-T H.323 end point shall respond with STATUS message containing **ClearToken** field with DH-OID "DHdummy" in the **tokenOID** field and the rest of the fields omitted.

NOTE 2 – The ITU-T H.323 entity which does not support this procedure is supposed to respond to STATUS INQUIRY with STATUS message without DH instances.

The ITU-T H.323 end point receiving the STATUS INQUIRY message with DH instance shall recalculate the DH shared secret from this DH instance and the latest set of DH instance(s) which has been sent by this ITU-T H.323 end point in the particular call.

If an ITU-T H.323 GK receives a STATUS INQUIRY message containing **ClearToken** field with a DH instance, or with DH-OID "DHdummy" in the **tokenOID** field, then, with the exception of the several cases outlined below, it shall forward the message to the second leg of the call in the context of which the message has been received.

If an ITU-T H.323 GK receives a STATUS response to the STATUS INQUIRY message it has forwarded, the GK shall forward back the STATUS message to the call leg on which the STATUS INQUIRY message has been received.

If an ITU-T H.323 GK is waiting for a response to the STATUS INQUIRY message containing **ClearToken** field, with DH-OID "DHdummy" in the **tokenOID** field which it has sent, receives STATUS INQUIRY message containing **ClearToken** field with DH-OID "DHdummy" in the **tokenOID** field and with CRV flag set to value 1, then the GK shall respond with the STATUS message containing **ClearToken** field with DH-OID "DHdummy" in the **tokenOID** field (see Figure 3).

If an ITU-T H.323 GK receives a STATUS INQUIRY message containing **ClearToken** field with a DH instance or with DH-OID "DHdummy" in the **tokenOID** field while the second leg of the call is not established, the GK shall wait for the establishment of the second leg of the call, send an empty capability set on this call leg and then forward to it the received STATUS INQUIRY message (see Figure 3).

An ITU-T H.323 GK shall not initiate procedures defined in this clause after it has sent STATUS message containing a DH instance and before it has received the STATUS INQUIRY message containing a DH instance.

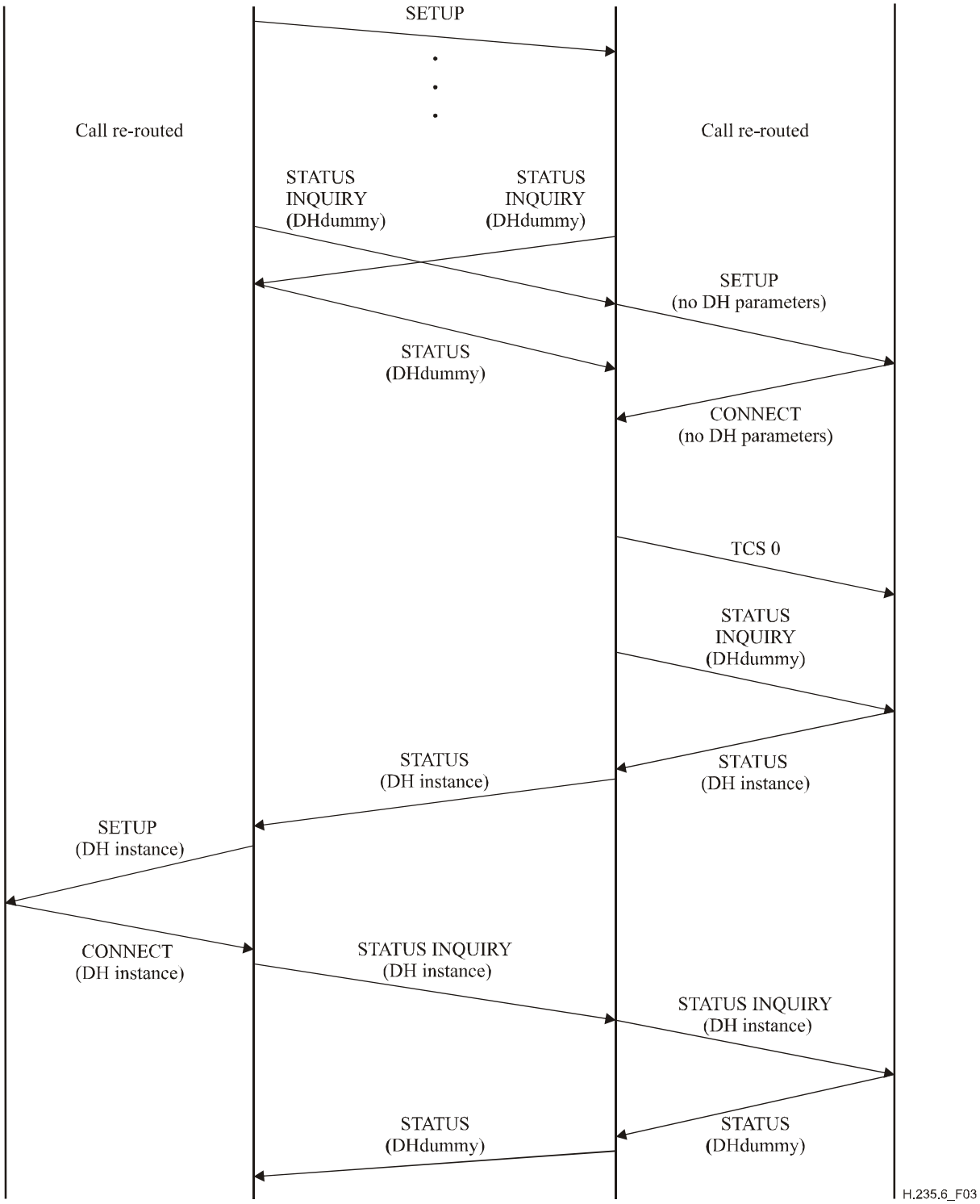


Figure 3 – Usage of "Requesting DH parameters in the middle of the call" for simultaneous call re-route by both GKs

8 Signalling and procedures

The procedures outlined in clause 8 of [ITU-T H.323] (Call signalling procedures) shall be followed. The ITU-T H.323 end points shall have the ability to encode and recognize the presence (or absence) of security requirements (for the ITU-T H.245 channel) signalled in the ITU-T H.225.0 messages.

In the case where the ITU-T H.225.0 channel itself is to be secured, the same procedures in clause 8 of [ITU-T 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 ([IETF RFC 2246], [IETF RFC 3546])). Due to the fact that the ITU-T H.225.0 messages are the first exchanged when establishing ITU-T H.323 communications, there can be no security negotiations "in band" for ITU-T H.225.0. In other words, both parties must know *a priori* that they are using a particular security mode. For ITU-T H.323 on IP, an alternative Well Known Port (1300) is utilized for TLS ([IETF RFC 2246], [IETF RFC 3546]) secured communications.

One purpose of ITU-T H.225.0 exchanges as they relate to ITU-T H.323 security is to provide a mechanism to set up the secure ITU-T H.245 channel. Optionally, authentication may occur during the exchange of ITU-T H.225.0 messages. This authentication may be certificate-based or password-based, utilizing encryption and/or hashing (i.e., signing). The specifics of these modes of operation are described in clauses 8.1 to 8.2.3 of [ITU-T H.235.0].

An ITU-T H.323 end point 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 a sufficient or 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 ITU-T H.245 channel in the indicated secure mode. Failure to set up the ITU-T H.245 channel in the secure mode determined here should be considered a protocol error and the connection terminated.

8.1 Revision 1 compatibility

A security capable end point shall not return any security-related fields, indications or status to the non-security capable end point. 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***.

8.2 Version 3 feature indication

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

- the improved key transport (**V3KeySyncMaterial**, see clause 8.3.1)
- the improved key update; see clause 8.6.2.

Since end points usually do not know about their mutual support of ITU-T H.235 version 3 or higher versions, an explicit version indication is added during call set-up.

ITU-T H.235 version 3 and higher version end points should always use the procedure described in this clause for determining version 3 capability (improved key transport, improved encryption sync). Depending on the outcome of the logical signalling procedure, the end points may use the procedures (see clause 8.3) for backward compatibility with ITU-T H.235 version 1 or with ITU-T H.235 version 2 end points.

In order to indicate whether to use the improved ITU-T H.235 version 3 procedures, the calling and the called end point shall include an additional **ClearToken** indicating version 3 capability during the call signalling (SETUP, CONNECT, etc.). Absence of such a **ClearToken** would indicate support of only ITU-T H.235 version 1 or version 2. In this case, the end point shall use the procedure of clause 8.3. Otherwise, the end point may use the improved procedures as described in clause 8.3.1, or use the ITU-T H.235 version 1 or the ITU-T H.235 version 2 procedure of clause 8.3.

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

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

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

8.3 Key transport

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

- A procedure primarily for ITU-T H.235 version 1 or version 2 end points; this is described in this clause.
- An improved procedure for ITU-T H.235 version 3 and higher end points, this is described in clause 8.3.1.

ITU-T H.235 version 1 or ITU-T H.235 version 2 end points apply the following procedure for session key transport:

KeySyncMaterial holds the end point identifier of the master within **generalID** and carries the session key material within **keyMaterial**. The **generalID** value should be included to provide a minimal level of authentication of the source of the session key (see also clause 8.6). The recipient should verify correctness of the received **generalID**.

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

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

8.3.1 Improved key transport in ITU-T H.235 version 3

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

Furthermore, version 3 of ITU-T H.235 shall be capable of transporting additional key material:

- Secure transport of a salting key to the peer(s). Such a salting key is being introduced for the enhanced OFB mode; see clause 8.4.

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

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

algorithmOID indicates the applied encryption algorithm and the operation mode.

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

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

encryptedSessionKey holds the ciphertext of the encrypted raw session key.

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

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

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

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

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

ITU-T H.235 version 3 or higher end points should always use **secureSharedSecret** within **H235Key,** but depending on the outcome of the logical signalling procedure in clause 8.2, using the indicating version 3 **ClearToken**, may use **sharedSecret** for backwards compatibility with ITU-T H.235 version 1 or with ITU-T H.235 version 2 end points.

8.4 Enhanced OFB mode

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

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

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

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

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

EOFB is defined as $C_i = P_i \oplus S_i$ with $S_i = E_{KE}(KS \oplus S_{i-1})$ for $i = 1 \ldots n$ and $S_0 = IV$ where C_i is the i-th ciphertext block, P_i the i-th plaintext block, S_i the i-th key stream block, KE the encryption key and $\oplus$ bitwise XOR. EOFB is illustrated in Figure I.6.

EOFB may also run in standard OFB mode, making EOFB backwards compatible with OFB. In those cases where backwards compatibility with standard OFB mode is desired, the salting key KS shall be either set to all zeroes or equally, leaving **encryptedSaltingKey** within **V3KeySyncMaterial** empty. However, usage of an actual salting key is highly recommended for those cases when encrypting RTP payloads with a block cipher that has a shorter block size such as DES-56 or RC2-compatible.

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

Clause 11 defines object identifiers for DES-56-EOFB, RC2-compatible-EOFB, 3-DES-EOFB and AES-EOFB.

8.5 Key management

End points conforming to this Recommendation should use the fast connect procedure according to clause 7.6.1. If fast start is not applied, then ITU-T H.245 tunnelling shall be used to secure the ITU-T H.245 call control messages by this Recommendation. The fast start procedures allow the establishment of either one or two unidirectional logical channels. The fast start procedure cares for negotiation of the security capabilities, for distribution of a common shared secret (shared DH secret) which acts as a master key, and for secure distribution of an encryption key.

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

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

  NOTE 1 – No particular DH group is defined; the OID references any non-standard DH group.
- An instance of a 512-bit DH group shall be used to generate a master key for distribution of session key(s) for RC2-compatible ("X") or for DES-56 bit encryption algorithms ("Y").
- "DH1024": The OID references a standardized, fixed 1024-bit DH group. This DH group shall be used to generate a master key for distribution of session key(s) for Triple-DES ("Z") encryption algorithms.
- "DH1536": This DH group is offered as an option for version 3 end points having security requirements that exceed the security of a 1024-bit DH group. The OID references a fixed 1536-bit DH group. This DH group shall be used to generate a master key for distribution of session key(s) for Triple-DES ("Z", "Z1") or for AES-128 ("Z2", "Z3") encryption algorithms.
- "DH2048": This DH group is offered as an option for version 3 end points having security requirements that exceed the security of a 1536-bit DH group. The OID references a fixed 2048-bit DH group. This DH group shall be used to generate a master key for distribution of session key(s) for AES-192 or AES-256 ("Z4", "Z5") encryption algorithms.
- "DH3072": This DH group is offered as an option for version 4 end points having security requirements that exceed the security of a 2048-bit DH group. The OID references a fixed 3072-bit DH group. This DH group shall be used to generate a master key for distribution of session key(s) for AES-192 or AES-256 ("Z4", "Z5") encryption algorithms.
- "DH4096": This DH group is offered as an option for version 4 end points having security requirements that exceed the security of a 3072-bit DH group. The OID references a fixed 4096-bit DH group. This DH group shall be used to generate a master key for distribution of session key(s) for AES-256 ("Z5") encryption algorithms.

- "DH6144": This DH group is offered as an option for version 4 end points having security requirements that exceed the security of a 6144-bit DH group. The OID references a fixed 4096-bit DH group. This DH group shall be used to generate a master key for distribution of session key(s) for AES-256 ("Z5") encryption algorithms.
- "DH8192": This DH group is offered as an option for version 4 end points having security requirements that exceed the security of a 8192-bit DH group. The OID references a fixed 4096-bit DH group. This DH group shall be used to generate a master key for distribution of session key(s) for AES-256 ("Z5") encryption algorithms.

It is recommended to apply the defined 1024-bit or optionally, larger DH groups unless other security needs would make other Diffie-Hellman parameters preferential. Furthermore, it is recommended to consider using the defined OIDs identifying the DH groups, see clause 7.8. Nevertheless, implementations should be prepared to obtain the DH group parameters literally without explicit OID indication. In this case, implementations should ascertain that the correct DH group is being conveyed according to Table 4.

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

NOTE 2 – The choice of the DH group does not eliminate the need to negotiate the actual media encryption algorithm. This is accomplished with the ITU-T H.245 terminal capability negotiation procedure.

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

Table 4 – Diffie-Hellman groups

Encryption algorithm OID	DH-OID	D-H group description
"X", "X1" (RC2-compatible), "Y", "Y1" (DES)	"DHdummy"	Mod-P, any suitable 512-bit prime
"Z", "Z1" (Triple-DES), "Z2", "Z3" (AES)	"DH1024"	Mod-P, 1024-bit prime Prime = $2^{1024} – 2^{960} – 1 + 2^{64} × \{ [2^{894}$ pi] + 129093 } Hexadecimal value = FFFFFFFF FFFFFFFF C90FDAA2 2168C234 C4C6628B 80DC1CD1 29024E08 8A67CC74 020BBEA6 3B139B22 514A0879 8E3404DD EF9519B3 CD3A431B 302B0A6D F25F1437 4FE1356D 6D51C245 E485B576 625E7EC6 F44C42E9 A637ED6B 0BFF5CB6 F406B7ED EE386BFB 5A899FA5 AE9F2411 7C4B1FE6 49286651 ECE65381 FFFFFFFF FFFFFFFF Generator (Note) = 2
"Z", "Z1" (Triple-DES), "Z2", "Z3" (AES)	"DH1536"	Mod-P, 1536-bit prime Prime = $2^{1536} – 2^{1472} – 1 + 2^{64} × \{ [2^{1406}$ pi] + 741804 } Hexadecimal value = FFFFFFFF FFFFFFFF C90FDAA2 2168C234 C4C6628B 80DC1CD1 29024E08 8A67CC74 020BBEA6 3B139B22 514A0879 8E3404DD EF9519B3 CD3A431B 302B0A6D F25F1437 4FE1356D 6D51C245 E485B576 625E7EC6 F44C42E9 A637ED6B 0BFF5CB6 F406B7ED EE386BFB 5A899FA5 AE9F2411 7C4B1FE6 49286651 ECE45B3D

Table 4 – Diffie-Hellman groups

Encryption algorithm OID	DH-OID	D-H group description
		C2007CB8 A163BF05 98DA4836 1C55D39A 69163FA8 FD24CF5F 83655D23 DCA3AD96 1C62F356 208552BB 9ED52907 7096966D 670C354E 4ABC9804 F1746C08 CA237327 FFFFFFFF FFFFFFFF Generator (Note) = 2
"Z2", "Z3", "Z4", "Z5" (AES)	"DH2048"	Mod-P, 2048-bit prime Prime = $2^{2048} - 2^{1984} - 1 + 2^{64} \times \{ [2^{1918} \text{ pi}] + 124476 \}$ Hexadecimal value = FFFFFFFF FFFFFFFF C90FDAA2 2168C234 C4C6628B 80DC1CD1 29024E08 8A67CC74 020BBEA6 3B139B22 514A0879 8E3404DD EF9519B3 CD3A431B 302B0A6D F25F1437 4FE1356D 6D51C245 E485B576 625E7EC6 F44C42E9 A637ED6B 0BFF5CB6 F406B7ED EE386BFB 5A899FA5 AE9F2411 7C4B1FE6 49286651 ECE45B3D C2007CB8 A163BF05 98DA4836 1C55D39A 69163FA8 FD24CF5F 83655D23 DCA3AD96 1C62F356 208552BB 9ED52907 7096966D 670C354E 4ABC9804 F1746C08 CA18217C 32905E46 2E36CE3B E39E772C 180E8603 9B2783A2 EC07A28F B5C55DF0 6F4C52C9 DE2BCBF6 95581718 3995497C EA956AE5 15D22618 98FA0510 15728E5A 8AACAA68 FFFFFFFF FFFFFFFF Generator (Note) = 2
"Z2", "Z3", "Z4", "Z5" (AES)	"DH3072"	Mod-P, 3072-bit prime Prime = $2^{3072} - 2^{3008} - 1 + 2^{64} \times \{ [2^{2942} \text{ pi}] + 1690314 \}$ Hexadecimal value = FFFFFFFF FFFFFFFF C90FDAA2 2168C234 C4C6628B 80DC1CD1 29024E08 8A67CC74 020BBEA6 3B139B22 514A0879 8E3404DD EF9519B3 CD3A431B 302B0A6D F25F1437 4FE1356D 6D51C245 E485B576 625E7EC6 F44C42E9 A637ED6B 0BFF5CB6 F406B7ED EE386BFB 5A899FA5 AE9F2411 7C4B1FE6 49286651 ECE45B3D C2007CB8 A163BF05 98DA4836 1C55D39A 69163FA8 FD24CF5F 83655D23 DCA3AD96 1C62F356 208552BB 9ED52907 7096966D 670C354E 4ABC9804 F1746C08 CA18217C 32905E46 2E36CE3B E39E772C 180E8603 9B2783A2 EC07A28F B5C55DF0 6F4C52C9 DE2BCBF6 95581718 3995497C EA956AE5 15D22618 98FA0510 15728E5A 8AAAC42D AD33170D 04507A33 A85521AB DF1CBA64 ECFB8504 58DBEF0A 8AEA7157 5D060C7D B3970F85 A6E1E4C7 ABF5AE8C DB0933D7 1E8C94E0 4A25619D CEE3D226 1AD2EE6B F12FFA06 D98A0864 D8760273 3EC86A64 521F2B18 177B200C BBE11757 7A615D6C 770988C0 BAD946E2 08E24FA0 74E5AB31 43DB5BFC E0FD108E 4B82D120 A93AD2CA FFFFFFFF FFFFFFFF Generator (Note) = 2

Table 4 – Diffie-Hellman groups

Encryption algorithm OID	DH-OID	D-H group description
"Z2", "Z3", "Z4", "Z5" (AES)	"DH4096"	Mod-P, 4096-bit prime Prime = $2^{4096} – 2^{4032} – 1 + 2^{64} × \{ [2^{3966}$ pi] + 240904 } Hexadecimal value = FFFFFFFF FFFFFFFF C90FDAA2 2168C234 C4C6628B 80DC1CD1 29024E08 8A67CC74 020BBEA6 3B139B22 514A0879 8E3404DD EF9519B3 CD3A431B 302B0A6D F25F1437 4FE1356D 6D51C245 E485B576 625E7EC6 F44C42E9 A637ED6B 0BFF5CB6 F406B7ED EE386BFB 5A899FA5 AE9F2411 7C4B1FE6 49286651 ECE45B3D C2007CB8 A163BF05 98DA4836 1C55D39A 69163FA8 FD24CF5F 83655D23 DCA3AD96 1C62F356 208552BB 9ED52907 7096966D 670C354E 4ABC9804 F1746C08 CA18217C 32905E46 2E36CE3B E39E772C 180E8603 9B2783A2 EC07A28F B5C55DF0 6F4C52C9 DE2BCBF6 95581718 3995497C EA956AE5 15D22618 98FA0510 15728E5A 8AAAC42D AD33170D 04507A33 A85521AB DF1CBA64 ECFB8504 58DBEF0A 8AEA7157 5D060C7D B3970F85 A6E1E4C7 ABF5AE8C DB0933D7 1E8C94E0 4A25619D CEE3D226 1AD2EE6B F12FFA06 D98A0864 D8760273 3EC86A64 521F2B18 177B200C BBE11757 7A615D6C 770988C0 BAD946E2 08E24FA0 74E5AB31 43DB5BFC E0FD108E 4B82D120 A9210801 1A723C12 A787E6D7 88719A10 BDBA5B26 99C32718 6AF4E23C 1A946834 B6150BDA 2583E9CA 2AD44CE8 DBBBC2DB 04DE8EF9 2E8EFC14 1FBECAA6 287C5947 4E6BC05D 99B2964F A090C3A2 233BA186 515BE7ED 1F612970 CEE2D7AF B81BDD76 2170481C D0069127 D5B05AA9 93B4EA98 8D8FDDC1 86FFB7DC 90A6C08F 4DF435C9 34063199 FFFFFFFF FFFFFFFF Generator (Note) = 2
"Z5" (AES)	"DH6144"	Mod-P, 8192-bit prime Prime = $2^{6144} – 2^{6080}– 1 + 2^{64} × \{ [2^{6014}$ pi] + 929484 } Hexadecimal value = FFFFFFFF FFFFFFFF C90FDAA2 2168C234 C4C6628B 80DC1CD1 29024E08 8A67CC74 020BBEA6 3B139B22 514A0879 8E3404DD EF9519B3 CD3A431B 302B0A6D F25F1437 4FE1356D 6D51C245 E485B576 625E7EC6 F44C42E9 A637ED6B 0BFF5CB6 F406B7ED EE386BFB 5A899FA5 AE9F2411 7C4B1FE6 49286651 ECE45B3D C2007CB8 A163BF05 98DA4836 1C55D39A 69163FA8 FD24CF5F 83655D23 DCA3AD96 1C62F356 208552BB 9ED52907 7096966D 670C354E 4ABC9804 F1746C08 CA18217C 32905E46 2E36CE3B E39E772C 180E8603 9B2783A2 EC07A28F B5C55DF0 6F4C52C9 DE2BCBF6 95581718 3995497C EA956AE5 15D22618 98FA0510 15728E5A 8AAAC42D AD33170D 04507A33 A85521AB DF1CBA64 ECFB8504 58DBEF0A 8AEA7157 5D060C7D B3970F85 A6E1E4C7 ABF5AE8C DB0933D7 1E8C94E0 4A25619D CEE3D226 1AD2EE6B F12FFA06 D98A0864 D8760273 3EC86A64 521F2B18 177B200C BBE11757 7A615D6C 770988C0 BAD946E2 08E24FA0 74E5AB31 43DB5BFC E0FD108E 4B82D120 A9210801 1A723C12 A787E6D7 88719A10 BDBA5B26 99C32718 6AF4E23C 1A946834 B6150BDA 2583E9CA 2AD44CE8 DBBBC2DB 04DE8EF9 2E8EFC14 1FBECAA6

Table 4 – Diffie-Hellman groups

Encryption algorithm OID	DH-OID	D-H group description
		287C5947 4E6BC05D 99B2964F A090C3A2 233BA186 515BE7ED 1F612970 CEE2D7AF B81BDD76 2170481C D0069127 D5B05AA9 93B4EA98 8D8FDDC1 86FFB7DC 90A6C08F 4DF435C9 34028492 36C3FAB4 D27C7026 C1D4DCB2 602646DE C9751E76 3DBA37BD F8FF9406 AD9E530E E5DB382F 413001AE B06A53ED 9027D831 179727B0 865A8918 DA3EDBEB CF9B14ED 44CE6CBA CED4BB1B DB7F1447 E6CC254B 33205151 2BD7AF42 6FB8F401 378CD2BF 5983CA01 C64B92EC F032EA15 D1721D03 F482D7CE 6E74FEF6 D55E702F 46980C82 B5A84031 900B1C9E 59E7C97F BEC7E8F3 23A97A7E 36CC88BE 0F1D45B7 FF585AC5 4BD407B2 2B4154AA CC8F6D7E BF48E1D8 14CC5ED2 0F8037E0 A79715EE F29BE328 06A1D58B B7C5DA76 F550AA3D 8A1FBFF0 EB19CCB1 A313D55C DA56C9EC 2EF29632 387FE8D7 6E3C0468 043E8F66 3F4860EE 12BF2D5B 0B7474D6 E694F91E 6DCC4024 FFFFFFFF FFFFFFFF Generator (Note) = 2
"Z5" (AES)	"DH8192"	Mod-P, 8192-bit prime Prime = $2^{8192} – 2^{8128} – 1 + 2^{64} × \{ [2^{8062} \text{ pi}] + 4743158 \}$ Hexadecimal value = FFFFFFFF FFFFFFFF C90FDAA2 2168C234 C4C6628B 80DC1CD1 29024E08 8A67CC74 020BBEA6 3B139B22 514A0879 8E3404DD EF9519B3 CD3A431B 302B0A6D F25F1437 4FE1356D 6D51C245 E485B576 625E7EC6 F44C42E9 A637ED6B 0BFF5CB6 F406B7ED EE386BFB 5A899FA5 AE9F2411 7C4B1FE6 49286651 ECE45B3D C2007CB8 A163BF05 98DA4836 1C55D39A 69163FA8 FD24CF5F 83655D23 DCA3AD96 1C62F356 208552BB 9ED52907 7096966D 670C354E 4ABC9804 F1746C08 CA18217C 32905E46 2E36CE3B E39E772C 180E8603 9B2783A2 EC07A28F B5C55DF0 6F4C52C9 DE2BCBF6 95581718 3995497C EA956AE5 15D22618 98FA0510 15728E5A 8AAAC42D AD33170D 04507A33 A85521AB DF1CBA64 ECFB8504 58DBEF0A 8AEA7157 5D060C7D B3970F85 A6E1E4C7 ABF5AE8C DB0933D7 1E8C94E0 4A25619D CEE3D226 1AD2EE6B F12FFA06 D98A0864 D8760273 3EC86A64 521F2B18 177B200C BBE11757 7A615D6C 770988C0 BAD946E2 08E24FA0 74E5AB31 43DB5BFC E0FD108E 4B82D120 A9210801 1A723C12 A787E6D7 88719A10 BDBA5B26 99C32718 6AF4E23C 1A946834 B6150BDA 2583E9CA 2AD44CE8 DBBBC2DB 04DE8EF9 2E8EFC14 1FBECAA6 287C5947 4E6BC05D 99B2964F A090C3A2 233BA186 515BE7ED 1F612970 CEE2D7AF B81BDD76 2170481C D0069127 D5B05AA9 93B4EA98 8D8FDDC1 86FFB7DC 90A6C08F 4DF435C9 34028492 36C3FAB4 D27C7026 C1D4DCB2 602646DE C9751E76 3DBA37BD F8FF9406 AD9E530E E5DB382F 413001AE B06A53ED 9027D831 179727B0 865A8918 DA3EDBEB CF9B14ED 44CE6CBA CED4BB1B DB7F1447 E6CC254B 33205151 2BD7AF42 6FB8F401 378CD2BF 5983CA01 C64B92EC F032EA15 D1721D03 F482D7CE 6E74FEF6 D55E702F 46980C82 B5A84031 900B1C9E 59E7C97F BEC7E8F3 23A97A7E 36CC88BE 0F1D45B7 FF585AC5 4BD407B2 2B4154AA CC8F6D7E BF48E1D8 14CC5ED2 0F8037E0 A79715EE F29BE328

Table 4 – Diffie-Hellman groups

Encryption algorithm OID	DH-OID	D-H group description
		06A1D58B B7C5DA76 F550AA3D 8A1FBFF0 EB19CCB1 A313D55C DA56C9EC 2EF29632 387FE8D7 6E3C0468 043E8F66 3F4860EE 12BF2D5B 0B7474D6 E694F91E 6DBE1159 74A3926F 12FEE5E4 38777CB6 A932DF8C D8BEC4D0 73B931BA 3BC832B6 8D9DD300 741FA7BF 8AFC47ED 2576F693 6BA42466 3AAB639C 5AE4F568 3423B474 2BF1C978 238F16CB E39D652D E3FDB8BE FC848AD9 22222E04 A4037C07 13EB57A8 1A23F0C7 3473FC64 6CEA306B 4BCBC886 2F8385DD FA9D4B7F A2C087E8 79683303 ED5BDD3A 062B3CF5 B3A278A6 6D2A13F8 3F44F82D DF310EE0 74AB6A36 4597E899 A0255DC1 64F31CC5 0846851D F9AB4819 5DED7EA1 B1D510BD 7EE74D73 FAF36BC3 1ECFA268 359046F4 EB879F92 4009438B 481C6CD7 889A002E D5EE382B C9190DA6 FC026E47 9558E447 5677E9AA 9E3050E2 765694DF C81F56E8 80B96E71 60C980DD 98EDD3DF FFFFFFFF FFFFFFFF Generator (Note) = 2
NOTE – The generator is used to generate the DH token.		

8.6 Key update and synchronization

For 64-bit block ciphers, the key refresh rate *shall* be such that no more than 2^{32} blocks are encrypted using the same key. Implementations *should* refresh keys before 2^{30} blocks have been encrypted using the same key (see clause 9.1). For 128-bit block ciphers, the key refresh rate *shall* be such that no more than 2^{64} blocks are encrypted using the same key. Implementations *should* refresh keys before 2^{62} blocks have been encrypted using the same key (see clause 9.1). Both involved entities are free to change the media session key as often as considered necessary due to their security policy. For example, the master may distribute a new session key using **encryptionUpdate** or **encryptionUpdateCommand** of the **miscellaneousCommand** message. On the other hand, the slave can request a new session key from the master by using the **encryptionUpdateRequest** of the **miscellaneousCommand** message.

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

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

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

- **algorithmOID**: set to "X", "X1" for the 56-bit RC2-compatible, set to "Y", "Y1" for 56-bit DES, set to "Z", "Z1" for 168-bit Triple-DES, set to "Z3" for 128-bit AES, "Z4" for 192-bit AES or set to "Z5" for 256-bit AES.

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

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

As part of the **KeySyncMaterial**:

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

  NOTE 2 – This Recommendation assumes that each end point has registered with a gatekeeper and has obtained an end point identifier that can be conveyed within **generalID**. This Recommendation does not support scenarios without gatekeepers; this remains for further study.
- **keyMaterial**: set to the new session key. For DES and RC2-compatible this is a 56-bit key; for Triple-DES this is a 168-bit key and for AES this is a 128-bit, 192-bit or 256-bit key. The master shall generate a new session key that meets at least the following security criteria: it is not a weak or semi-weak DES-key and uses a sufficiently secure random source.

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

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

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

- Payload-type-based key update and synchronization defines a new dynamic payload type for that new session key; see clauses 8.6.1, 8.6.2 and 8.6.3.

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

8.6.1 Unacknowledged key update

Figure 4 shows the unacknowledged handshake for session key distribution/key update. If the slave desires an updated session key, the slave may request a new session key from the master by issuing an **encryptionUpdateRequest** to the master. The master shall send a new session key (with or without prior **encryptionUpdateRequest** from the slave) to the slave within an **EncryptionUpdate** message.

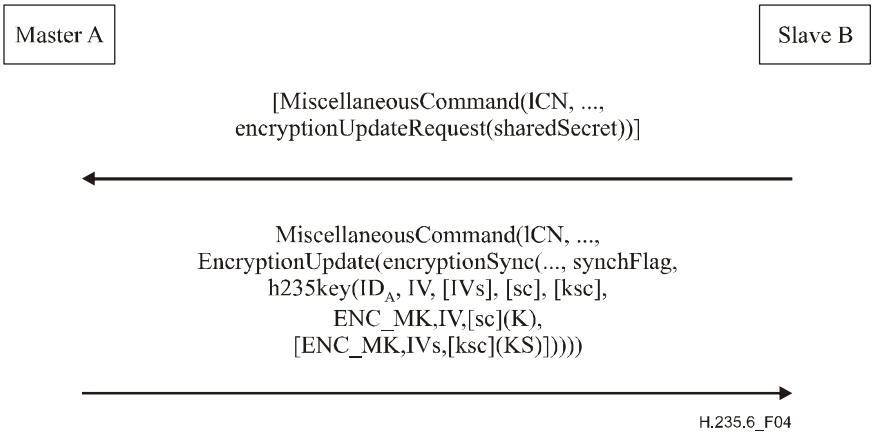


Figure 4 – Unacknowledged session key distribution/key update from the master to the slave(s)

where:

lCN	is the logical channel number;
synchFlag	is the new dynamic RTP payload number;
ID_A	is the **generalID** of the source;
IV	is the initial value/vector for encryption of the sessionkey;
IVs	is the initial value/vector for encryption of the salting key;
ENC_MK,IV,[sc](K)	means encryption of plaintext *K* using key *MK*, initial vector *IV* [and a salting key *sc*, only for EOFB];
KS	is the salting key for the media (for EOFB mode only);
K	is the plaintext session key;
sc	is the unencrypted salting key when EOFB mode is being used for encrypting the session key;
ksc	is the unencrypted salting key when EOFB mode is being used for encrypting the salting key;
s2M/m2S	is the **direction** flag ([ITU-T H.235v3] and higher only) (s2m = slave-to-master, m2s = master-to-slave);
[]	represents an optional part;
MK	is the master key.

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

8.6.2 Improved key update

ITU-T H.235 version 3 and higher end points shall perform an explicit/implicit acknowledged key update procedure. This is to provide reliable key update methods that are based upon the unacknowledged key update method as provided by pre-H.235v3-based versions. The capability for such a procedure shall be negotiated using the version 3 feature indication according to clause 8.2.

Figure 5 shows the key update procedures for a logical channel owned by the slave. In case the slave initiates the key update and requests a new session key from the master, the slave shall send a **MiscellaneousCommand** to the master where **logicalChannelNumber** shall hold the logical channel number (as defined by the slave), **sharedSecret** shall be set to true, the **direction** flag shall be set to **slaveToMaster** and the new dynamic payload number shall be requested in **synchFlag** within **EncryptionUpdateRequest**. If, otherwise, the master initiates the key update, this **EncryptionUpdateRequest** message shall not be sent.

The master, either responding to the slave's request or on its own behalf, shall issue an **EncryptionUpdateCommand** where the **logicalChannelNumber** shall hold the logical channel number, **direction** shall be set to **slaveToMaster** within **MiscellaneousCommand** and **synchFlag** within **encryptionSync** reflects the new dynamic payload number.

h235key shall carry the new session key. **h235key** shall hold the identity of the master in **generalID** and the applied initial vector *IV* in **paramS**. The encrypted media session key shall be conveyed within **encryptedSessionKey**, where the encryption function shall apply the master session key and the initial value in **paramS** to the session key *K***.** For EOFB, an unencrypted salting key is conveyed in **ClearSalt** within **paramS** (*sc*). **encryptedSaltingKey** shall convey the encrypted media salting key, where the encryption function shall apply the master session key and the initial value **paramSsaltIV** to the media salting key *KS*. For EOFB, an unencrypted salting key (*ksc*) is conveyed in **ClearSalt** within **paramSsalt**. **clearSaltingKey** may hold an unencrypted media salting key in which case, **encryptedSaltingKey** shall remain empty and vice versa. The transmission of an unencrypted salting key shall only be achieved if the security does not suffer; in any other case, it is recommended that the media salting key be encrypted.

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

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

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

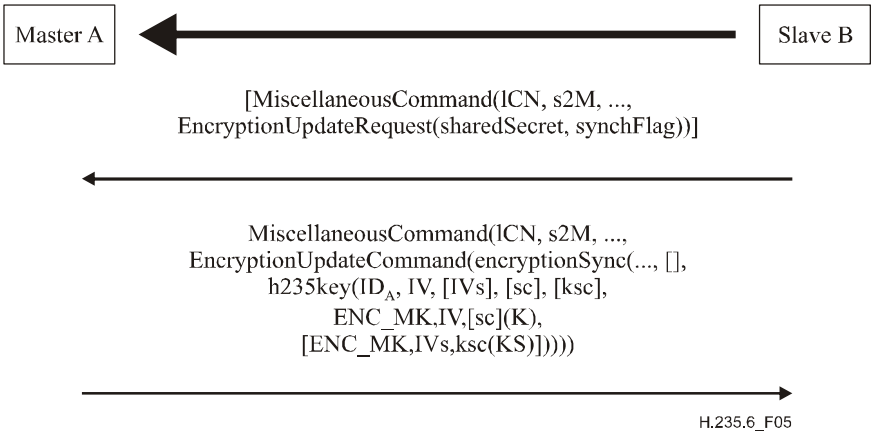


Figure 5 – Session key update on slave's logical channel

Figure 6 shows the key update procedures for a logical channel owned by the master. In case the slave initiates the key update and requests a new session key from the master, the slave shall send a **MiscellaneousCommand** to the master where **logicalChannelNumber** shall hold the logical channel number (as defined by the master), **sharedSecret** shall be set to true, the **direction** flag shall be set to **masterToSlave**. If, otherwise, the master initiates the key update, this **EncryptionUpdateRequest** message shall not be sent.

The master, either responding to the slave's request or on its own behalf, shall issue an **EncryptionUpdateCommand** where the **logicalChannelNumber** shall hold the logical channel number, **direction** shall be set to **masterToSlave**, **encryptionSync** shall provide the **synchFlag** with the new dynamic payload number. **h235key** shall carry the new session key. **h235key** shall hold the identity of the master in **generalID** and the applied initial vector *IV* in **paramS**. The encrypted media session key shall be conveyed within **encryptedSessionKey**, where the encryption function shall apply the master key and the initial value in **paramS** to the session key *K*. For EOFB, an unencrypted salting key is conveyed in **ClearSalt** within **paramS** (*sc*). For EOFB,

encryptedSaltingKey shall convey the encrypted media salting key, where the encryption function shall apply the master session key and the initial value **paramSsaltIV** to the salting key *KS*. For EOFB, an unencrypted salting key (*ksc*) is conveyed in **ClearSalt** within **paramSsalt**. **clearSaltingKey** may hold an unencrypted media salting key in which case **encryptedSaltingKey** shall remain empty and vice versa. The transmission of an unencrypted salting key shall only be achieved if the security does not suffer; in any other case, it is recommended that the media salting key be encrypted.

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

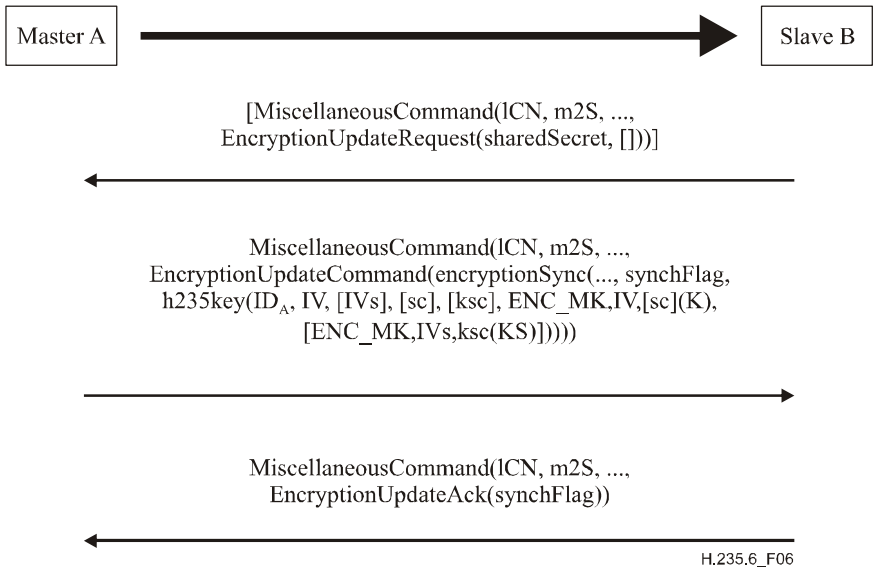


Figure 6 – Session key update on master's logical channel

8.6.3 Payload-type-based key update and synchronization

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

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

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

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

8.7 Non-terminal interactions

8.7.1 Gateway

As stated in clause 6.6 of [ITU-T H.235.0], an ITU-T H.323 gateway should be considered a trusted element. This includes protocol gateways (ITU-T H.323, ITU-T H.320, etc.) and security gateways (proxy/firewalls). The media privacy can be assured between the communicating end point and the

gateway device; but what occurs on the far side of the gateway should be considered insecure by default.

8.7.2 New keys

The procedures outlined in clause 8.5 of [ITU-T 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.

8.7.3 ITU-T 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 [ITU-T H.225.0] is secured *and* routed through the gatekeeper, it must also be trusted. If any of these ITU-T 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 ITU-T H.323-specific elements) may also be trusted, since they terminate connections, and may well have to manipulate the messages and media streams.

8.8 Multipoint procedures

8.8.1 Authentication

Authentication shall occur between an end point 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 clause 6.6 of [ITU-T H.235.0], the MC(U) is trusted; existing end points in a conference may be limited by the authentication level employed by the MC(U). New **ConferenceRequest**/**ConferenceResponse** commands allow end points to obtain the certificates of other participants in the conference from the MC(U). As outlined in ITU-T H.245 procedures, end points in a multipoint conference may request other end point certificates via the MC, but may not be able to perform direct cryptographic authentication within the ITU-T H.245 channel.

8.8.2 Privacy

MC(U) shall win all master/slave exchanges and, as such, it 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 end point 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.

9 Media stream encryption procedures

Media streams shall be encoded using the algorithm and key as presented in the ITU-T H.245 channel. Figures 7 and 8 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.

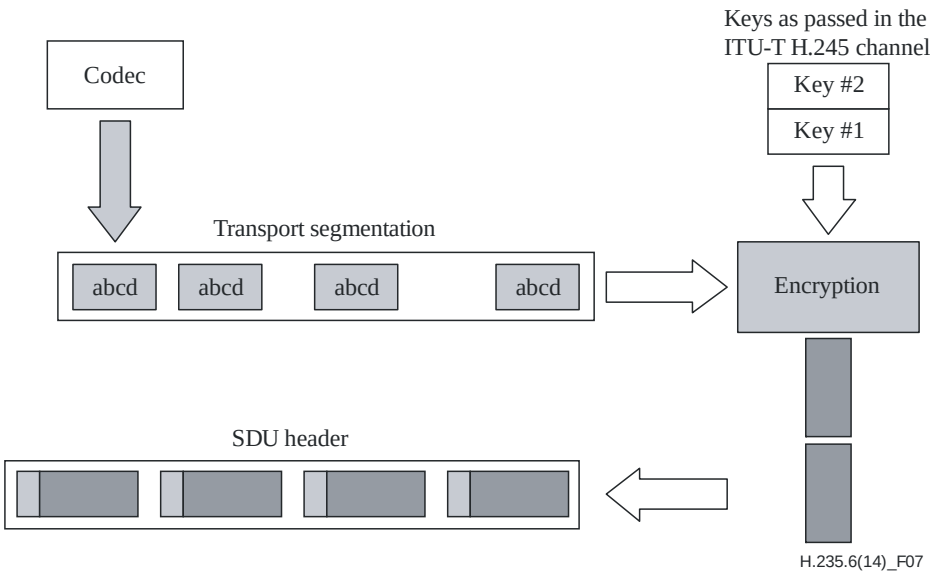


Figure 7 – Encryption of media

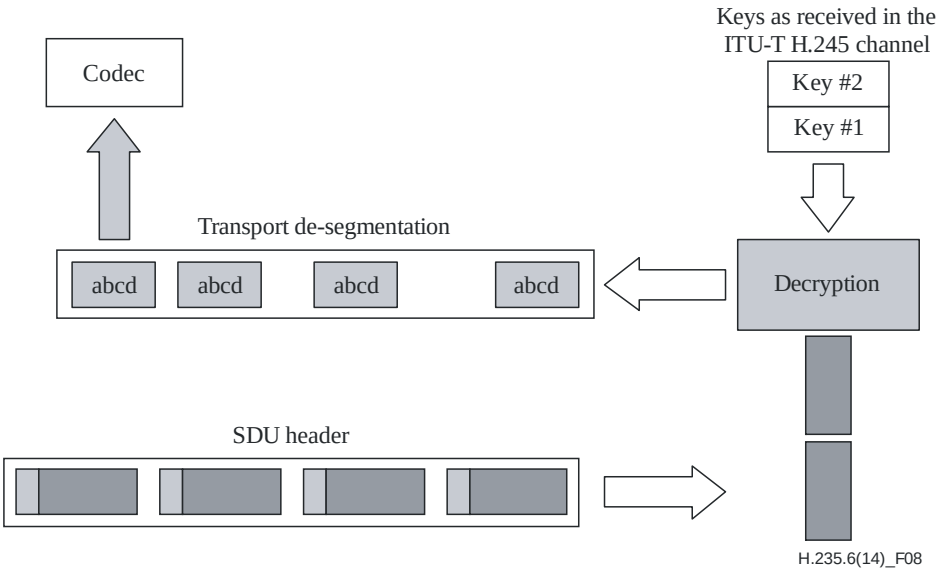


Figure 8 – Decryption of media

9.1 Media session keys

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

- If the ITU-T 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 ITU-T H.245 channel as a whole (i.e., outside ITU-T 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 ITU-T H.245 channel is not secure, but may also be used in addition to the secure ITU-T 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.

9.2 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 clause 9.2.1. The cryptographic algorithms are by:

- an encryption algorithm (e.g., DES in MAC mode, see [ISO/IEC 9797-1] and [ISO/IEC 9797-2]). DES-MAC is indicated using the OID "N" while Triple-DES-MAC is indicated using OID "O"; or
- using a cryptographic one-way function (e.g., SHA1). The OID to be used is "M".

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

Anti-spamming uses the following RTP packet format (see Figure 9) where the RTP padding sequence is interpreted as follows (see clause 5 of [b-IETF RFC 3550]).

- 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:

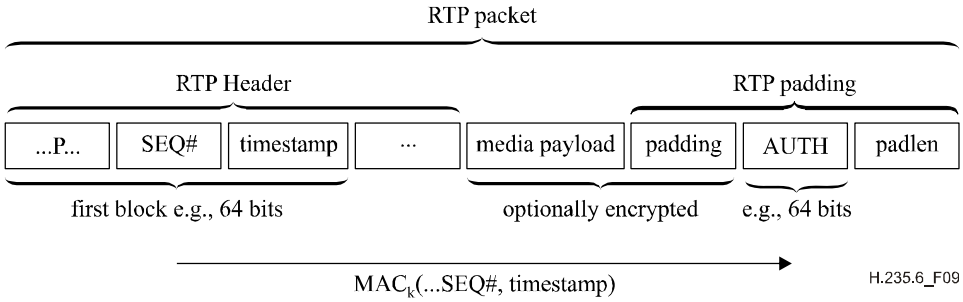


Figure 9 – RTP packet format for media anti-spamming

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 time stamp 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.235.0]. For larger block sizes (more than 64 bits), some sufficient additional bits of the RTP header, or even the first media payload, shall be taken.

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

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

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

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

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

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

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

9.2.1 List of object identifiers

Table 5 lists all the referenced OIDs.

Table 5 – 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 bits) MAC (see [ISO/IEC 9797-1] and [ISO/IEC 9797-2]) with 64-bit MAC
"O"	{iso(1) identified-organization(3) oiw(14) secsig(3) algorithm(2) desEDE(17)}	anti-spamming using Triple-DES (168-bits) MAC (see [ISO/IEC 9797-1] and [ISO/IEC 9797-2])

9.3 RTP/RTCP issues

The use of encryption on the RTP stream will follow the general methodology recommended in [b-IETF RFC 3550]. The encryption of the media shall occur in an independent, packet-by-packet basis.

NOTE – 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.

The RTP header shall not be encrypted. For audio/video codecs, the entire audio/video codec payload including any audio/video payload header(s) shall be encrypted. Synchronization of new keys and encrypted text is based upon dynamic payload type (see clause 8.6.3).

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

Deciphering the stream must be stateless due to the fact that packets may be lost, each packet should be deciphered independently. Two requirements of block algorithm mode shall operate as indicated in the following subclauses.

9.3.1 Initialization vectors

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

9.3.1.1 CBC initialization vector

An initialization vector (IV) is required when using a block cipher in CBC mode to encrypt RTP packet payloads. The size of an IV is the same as the block size for the particular block cipher. For example, the IV size for DES and 3-DES is 64 bits, while for AES it is 128, 192 or 256 bits.

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

9.3.1.2 EOFB initialization vector

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

Each RTP packet is associated with an implicit 48-bit packet index *i* as defined in [b-IETF RFC 3711] where $i = 2^{16} \times \text{ROC} + \text{SEQ}$ with SEQ the sequence number taken from the RTP header and ROC is the 32-bit rollover counter counting how often the sequence number SEQ has been wrapped around through 65535.

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

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

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

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

$i = 2^{16} \times v + \text{SEQ}$ where *v* is chosen from the set {ROC-1, ROC, ROC+1} modulo 2^{32} such that *v* is closest (in 2^{48} sense) to the value $2^{16} \times \text{ROC} + s_l$ where s_l is the maintained sequence number at the receiver. After the packet has been processed using the estimated index, the receiver shall decide if s_l and ROC should be updated. For instance, a simple (but not error robust) method is to simply set s_l to SEQ (if SEQ > s_l) and, if the value $v = \text{ROC} + 1$ was used, to update ROC to *v*; see also [b-IETF RFC 3711], section 3.2.1, for more information.

9.3.2 Padding

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

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

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

2) Padding in the manner prescribed by [b-IETF RFC 3550], section 5.1.

[b-IETF RFC 3550], section 5.1 describes a method of padding in which the payload shall be padded to a multiple of blocks. The last octet shall be set to the number of padding octets (including the last), and the *P* bit set in the RTP header. The value of the pad should be determined by the normal convention of the cipher algorithm.

All ITU-T 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.

9.3.3 RTCP protection

The application of cryptographic techniques to RTCP elements is for further study.

9.3.4 Secured payload stream

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

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

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

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

The **encryptionUpdateCommand** shall be used for the improved key update procedure to distribute new session key material (see clause 8.6.2). **multiplePayloadStream** is only used when a multiple payload stream is to be re-keyed, in which case the dynamic payload type within **EncryptionSync** shall be ignored.

9.3.5 Interworking with Recommendation ITU-T J.170

For further study.

9.4 Triple-DES in outer CBC mode

168-bit Triple-DES in outer CBC mode, as illustrated in Figure 10, 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 [b-IETF RFC 2405].

More information on Triple-DES may be obtained from [b-Schneier] and [b-IETF RFC 2405]. See also [b-DES FIPS 74] and [b-DES FIPS 81].

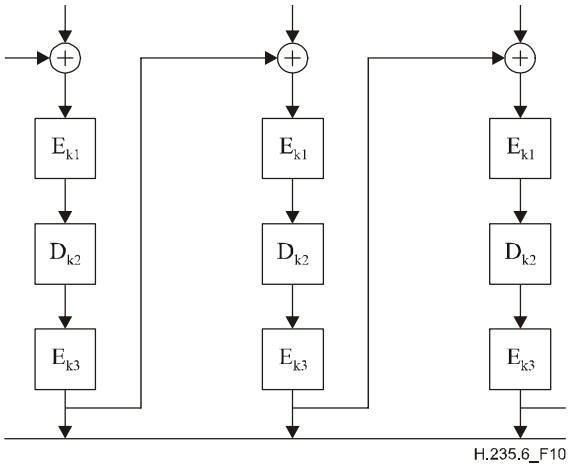


Figure 10 – Triple-DES encryption in outer CBC mode

9.5 DES algorithm operating in EOFB mode

Voice may be encrypted using the DES algorithm operating in the EOFB stream cipher block-chaining mode. EOFB mode allows exploiting parallelism in implementations. When operating in EOFB mode, it is recommended for both performance and security reasons, to feed back the entire crypto block (i.e., the full 64-bits for DES, for example, with $n = j = 64$). However, due to the fact that EOFB does not provide chaining across the blocks and bits, EOFB may be susceptible to specific attacks depending on the statistical properties of the input plaintext data. Thus, key updating (see clause 8.6) should be performed regularly but, at the latest, before the initial value wraps around. For the computation of the initial value see clause 9.3.1.2.

9.6 Triple-DES in outer EOFB mode

168-bit Triple-DES in outer EOFB mode, as illustrated in Figure 11, may be used within this security profile. In the figure, each k_i refers to a 56-bit key. A different 56-bit key *shall* be used within each encryption (E) and decryption (D) block. None of the 64 weak keys for DES are known to cause any weakness within Triple-DES. However, implementations complying with this profile should reject the key when a weak DES key is involved [b-IETF RFC 2405].

More information on Triple-DES may be obtained from [b-Schneier] and [b-IETF RFC 2405]. See also [b-DES FIPS 74] and [b-DES FIPS 81].

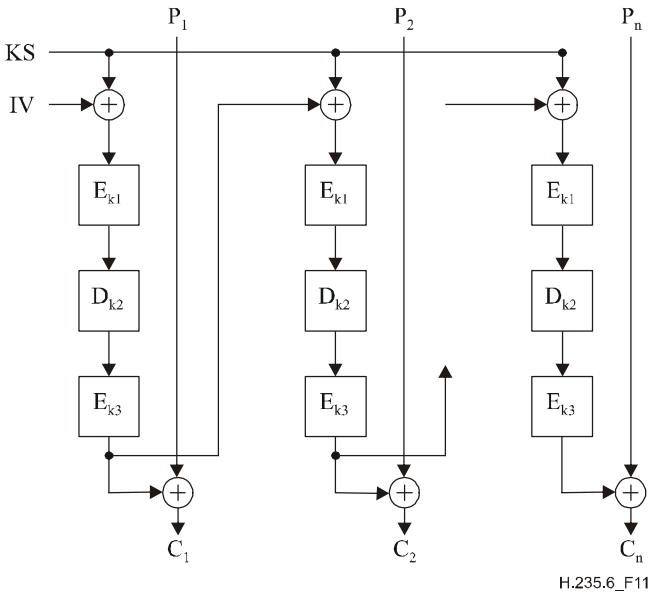


Figure 11 – Triple-DES encryption in outer EOFB mode

10 Lawful interception

For further study (see [b-ETSI TR 101 772]).

11 List of object identifiers

Table 6 lists all the referenced OIDs (see also [b-NIST SP 500-224] and [b-WEBOIDs]). There are object identifiers for [ITU-T H.235v1] and [ITU-T H.235v2].

Table 6 – Object identifiers

Object identifier reference	Object identifier value(s)	Description
"DHdummy"	{itu-t (0) recommendation (0) h (8) 235 version (0) 2 40} {itu-t (0) recommendation (0) h (8) 235 version (0) 3 40}	Non-standard DH-group explicitly provided
"DH1024"	{itu-t (0) recommendation (0) h (8) 235 version (0) 2 43} {itu-t (0) recommendation (0) h (8) 235 version (0) 3 43}	1024-bit DH group
"DH1536"	{itu-t (0) recommendation (0) h (8) 235 version (0) 3 44}	1536-bit DH group
"DH2048"	{itu-t (0) recommendation (0) h (8) 235 version (0) 3 45}	2048-bit DH group
"DH3072"	{itu-t (0) recommendation (0) h (8) 235 version (0) 3 46}	3072-bit DH group
"DH4096"	{itu-t (0) recommendation (0) h (8) 235 version (0) 3 47}	4096-bit DH group
"DH6144"	{itu-t (0) recommendation (0) h (8) 235 version (0) 4 77}	6144-bit DH group
"DH8192"	{itu-t (0) recommendation (0) h (8) 235 version (0) 4 78}	8192-bit DH group
"X"	{iso(1) member-body(2) us(840) rsadsi(113549) encryptionalgorithm(3) 2}	Encryption using RC2-compatible (56 bits) or RC2-compatible in CBC mode.
"X1"	{itu-t (0) recommendation (0) h (8) 235 version (0) 3 27}	Encryption using RC2-compatible (56 bits) or RC2-compatible in EOFB mode
"Y"	{iso(1) identified-organization(3) oiw(14) secsig(3) algorithm(2) descbc(7)}	Encryption using DES (56 bits) in CBC mode
"Y1"	{itu-t (0) recommendation (0) h (8) 235 version (0) 3 28}	Encryption using DES (56 bits) in EOFB mode with 64-bit feedback
"Z"	{iso(1) identified-organization(3) oiw(14) secsig(3) algorithm(2) desEDE(17)}	Encryption using Triple-DES (168 bits) in outer-CBC mode
"Z1"	{itu-t (0) recommendation (0) h (8) 235 version (0) 3 29}	Encryption using Triple-DES (168 bits) in outer-EOFB mode with 64-bit feedback
"Z2"	{itu-t (0) recommendation (0) h (8) 235 version (0) 3 30}	Encryption using AES (128 bits) in EOFB mode
"Z3"	{joint-iso-itu-t(2) country(16) us(840) organization(1) gov(101) csor(3) nistAlgorithm(4) aes128-cbc(1) cbc(2)}	Encryption using AES (128 bits) in CBC mode
"Z4"	{joint-iso-itu-t(2) country(16) us(840) organization(1) gov(101) csor(3) nistAlgorithm(4) aes(1) aes192-cbc(22)}	Encryption using AES (192 bits) in CBC mode
"Z5"	{joint-iso-itu-t(2) country(16) us(840) organization(1) gov(101) csor(3) nistAlgorithm(4) aes(1) aes256-cbc(42)}	Encryption using AES (256 bits) in CBC mode

Appendix I

ITU-T H.323 implementation details

(This appendix does not form an integral part of this Recommendation.)

I.1 Ciphertext padding methods

There is a description of ciphertext stealing in [b-Schneier], pages 191 and 196. Figures I.1 to I.5 illustrate the technique.

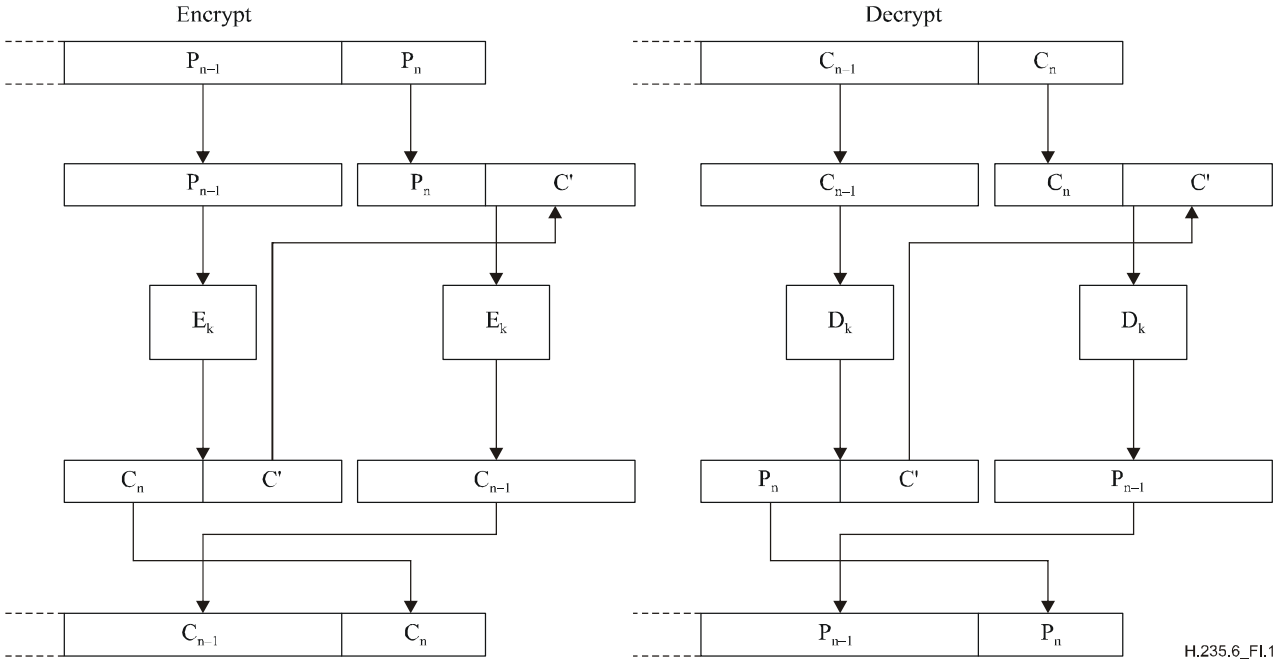


Figure I.1 – Ciphertext stealing in ECB mode

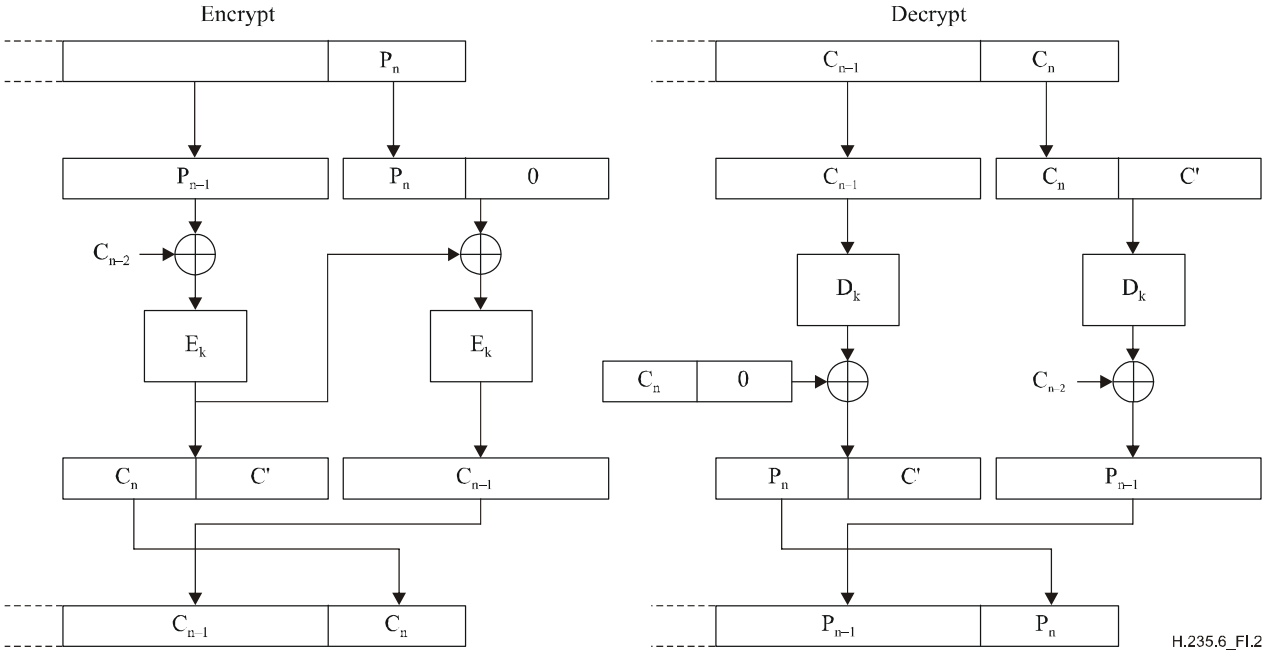


Figure I.2 – Ciphertext stealing in CBC mode

NOTE – Ciphertext stealing in ECB or CBC modes requires the payload to convey at least one complete block. Implementations deploying ciphertext stealing in ECB mode or CBC modes should ascertain that the payload conveys always at least one crypto block; e.g., by proper choice of the sampling/packetization rate or selection of the encryption algorithm.

In case the payload spans less than one single block, the initial vector (IV) shall be used as the previous ciphertext block when ciphertext stealing mode is applied in CBC mode.

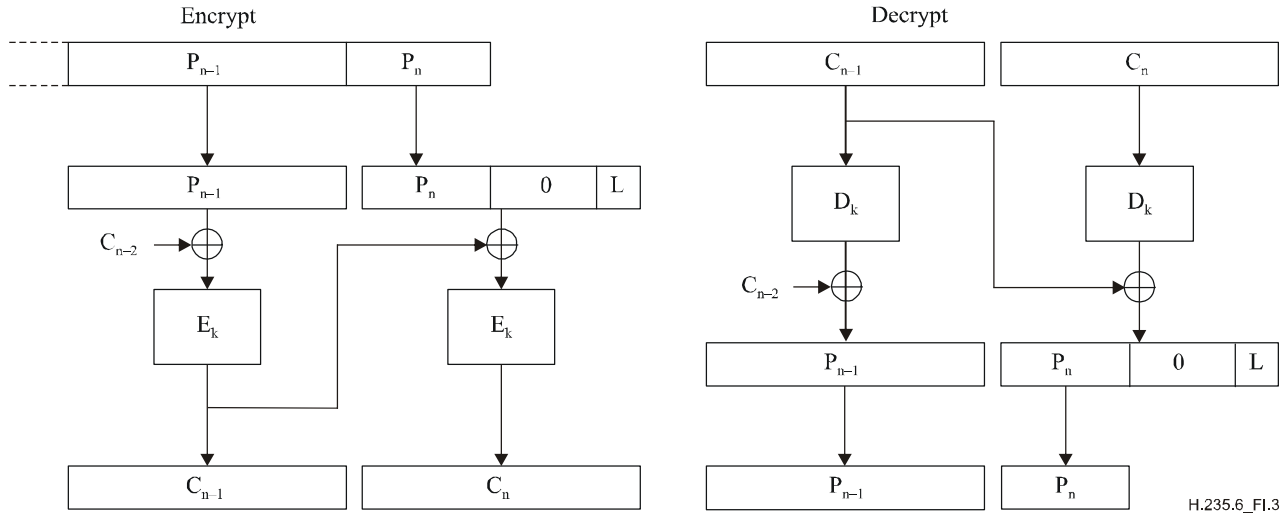


Figure I.3 – Zero padding in CBC mode

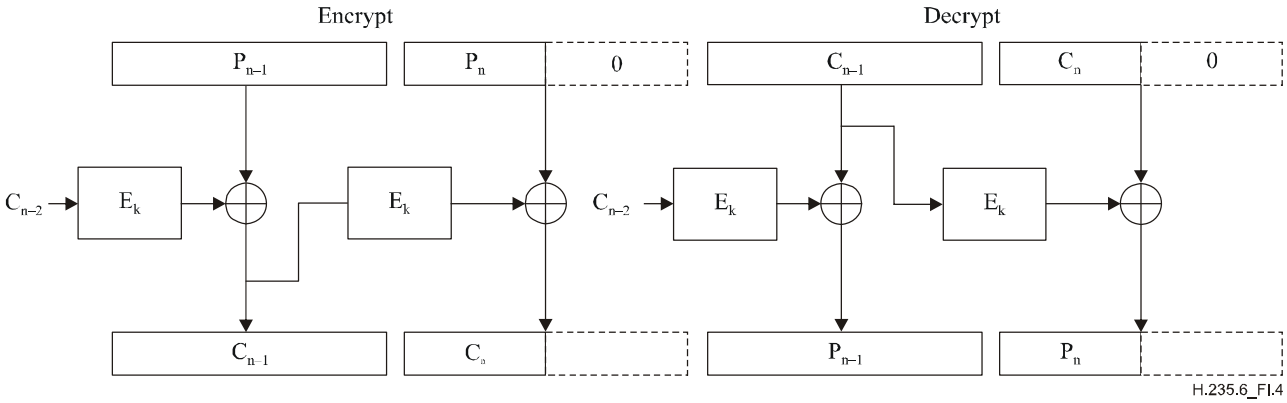
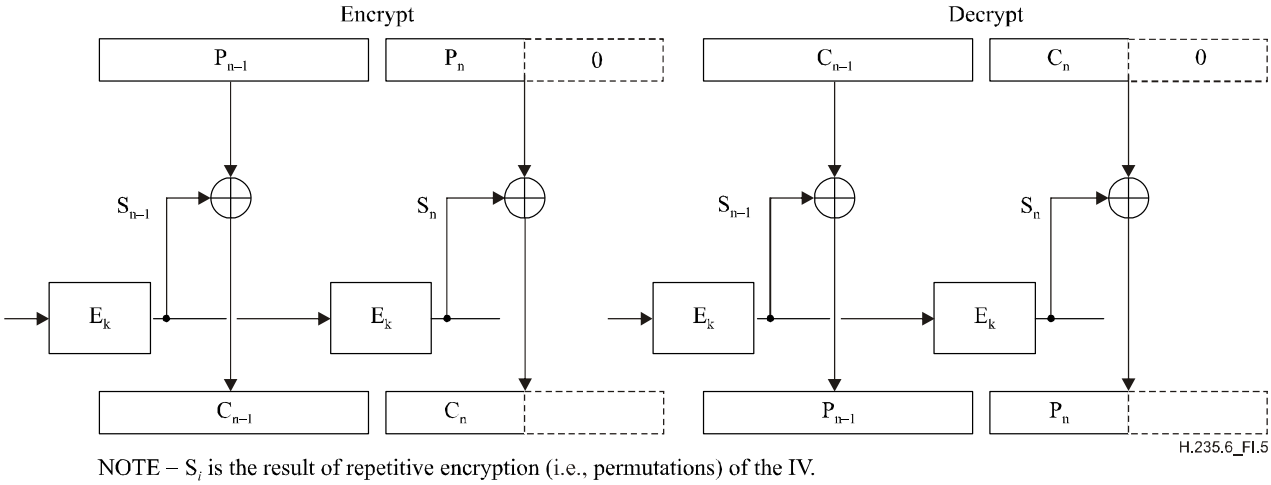


Figure I.4 – Zero padding in CFB mode



NOTE – S_i is the result of repetitive encryption (i.e., permutations) of the IV.

Figure I.5 – Zero padding in OFB mode

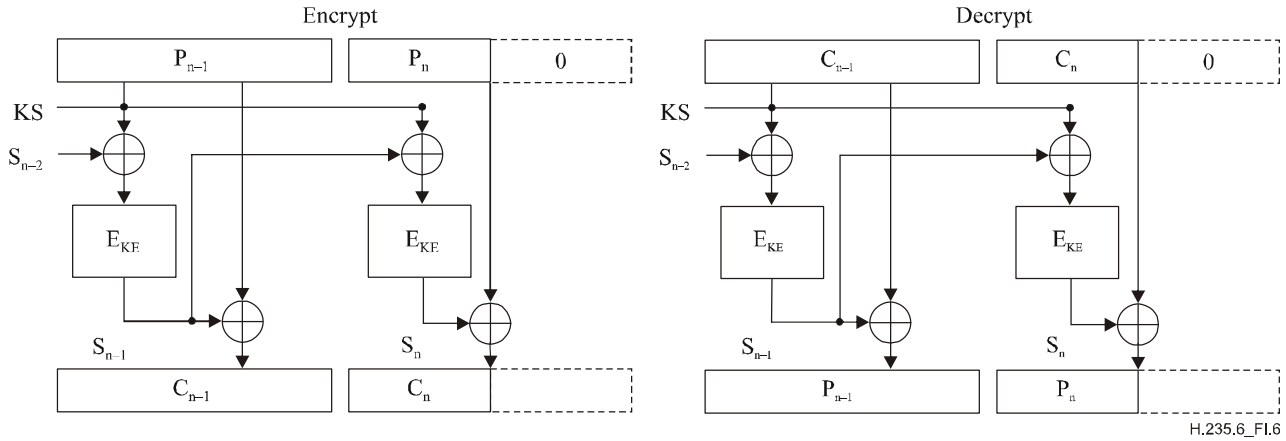


Figure I.6 – EOFB mode with zero padding

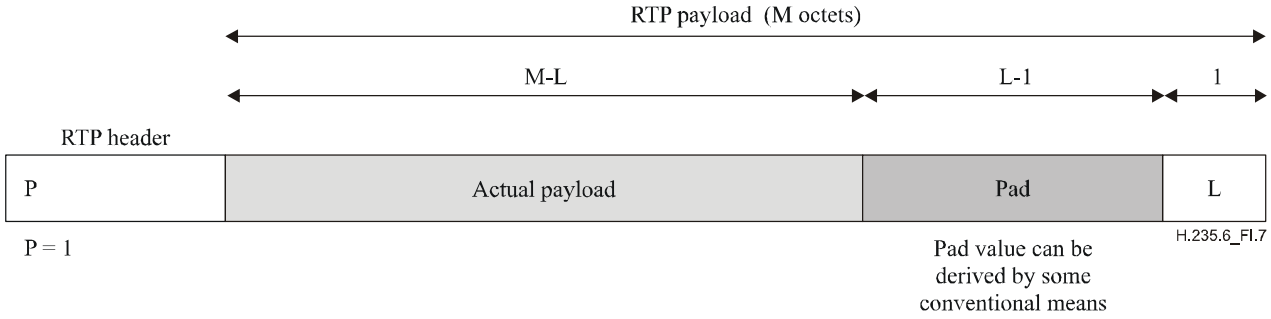


Figure I.7 – Padding as prescribed by RTP

I.2 New keys

The procedures outlined in clause 8.5 of [ITU-T 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.

Bibliography

[b-ITU-T J.170] Recommendation ITU-T J.170 (2005), *IPCablecom security specification*.

[b-ETSI TR 101 772] ETSI TR 101 772 V1.1.2 (2001), *Telecommunications and Internet Protocol Harmonization Over Networks (TIPHON) Release 3; Service independent requirements definition; Lawful interception – top level requirements.*

[b-IETF RFC 2268] IETF RFC 2268 (1998), *A Description of the RC2(r) Encryption Algorithm*.

[b-IETF RFC 2405] IETF RFC 2405 (1998), *The ESP DES-CBC Cipher Algorithm With Explicit IV*.

[b-IETF RFC 2406] IETF RFC 2406 (1998), *IP Encapsulating Security Payload (ESP).*

[b-IETF RFC 2408] IETF RFC 2408 (1998), *Internet Security Association and Key Management Protocol (ISAKMP).*

[b-IETF RFC 2409] IETF RFC 2409 (1998), *The Internet Key Exchange (IKE).*

[b-IETF RFC 2412] IETF RFC 2412 (1998), *The OAKLEY Key Determination Protocol*.

[b-IETF RFC 3526] IETF RFC 3526 (2003), *More Modular Exponential (MODP) Diffie-Hellman groups for Internet Key Exchange (IKE).*

[b-IETF RFC 3550] IETF RFC 3550 (2003), *RTP: A Transport Protocol for Real-Time Applications.*

[b-IETF RFC 3711] IETF RFC 3711 (2004), *The Secure Real-Time Transport Protocol (SRTP).*

[b-DES FIPS 46-3] NIST FIPS 46-3 (1999), *Data Encryption Standard* <http://csrc.nist.gov/publications/fips/fips46-3/fips46-3.pdf>

[b-DES FIPS 74] NIST FIPS 74 (1981), *Guidelines for Implementing and Using the Data Encryption Standard*.

[b-DES FIPS 81] NIST FIPS 81 (1980), *DES Modes of Operation, Federal Information Processing Standard*.

[b-FIPS PUB 180-4] NIST FIPS PUB 180-4 (2012), *Secure Hash Standard* <http://csrc.nist.gov/publications/fips/fips180-4/fips-180-4.pdf>.

[b-Daemon] Daemon, J.(1995), *Cipher and Hash function design*, Ph.D. Thesis, Katholieke Universiteit Leuven.

[b-NIST SP 500-224] NIST SP 500-224 (1994), *Stable Implementation Agreements for Open Systems Interconnection Protocols*, Version 8.

[b-WEBOIDs] OID assignments from the top node http://www.alvestrand.no/objectid/top.html.

[b-Schneier] Schneier B. (1995), *Applied Cryptography: Protocols, Algorithms, and Source Code in C*, 2nd Edition, John Wiley & Sons.

SERIES OF ITU-T RECOMMENDATIONS

Series A	Organization of the work of ITU-T
Series D	General tariff principles
Series E	Overall network operation, telephone service, service operation and human factors
Series F	Non-telephone telecommunication services
Series G	Transmission systems and media, digital systems and networks
Series H	**Audiovisual and multimedia systems**
Series I	Integrated services digital network
Series J	Cable networks and transmission of television, sound programme and other multimedia signals
Series K	Protection against interference
Series L	Construction, installation and protection of cables and other elements of outside plant
Series M	Telecommunication management, including TMN and network maintenance
Series N	Maintenance: international sound programme and television transmission circuits
Series O	Specifications of measuring equipment
Series P	Terminals and subjective and objective assessment methods
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, open system communications and security
Series Y	Global information infrastructure, Internet protocol aspects and next-generation networks
Series Z	Languages and general software aspects for telecommunication systems

Printed in Switzerland
Geneva, 2014