INTERNATIONAL TELECOMMUNICATION UNION

ITU-T

TELECOMMUNICATION
STANDARDIZATION SECTOR
OF ITU

H.235

Amendment 1

(04/2004)

SERIES H: AUDIOVISUAL AND MULTIMEDIA SYSTEMS

Infrastructure of audiovisual services – Systems aspects

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

Amendment 1

ITU-T Recommendation H.235 (2003) – Amendment 1

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 AND TRIPLE-PLAY MULTIMEDIA SERVICES	
Broadband multimedia services over VDSL	H.610–H.619

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

ITU-T Recommendation H.235

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

Amendment 1

Summary

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

This amendment extends version 3 of ITU-T Rec. H.235 by inclusion of new Annex H and by extending the functionality of Annex I. The ASN.1 changes are added in support of Annex H, they may be used by any other purpose as identified by the ClearToken **profileInfo**. This amendment also includes some corrections to and updates the ITU-T Rec. H.235 version 3 text.

Source

Amendment 1 to ITU-T Recommendation H.235 (2003) was approved on 6 April 2004 by ITU-T Study Group 16 (2001-2004) under the ITU-T Recommendation A.8 procedure.

CONTENTS

ITU-T Recommendation H.235

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

Amendment 1

…

2 References

…

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

– ITU-T Recommendation H.235 (2003~~0~~), *Security and encryption for H-series (H.323 and other H.245-based) multimedia terminals*.

…

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

– IETF RFC 3546 (2003), *Transport Layer Security(TLS) Extensions*.

…

3 Terms and definitions

…

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

3.8*bis* EC-GDSA: Elliptic curve digital signature with appendix analog of the NIST Digital Signature Algorithm (DSA) (see also ISO/IEC 15946-2, chapter 5).

3.8*ter* elliptic Curve Cryptosystem: A public-key cryptosystem (see section 8.7 of *ATM Forum Security Specification Version 1.1*).

3.8*quat* elliptic Curve Key Agreement Scheme – Diffie-Hellman: The Diffie-Hellman key agreement scheme using elliptic curve cryptography.

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

4 Symbols and abbreviations

This Recommendation uses the following abbreviations:

X || Y	Concatenation of X and Y
3DES	Triple DES
AES	Advanced Encryption Algorithm
ASN.1	Abstract Syntax Notation No. 1

BES	Back-end Server
CA	Certificate Authority
CBC	Cipher Block Chaining
CFB	Cipher Feedback mode
CRL	Certificate Revocation List
CTR	Counter Mode (see NIST 800-38A)
DES	Data Encryption Standard
DH	Diffie-Hellman
DNS	Domain Name System
DSS	Digital Signature Standard
DTMF	Dual Tone Multi-Frequency
ECB	Electronic Code Book
ECC and EC	Elliptic Curve Cryptosystem
EC-GDSA	Elliptic curve digital signature with appendix analog of the NIST Digital Signature Algorithm (DSA)
ECKAS-DH	Elliptic Curve Key Agreement Scheme – Diffie-Hellman
EOFB	Enhanced OFB mode
EP	Endpoint
GCF	Gatekeeper ConFirm
GK	Gatekeeper
GRJ	Gatekeeper ReJect
GRQ	Gatekeeper ReQuest
GW	Gateway
HMAC	Hashed Message Authentication Code
ICV	Integrity Check Value
ID	Identifier
IPSEC	Internet Protocol Security
ISAKMP	Internet Security Association Key Management Protocol
IV	Initialization Vector
LCF	Location ConFirm
LDAP	Lightweight Directory Access Protocol
LRJ	Location ReJect
LRQ	Location ReQuest
MAC	Message Authentication Code
MCU	Multipoint Control Unit
MD5	Message Digest 5
MIM	Man-In-the-Middle

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

5 Conventions

...

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

This Recommendation uses the notion of setting a compound ASN.1 data structure to NULL; for example, "**paramS** set to NULL" (see D.6.3.2, D.6.3.3.3, D.6.3.4.1, D.6.3.4.2, E.5, E.7, E.13.1 and E.13.2). This shall mean that all optional elements in the particular SEQUENCE (i.e., **Params**) are absent.

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

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

…

6 System introduction

6.1 Summary

1) The call signalling channel may be secured using TLS ([RFC 2246~~TLS~~], [RFC 3546]) or IPSEC ([RFC 2402~~IPSEC~~], [ESP]) on a secure well-known port (ITU-T Rec. H.225.0).

…

6.2 Authentication

…

As a third option, the authentication may be completed within the context of a separate security protocol such as TLS ([RFC 2246~~TLS~~], [RFC 3546]) or IKE~~IPSEC~~ [IKE~~PSEC~~].

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

…

6.9 Security profiles

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

…

Annex A

H.235 ASN.1

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

-- EXPORTS All

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

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

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

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

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

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

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

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

AuthenticationBES ::= CHOICE

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

}

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

}

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

```
}

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

ProfileElement     ::= SEQUENCE
{
     elementID     INTEGER (0..255), -- element identifier, as defined by
                                     -- profile
     paramS        Params OPTIONAL,  -- any element-specific parameters
     element       Element OPTIONAL,        -- value in required form
     …
}

Element ::= CHOICE
{
     octets              OCTET STRING,
     integer             INTEGER,
     bits                BIT STRING,
     name                BMPString,
     flag                BOOLEAN,
     …
}


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


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

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

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

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

Params ::= SEQUENCE {
     ranInt        INTEGER OPTIONAL, -- some integer value
     iv8           IV8 OPTIONAL, -- 8-octet initialization vector
     ...,
     iv16          IV16 OPTIONAL,-- 16-octet initialization vector
     iv            OCTET STRING OPTIONAL, -- arbitrary length initialization vector
```

```
    clearSalt     OCTET STRING OPTIONAL -- unencrypted salting key for encryption
}

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

CryptoToken::= CHOICE
{

    cryptoEncryptedToken SEQUENCE -- General purpose/application specific token
    {
        tokenOID      OBJECT IDENTIFIER,
        token         ENCRYPTED { EncodedGeneralToken }
    },
    cryptoSignedToken  SEQUENCE -- General purpose/application specific token
    {
        tokenOID      OBJECT IDENTIFIER,
        token         SIGNED { EncodedGeneralToken }
    },
    cryptoHashedToken SEQUENCE -- General purpose/application specific token
    {
        tokenOID           OBJECT IDENTIFIER,
        hashedVals         ClearToken,
        token HASHED { EncodedGeneralToken }
    },
    cryptoPwdEncr ENCRYPTED { EncodedPwdCertToken },
    ...
}

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

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

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

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

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



V3KeySyncMaterial  ::= SEQUENCE
{
     generalID              Identifier OPTIONAL, -- peer terminal ID
     algorithmOID           OBJECT IDENTIFIER OPTIONAL, -- encryption algorithm
     paramS                 Params, -- IV
     encryptedSessionKey    OCTET STRING OPTIONAL, -- encrypted session key
     encryptedSaltingKey    OCTET STRING OPTIONAL, -- encrypted media salting
                                                   -- key
     clearSaltingKey        OCTET STRING OPTIONAL, -- unencrypted media salting
                                                   -- key
     paramSsalt             Params OPTIONAL, -- IV (and clear salt) for salting
                                                   -- key encryption
     keyDerivationOID       OBJECT IDENTIFIER OPTIONAL, -- key derivation
                                                   -- method
     ...
}

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

…

The following Annex H is entirely new.

Annex H

Framework for secure authentication in RAS using weak shared secrets

Summary

This annex provides the framework for mutual party authentication during H.225.0 RAS exchanges. The "proof-of-possession" methods described permit secure use of shared secrets such as passwords which, if used by themselves, would not provide sufficient security.

Extensions to the framework to permit simultaneous negotiation of Transport Layer Security parameters for protection of a subsequent call signalling channel are also described.

Keywords

Authentication, Passwords, Security

H.1 Introduction

In many applications, an endpoint (or its user) and its gatekeeper may share only a "small" secret such as a password or a "personal identification number" (PIN). Such a secret (which we shall hereafter refer to as a "password"), and any encryption key derived from it, is cryptographically weak. The challenge/response authentication schemes as described in clause 10, provide samples of plaintext and corresponding ciphertext, and are therefore subject to a brute-force attack by an observer of the transaction when the authentications are keyed by simple passwords. Thus, the observer may recover the password or PIN and later pose as the endpoint to obtain service.

A family of protocols under the generic heading of Encrypted Key Exchange use a shared secret to "obscure" a Diffie-Hellman key exchange in such a way that the attacker must solve a series of finite logarithm problems in order to validate a brute-force attack against the shared secret. In the Encrypted Key Exchange (EKE) of Bellovin and Merritt [B&M], the shared secret is used to encrypt the Diffie-Hellman public keys under a symmetric algorithm. In the SPEKE method of Jablon [Jab], the shared secret is used to choose a different generator of the Diffie-Hellman group. These protocols combine the security of a strong Diffie-Hellman key exchange with use of the shared secret in such a way that an attacker cannot obtain known plaintext for use in a brute-force attack against the secret without solving the Diffie-Hellman finite logarithm problem. An advantage of such protocols is that they multiply the strengths of the Diffie-Hellman problem by the strength of the secret-key encryption (or vice versa). A potential disadvantage is that they are typically subject to patent protection.

H.2 Scope

This annex is usable by any gatekeeper or endpoint using the H.225.0 RAS protocols.

H.3 References

H.3.1 Normative references

The following ITU-T Recommendations and other references contain provisions which, through reference in this text, constitute provisions of this Recommendation. At the time of publication, the editions indicated were valid. All Recommendations and other references are subject to revision; 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

[H.323]	ITU-T Recommendation H.323 (2003), *Packet-based multimedia communications systems.*
[NIST 800-38A]	NIST Special Publication 800-38A 2001, Recommendation for Block Cipher Modes of Operation – Methods and Techniques. http://www.csrc.nist.gov/publications/nistpubs/800-38a/sp800-38a.pdf.

H.3.2 Informative references

[AES]	CHOWN (P.): Advanced Encryption Standard (AES) Ciphersuites for Transport Layer Security (TLS), RFC 3268, June 2002.
[B&M]	BELLOVIN (S.), MERRITT (M.): U.S. Patent 5,241,599, August 31, 1993, originally assigned to AT&T Bell Laboratories, now assigned to Lucent Technologies.
[Jab]	JABLON (D.): Strong Password-Only Authenticated Key Exchange, Computer Communication Review, ACM SIGCOMM, Vol. 26, No. 5, pp. 5-26, October 1996.

[NIST 800-57]	NIST Draft Special Publication 800-57, Recommendation on Key Management, Part 1: General Guideline, http://csrc.nist.gov/CryptoToolkit/kms/guideline-1-Jan03.pdf. http://csrc.nist.gov/CryptoToolkit/kms/.

H.4 Definitions

None.

H.5 Abbreviations

See clause 4.

H.6 Basic framework

H.6.1 Improved negotiation capabilities in H.235v3

Version 3 of ITU-T Rec. H.235 has been extended ([H235Amd.1]) to support this security framework via the addition of the following generic element to the **ClearToken**:

- **profileInfo** is a sequence of profile-specific elements, each identified by its own integer value as defined by the specific profile whose OID is carried in the **ClearToken.tokenOID**.

In the following descriptions, several elements are passed in **profileInfo**; each of these elements will be given a name, rather than an identifying value, for ease of discussion.

H.6.2 Use between endpoint and gatekeeper

The basic framework, in which the requestor is an endpoint wishing to register with a gatekeeper, and the responder is that gatekeeper, proceeds in a straightforward manner. In the following, it is implicitly assumed that each **ClearToken** mentioned is identified with the **tokenOID** of the authentication profile. The **ClearToken** is assumed to be extended. The **random** and/or **random2** elements may be used by a profile in either of two ways: they may be included in the computation of the authentication key, and/or they may be included in a profile **ClearToken** in each subsequent RAS message (e.g., RRQ/RCF) to prevent replay. The endpoint registration exchange proceeds as follows:

1) The endpoint announces its willingness to participate in one or more key negotiation and authentication schemes by including the appropriate object ID(s) for the desired profile(s) in **authenticationMechanism.keyExch** elements of the **authenticationCapability** element of the **G**atekeeper**R**e**Q**uest. It is assumed that each specific OID completely defines an authentication procedure in terms of public key system (e.g., Diffie-Hellman or Elliptic Curve) and specific group (e.g., one of the OAKLEY groups from RFC 2412), symmetric encryption algorithm (e.g., AES-128-CBC with ciphertext stealing), key derivation function (e.g., via the Pseudo-Random Function of Annex B), message authentication code (e.g., HMAC-SHA1-96), and the sequence in which they are used. The endpoint also includes one or more profile **ClearToken**s in the GRQ, each of which carries the OID for the specific profile offered and the necessary (encrypted) public key material in the following form:
    a) **tokenOID** carries the profile OID as offered in the **authenticationCapability** of the encapsulating GRQ.
    b) **timeStamp** may be used to assure currency and protect against replay.
    c) **password** shall not be used for the actual password.
    d) **dhkey** carries the Diffie-Hellman key parameters, if used. The enclosed **halfkey** element is encrypted as specified by the selected profile.

e) **challenge** is not required.

f) **random** is supplied by the initiating party and is used to prevent replay attacks.

g) **certificate** may be used if certificate exchange is part of the profile.

h) **generalID** may be used if required by the profile.

i) **eckasdhkey** carries the Elliptic Curve key parameters, if used by the profile. The enclosed **public-key** element should be encrypted as specified by the profile.

j) **sendersID** may be used as specified by the profile.

k) **profileInfo** element, **initVect**, may be supplied along with the (encrypted) public key material (**dhkey** or **eckasdhkey**) if the profile requires an initialization vector for decryption.

l) If the initiator wishes to use key material derived from an earlier exchange, it shall include a **profileInfo** element, denoted **sessionID**, containing the identifier assigned during the earlier exchange. In this case, **dhkey**, **eckasdhkey** and/or **initVect** should not be included.

m) If the initiator wishes to establish a TLS session for a call signalling connection, it may include one or more **profileInfo** elements containing TLS ciphersuites; the message shall contain only one cipher suite (the one previously negotiated) if **sessionID** is present.

n) If the initiator wishes to establish a TLS session for call signalling, it may include a **profileInfo** element containing a list of compression methods; only one compression method (the one previously negotiated) shall be included if **sessionID** is present.

o) More **profileInfo** elements may be used for any additional parameters required for the procedures under the profile.

2) Upon receiving the GRQ, the gatekeeper selects an **AuthenticationMechanism** profile from the offered list, generates a suitable private key, computes the corresponding public key, generates an initialization vector if needed for symmetric encryption using the password, encrypts the public key, generates a unique session ID, and generates a random quantity, all of which are encoded into a **ClearToken**. Depending on the profile, the following use is made of the ClearToken elements:

a) **tokenOID** carries the profile OID, as selected from the **authenticationMethod** of the encapsulating GCF.

b) **timeStamp** may be used to assure currency and protect against replay.

c) **password** shall not be used for the actual password.

d) **dhkey** carries the Diffie-Hellman key parameters, if used. The enclosed **halfkey** element is encrypted as specified by the selected profile.

e) **challenge** is used to carry an initialization vector, if required for key encryption as specified by the profile, or it may be used to carry a random string to be returned by the endpoint to prevent replay attacks.

f) **random** may contain the unpredictable, unique value supplied by the requestor to prevent replay attacks.

g) **certificate** may be used if certificate exchange is part of the profile.

h) **generalID** may be used if required by the profile.

i) **eckasdhkey** carries the Elliptic Curve key parameters, if used by the profile. The enclosed **public-key** element should be encrypted as specified by the profile.

j) **sendersID** may be used as specified by the profile.

k) **random** (or an additional **profileInfo** element, denoted **random2**, if the profile requires both random numbers to remain in the message exchange) should contain an unpredictable, unique value supplied by the responder to protect against replay attacks.

l) **initVect** is supplied along with the (encrypted) public key material (**dhkey** or **eckasdhkey**) if the profile requires an initialization vector for decryption.

m) **sessionID** is a unique (to the gatekeeper) identifier used to identify this registration session. Under certain profiles, it may also be used as a TLS session ID for rapid establishment of a TLS-protected call signalling channel.

n) **profileInfo** may be used for any additional parameters required for the procedures under the profile.

The gatekeeper then computes the shared secret or master key using its private key and the (decrypted) public key from the GCF, and derives from the master key the necessary encryption keys, authentication keys, or other material, according to the profile. The above-described **ClearToken** is placed within the **G**atekeeper**C**on**F**irm message. The GCF shall be integrity checked/authenticated using the derived authentication key, then sent to the endpoint. The authentication/integrity check may be returned in one of several ways, as specified by the profile: via a profile-specific **profileInfo** element, or via one of the procedures specified in Annex D.

3) The endpoint examines the selected **authenticationMechanism.keyExch** from the GCF and extracts the parameters from the **ClearToken** identified by the corresponding **tokenOID**. The endpoint then selects its private key, computes the corresponding public key, and selects any other parameters required by the profile. The endpoint then computes the shared secret or master key using its private key and the (decrypted) public key from the GCF, and derives the necessary encryption keys, authentication keys, or other material from it, according to the profile. The endpoint shall then verify the integrity of the GCF. If the GCF does not verify correctly, the endpoint shall discard it, along with all the key material derived from it, and continue waiting for a valid GRQ message. Standard RAS recovery will lead to a retransmission of the GRQ, and, presumably, receipt of an undamaged GCF. If a few retransmissions fail to produce a successful response, the endpoint should cease attempting to register and inform its user that something is amiss. Note that each GRQ sent gives a gateway imposter one more chance to guess the user's password and have its guess validated by acceptance of the GRQ. If the integrity check of the GCF succeeds, the endpoint has validated the gatekeeper, and may proceed to register, and, in the process, authenticate itself to the gatekeeper.

4) The endpoint then populates a **ClearToken** with the profile **tokenOID** in a manner similar to that done by the gatekeeper as described above. Any fields from the GCF clear token that are considered as a challenge by the profile should be included in the **ClearToken**. If specified by the profile to avoid replay, the **ClearToken** shall include **random** and **random2** from the GCF received above. The **ClearToken** is then placed in a **R**egistration **R**e**Q**uest to be sent back to the gatekeeper. The endpoint should then authenticate the full RRQ message and send it to the gatekeeper. From this point onward, the endpoint should not accept, nor should it send, RAS messages that are not authenticated by the agreed-upon profile using the authentication key derived from the shared key material.

5) The gatekeeper receives the RRQ, and shall use the shared key material to verify the integrity of the RRQ against the included authentication and integrity check. If the integrity check fails, the gatekeeper shall ignore the received RRQ, and wait for a verifiable RRQ. If none arrives, the endpoint will eventually abandon the registration attempt and return to the search for a gatekeeper. If the integrity check succeeds, the gatekeeper will prepare a Registration ConFirm message to send back to the endpoint. Depending on the profile, this RCF may contain a **ClearToken** that includes the **random**, **random2**, and/or **challenge** elements from the authentication profile **ClearToken** provided in the RRQ. The RCF, and all subsequent RAS messages, shall contain a verifiable authentication and integrity check computed using the negotiated authentication key and algorithm.

6) When the endpoint receives the RCF message, it verifies the integrity via the included authentication and integrity check element. If not verified, the RCF shall be discarded; if no valid RCF is received, even after the RRQ is retransmitted, then the session shall be abandoned and the endpoint shall return to seeking a new gatekeeper. If the RCF is verified, the session ID and selected ciphersuite, if present, may be extracted from its **ClearToken** for later use in the establishment of a secure call signalling channel.

H.6.3 Use of profile between gatekeepers

Essentially the same procedure may be used between gatekeepers in an LRQ/LCF exchange. In this situation, no explicit profile selection is possible; the originating gatekeeper shall offer one or more profiles by including the appropriate **ClearToken**(s) as described for the GRQ message, above. The responding gatekeeper may choose an offered profile and should return the corresponding **ClearToken** as described above for the GCF message. Note that, in this case, the calling gatekeeper does not authenticate itself to the responding gatekeeper until it establishes a call signalling channel to that gatekeeper.

This procedure may be employed in a multicast mode if a group of gatekeepers share a single secret to be used for this purpose. The multicast LRQ will be based on that secret; those gatekeepers that reply with LCF will use that key to decode the offered Diffie-Hellman public key, and will each choose their own **nonce** and Diffie-Hellman private key for their reply. The resulting session keys will be unique to the final pair of gatekeepers.

H.6.4 Signalling channel encryption and authentication

If gatekeeper routing is supported by the gatekeeper, the newly-negotiated master key material and identified cryptographic parameters may be used to authenticate and secure the call signalling channel, e.g., by establishing a TLS session for call signalling. If TLS is to be used, the gatekeeper shall include the selected **cipherSuite** and **compress** elements in the returned profile **ClearToken**.

H.7 A specific security profile (SP1)

This clause provides a standard security profile which is expected to provide a shared secret estimated to be equivalent to an 80-bit random number (see [NIST 800-57]). The profile consists of the following:

- Object ID for this profile (denoted "SP1") will be {itu-t (0) recommendation (0) h (8) 235 version (0) 3 60}.
- Master key, K_m, negotiation: Diffie-Hellman key exchange using the OAKLEY well-known group 2 [RFC 2412], followed by the SHA1 hash reduction of the Diffie-Hellman secret: K_m = SHA1(Diffie-Hellman shared secret).

- Symmetric encryption algorithm: shall be AES-128 in segmented counter mode with a 2-octet party discriminator, D, a 124-octet initialization vector, IV, and a 2-octet counter field, C, such that counter = D || IV || C, and C = 0 initially. See [NIST 800-38A] for a description of CTR mode. The party discriminator, D, is set to 0x3636 when the IV is generated by the party which issued the GRQ/RRQ, or LRQ, and is set to 0x5c5c when the IV is generated by the party which responded with GCF/RCF, or LCF. Each party must insure that each IV it generates is unique; it may use its own method to insure this uniqueness.
- Diffie-Hellman key encryption: shall use the AES-128 segmented counter mode to encrypt the Diffie-Hellman public key (represented as an octet string in network byte order); the initialization vector shall be carried in **ClearToken.initVect,** and the 16-octet key, K_p, shall be constructed as the high-order 128 bits of the SHA1 hash of the user password: K_p = Trunc(SHA1(user password), 16), where Trunc(x,y) truncates octet string x to y octets. Note that this is typically considered a weak key.
- Replay prevention: each party shall supply a 32-bit "random" number (which may contain a counter field to guarantee uniqueness); the random numbers are used explicitly in the computation of the derived keys, hence they each need only be transmitted once.
- Authentication key, K_a, derivation: using the PRF of Annex B, which we denote as PRF(*in_key*, *label*, *outkey_len*) with *in_key* = K_m, and *label* = "auth_key" || R_e || R_g, where R_e is a **nonce** obtained from **ProfileElement** of the GRQ and R_g is a **nonce** obtained from a **ProfileElement** of the GCF, and *outkey_len* = 128.
- Message authentication and integrity function: using a **ClearToken** with **tokenOID** set to "SP1" and a **ProfileElement.octets** set to the HMAC-SHA1-96 hash value computed over the entire message as described in ITU-T Rec. H.225.0; this procedure shall be applied to all RAS and call signalling messages (except a GRQ, or LRQ, which does not contain a **sessionID**).
- Element encryption key, K_e: selected elements of call signalling messages (or elements tunnelled therein) may be encrypted using AES-128 in segmented counter mode using key K_e = PRF(K_m, "encrypt_key" || R_e || R_g, 128). For example, this key may be used to encrypt media session keys for distribution in **h235Key** elements as used in Fast Connect and/or H.245. When used in this manner, "SP1" is used as the encryption algorithm OID.

This profile makes use of the **ProfileElement**s defined in Table H.1. These elements are carried in a **ClearToken.profileInfo** element sequence as defined in Amendment 1 to Annex A/H.235.

Table H.1/H.235 – Profile elements

Element name (used in text)	ElementID Value	Element choice (length)	Element description
initVect	1	**Octets (12)**	initialization vector for EKE encryption
nonce	2	**Octets (any)**	an unpredictable, unique value
cipherSuite	3	**Octets (2)**	a TLS ciphersuite
compression	4	**Octets (1)**	a TLS compression algorithm
sessionID	5	**Octets (1..)**	unique, may match a TLS session ID
integrityCheck	6	**Octets (12)**	keyed checkvalue

The registration sequence shall consist of:

– The endpoint shall send GRQ with the **authenticationCapability** element containing an **AuthenticationMechanism.keyExch** containing OID "SP1" and a corresponding **ClearToken** with **tokenID** = "SP1" and **dhkey** containing a 1024-bit public key encrypted using **initVect** as the IV and the key derived from the user password, and **nonce** = a 32-bit random number selected by the endpoint.

– The gatekeeper shall reply with GCF with the **authenticationMode** element equal to an **AuthenticationMechanism.keyExch** containing OID "SP1", and a **ClearToken** with **tokenID** = "SP1" and **dhkey** containing an unencrypted 1024-bit public key, and **nonce** = a 32-bit random number selected by the gatekeeper, along with an **integrityCheck** containing the authentication hash value computed using the derived authentication key, K_a. Note that it is not necessary for the gatekeeper to encrypt its Diffie-Hellman half-key in the GCF in this profile because it is the first party to authenticate itself by demonstrating its ability to authenticate the GCF using the derived authentication key. This mode permits the gatekeeper to reuse its Diffie-Hellman keys with more than one endpoint. See clause H.9.5.

– The endpoint shall reply with an RRQ with the authentication and integrity check value in a **ProfileElement** with **elementID** set to **integrityCheck**, and **element** set to the value computed using the derived authentication key, K_a.

– Subsequent RAS messages, including the RCF, shall be authenticated and integrity checked using the same procedure and key. H.225.0 Call Signalling messages (and tunnelled H.245 messages, if present) shall be authenticated using a **ClearToken**, with **tokenOID** set to "SP1", containing a **profileInfo ProfileElement** with **elementID** set to **integrityCheck** and **element** set to the computed value.

– The encryption key, K_e, and the encryption algorithm AES-128 in segmented counter mode may be used by the gatekeeper and the endpoint to encrypt selected information transported over RAS, call signalling, and/or H.245. For example, the gatekeeper may distribute media encryption keys secured under K_e and the profile encryption algorithm.

– If an endpoint is required to reregister, and it retains the original session ID and master secret, it should attempt to reregister using the original session ID and master secret by including the session ID explicitly in the GRQ (and not including a Diffie-Hellman half key) in its GRQ.

– This profile shall be usable between gatekeepers (see H.6.3).

H.8 Extensions to the framework (Informative)

The following elements may be incorporated into a security profile defined under this framework.

H.8.1 Using the master key to secure the call signalling channel via TLS

The key material negotiated during the RAS exchange may be used also to derive session keys for the protection of the call signalling channel under the TLS transport protocol ([RFC 2246], [RFC 3546]). In effect, the RAS negotiation replaces the initial TLS handshake protocol. This only makes sense, of course, if the call signalling will be gatekeeper routed. This is especially useful for intergatekeeper authentication and signalling using the LRQ/LCF exchange. In this case, there is no third RAS message by which the calling gatekeeper can authenticate itself to the called gatekeeper using the negotiated key material, but the caller can be implicitly authenticated by its ability to establish the call signalling channel with the correct TLS session parameters. Figure H.1 illustrates the flow of information involved: RAS is used to negotiate the session master key, the Session ID and the corresponding pre-master secret are distributed to the TLS software, and the Session ID is used by the call signalling layer to establish the call signalling channel over TLS. The means by which transfer of the secret is accomplished is implementation-dependent and beyond the scope of this Recommendation. Note that this Recommendation specifies port 1300 as the default TLS listen port for call signalling. The endpoint must, however, use one of the call signalling transport addresses supplied by the gatekeeper.

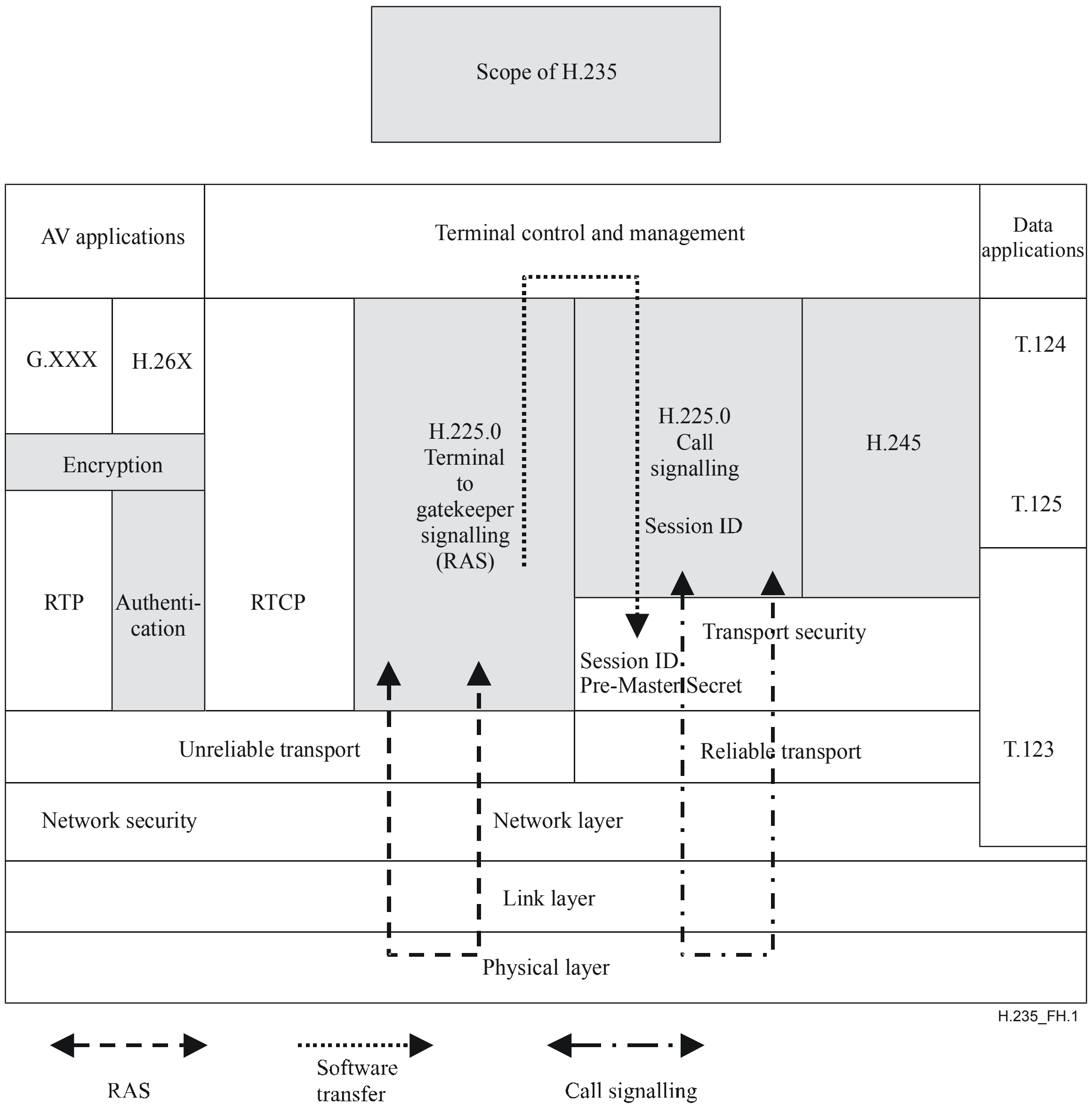


Figure H.1/H.235 – Information flow for security profile and TLS

The following description makes reference to the steps of the basic framework in Figure H.1.

H.8.1.1 Endpoint registration

An endpoint may test the ability of a gatekeeper to support TLS-protected call signalling by including one or more **cipherSuite** elements and one or more **compression** elements in the profile **ClearToken** in the GRQ message sent in step 1, above. If the endpoint wishes to use a previously-negotiated session, it shall also include the **sessionID** in the **ClearToken** (and shall specify only the single ciphersuite and the single compression method which match the requested session). If the negotiation is to be based on an existing TLS session, no cryptographic material is required in the profile **ClearToken**, other than **nonce**.

If a requested session does not exist, the gatekeeper shall select another authentication profile (if offered) or it shall return a GRJ with **GatekeeperRejectReason.resourceUnavailable**. If the requested session does exist, the master key material is obtained from the TLS session, and used (along with **random** from the GRQ and a gatekeeper-generated **random2**) to compute the authentication key for the RAS exchange. The **sessionID**, the **cipherSuite**, the **compress** method, and the gatekeeper's **nonce** shall be returned in the profile **ClearToken** of a GCF.

If the gatekeeper can support TLS session negotiation, it shall compute the master key material as specified by the profile, assign a new session ID and return it in the profile **ClearToken** in the **sessionID**. The profile **ClearToken** shall also contain the required security parameters from step 2, above, along with a single selected **cipherSuite**, a single selected **compress** method, and the non-zero **sessionID**. Note that the key exchange method of the selected ciphersuite is immaterial. If the gatekeeper agrees to TLS protection of call signalling, all call signalling transport addresses exchanged in the subsequent RRQ/RCF or ARQ/ACF messages shall be TLS-enabled.

If TLS negotiation and/or gatekeeper routing are not supported by the gatekeeper, then no TLS parameters shall be returned, but the authentication procedures may continue from step 3, as described above. The endpoint shall decide if it is prepared to proceed without TLS protection of the call signalling; it may choose to do so and still make use of the authentication profile. Upon successful completion of the registration sequence, the TLS session is available for use in rapid establishment of one or more call signalling connections to the gatekeeper, without the need to renegotiate keying material via public-key methods.

TLS sessions have finite lifetimes. Therefore, it may become necessary for an endpoint to renegotiate session parameters and obtain a new session ID. This may be accomplished by exchanging the necessary **ClearToken** elements as described above in a lightweight ("keepalive") registration sequence. Such a sequence shall not affect the RAS authentication key.

H.8.2 Use of certificates to authenticate the gatekeeper

Although it may be impractical to exchange verifiable certificate chains in RAS (due to UDP packet size limitations), it is possible to have a server authenticate itself to the endpoint if the endpoint can obtain a trusted copy of the server's public key via some other means. The server can simply include, in the GCF message, a **CryptoH323Token.cryptoGKCert** with the **ClearToken.tokenOID** set to the selected security profile OID.

H.8.3 Use of alternative signalling security mechanisms

The parameters negotiated as part of a security profile under this annex may be employed in transport and/or application level security mechanisms as determined by the specific profile. The **profileInfo** sequence added to the H.235 **ClearToken** has been provided for such use, if needed.

H.9 Threats (Informative)

H.9.1 Passive attack

At the present time, the scheme described above is not vulnerable to passive attack, subject to the provision that the Diffie-Hellman negotiation is not vulnerable to a passive attack.

H.9.2 Denial-of-Service attacks

This scheme is subject to an active Denial-of-Service attack in which a third party responds to the initial GRQ with a spurious GRJ. This type of attack may, or may not, be identifiable: if the rejecting gatekeeper is legitimate, and knows the shared secret (e.g., the gatekeeper is the endpoint's gatekeeper and the **rejectReason** is **resourceUnavailable**), then the gatekeeper could complete the key negotiation and authenticate the GRJ by returning, in the GRJ, the same elements described for the GCF (with the exception that the OID returned in **authenticationMode** of the GCF would be returned in a **ClearToken.profileInfo** element of the GRJ). This is left as part of the definition of a specific profile.

If the GRJ is not authenticated, it could be from an attacker. Before acting on the GRJ (e.g., looking for an alternate gatekeeper), the endpoint should wait for possible receipt of another GRJ or an authenticated GCF from the proper gatekeeper. Otherwise, the endpoint should try each gatekeeper suggested in any **altGKInfo** received in all GRJs (one of which is, presumably, legitimate). In any case, only the proper gatekeeper (which knows the shared secret) can return an authenticated GCF.

H.9.3 Man-in-the-Middle attacks

It is tempting to consider the exchange using an unencrypted Diffie-Hellman key exchange, followed by use of the password or PIN to derive session keys from the Diffie-Hellman secret. However, this form of the exchange is subject to a Man-in-the-Middle attack that can be used to discover the "small" shared secret by brute force using the Integrity Check Value provided by the legitimate gatekeeper in the GCF message.

Any MIM can, of course, manipulate any authenticated RAS message to insure that the message will be discarded due to integrity check failure. If all messages can be manipulated, service can be denied.

H.9.4 Guessing attacks

An attacker may pose as either a legitimate endpoint, a legitimate gatekeeper, or both (Man-in-the-Middle), and attempt to guess the shared secret by trial and error. For example, the attacker (who is presumed to know the details of the authentication profile, but not the shared secret) can guess a shared secret and attempt to register by sending a GRQ using this guess. In general, the gatekeeper will respond to this attempt with a GCF containing the GK's public key (encrypted using the real shared secret), and an ICV computed using the derived key which depends on the GK's decryption of the attacker's encrypted public key. The attacker can use this information to verify its guess of the shared secret. If the guess confirms the ICV of the GCF, then it is likely equal to the actual shared secret; this can be confirmed by continuing with the registration sequence. If the guess cannot be used to reproduce the ICV of the GCF, then the attacker must make another guess and try again. With a small keyspace for the shared secret, the number of guesses for a brute-force search may not be prohibitive. This attack requires the active participation of the gatekeeper (or the endpoint if the attacker is posing as the gatekeeper). The traditional method for countering such an attack is to monitor the number of unsuccessful attempts and, when a threshold is reached, treat all subsequent attempts as invalid (at least for a specified period) and raise an alarm, but such procedures are implementation dependent.

H.9.5 Unencrypted gatekeeper half-key

As mentioned above, the EKE exchange may remain secure, under certain conditions, if the responding gatekeeper does not encrypt its Diffie-Hellman half-key. In particular, the gatekeeper must be the first party to demonstrate its knowledge of the shared secret (PIN) via the ICV. If this is not the case, then the gatekeeper (or an interloper posing as the gatekeeper) could simply try all possible PINs to decrypt the endpoint's D-H half-key, compute the resulting D-H shared secret, derive the authentication key, and test it against the ICV supplied by endpoint. This is not possible if the endpoint can check the ICV supplied by the gatekeeper first, and refuse to continue with registration if the ICV is not as expected.

The use of an unencrypted half-key is advantageous to the gatekeeper in that it can reuse its corresponding private key with multiple endpoints. This would not be possible if the same key were distributed encrypted under multiple shared secrets or PINs. A third-party observer could collect examples of the half-key encrypted under two different PINs, say, then could search through the possible combinations of two PINs to see which pair produced the same half-key when decrypted. If there are, say, 10^8 possible PINs, then there are only 10^{16} possible combinations to try. This is a problem equivalent to searching for a 54-bit random number, which is not at all infeasible. Even if more than one possible solution is found, the correct one could be quickly determined by use of a third observation.

Annex I

Support of direct-routed calls

…

I.5 Symbols and abbreviations

This annex uses the following abbreviations:

ENC$_{K; S, IV}$(~~{~~M)~~}$_{K; S, IV}$~~ EOFB Encryption of *M* using secret key *K* and secret salting key *S* and initial vector *IV*

CT	ClearToken
DRC	Direct-Routed Call
EPID	Endpoint Identifier
GKID	Gatekeeper Identifier
K_{AG}	Shared secret (Annex D, Annex F) between EP A and GK G
K_{BH}~~$_{G}$~~	Shared secret (Annex D, Annex F) between EP B and GK H~~G~~
K_{GH}	Shared, secret (Annex D, Annex F) between GK G and GK H
KS_{AG}	Secret, shared salting key between EP A and GK G
KS_{BH}~~$_{G}$~~	Secret, shared salting key between EP B and GK H~~G~~
EK~~'~~$_{AG}$	The encryption key shared between EP A and GK G.
EK~~'~~$_{BH}$~~$_{G}$~~	The encryption key shared between EP B and GK H~~G~~.
K_{AB}	The encryption key shared between EP A and EP B.

I.6 Normative references

…

– ITU-T Recommendation H.235 Annex F (2003~~2~~), Corrigendum 1 to *Hybrid security profile*.

…

I.7 Overview

The Annex D baseline (see the main body of this Recommendation) as well as the Annex F hybrid security profile (see Annex F) (after the first handshake) apply a shared secret to assure message authentication and/or integrity in a hop-by-hop fashion using the gatekeeper as a trusted intermediate host. Using the direct-routed call model, a shared secret between two endpoints cannot be assumed. It is also not practical to use a pre-established shared secret to secure the communication since in this case all endpoints would have to know in advance which other endpoint will be called.

This annex addresses a scenario as shown in Figure I.1 where endpoints are attached to a ~~single~~ gatekeeper, and deploy direct-routed call signalling. The scenario assumes an unsecured IP network in the gatekeeper zone.

It is assumed that each endpoint has a communication relation and a security association with its~~the~~ gatekeeper, and that each endpoint has registered securely with the gatekeeper using either the baseline or the hybrid security profile.

Hence, the gatekeeper of the initiating endpoint is able to provide a shared secret for the directly communicating endpoints using a Kerberos-like approach.

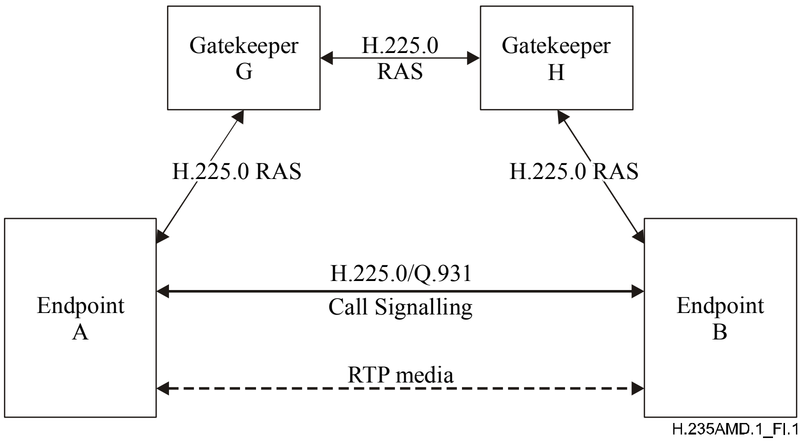


Figure I.1/H.235 – Direct-routed call scenario

I.8 Limitations

T~~his annex currently does not address direct-routed scenarios where endpoints are attached to distinct gatekeepers. Further, t~~his annex does not address direct-routed scenarios without any gatekeeper. This is ~~all~~ left as for further study.

I.9 Procedure DRC

Endpoints capable of supporting this security profile shall indicate this fact during **GRQ** and/or **RRQ** by including a separate ClearToken with **tokenOID** set to "I0"; any other fields in that ClearToken should not be used. The Annex I-capable gatekeeper that is willing to provide this functionality shall reply with **GCF** resp. **RCF** with a separate ClearToken included with **tokenOID** set to "I0" and all other fields in the ClearToken unused.

Before an endpoint A starts sending call signalling messages to another endpoint B directly, the endpoint A or B shall apply for admission at the gatekeeper G or H using **ARQ**. Endpoint A shall include within **ARQ** a separate ClearToken with **tokenOID** set to "I0" and all other fields in the ClearToken unused.

This procedure covers the case of both a single, common gatekeeper to the endpoints and the case of multiple, chained gatekeepers. In case of multiple involved gatekeepers, gatekeeper G – in which zone the call originates – should locate gatekeeper H using the (multicast) **LRQ** mechanism as described in 8.1.6/H.323, "Optional called endpoint signalling". The communication between two gatekeepers shall be secured according to Annex D. For this, it is assumed that a common shared secret K_{GH} is available. Since **LRQ** among gatekeepers is typically a multicast message, the shared secret K_{GH} typically cannot be a pair-wise shared secret but is assumed to be actually a group-based shared secret within the potential cloud of gatekeepers.

NOTE – This assumption limits scalability in the general case, and does not allow source authentication. However, it is believed that in corporate networks with a limited, small number of well-known gatekeepers such constraint and security limitations are still acceptable. Securing intergatekeeper multicast communication using digital signatures could overcome those limitations; yet this is left as for further study.

If the **LRQ** mechanism is used to locate the far-end gatekeeper, then **LRQ** shall convey a separate ClearToken with **tokenOID** set to "I0"; any other fields in that ClearToken should not be used. For the multicast case, the **generalID** in the CryptoToken of **LRQ** shall not be used. Intergatekeeper communication using H.501 and/or H.510 are left as for further study.

EK_{BH} denotes the encryption key that is shared between endpoint B and gatekeeper H. Gatekeeper H shall generate encryption key material EK_{BH} from the shared secret K_{BH} using the PRF-**based** key derivation procedure as defined in I.10 where **keyDerivationOID** in **V3KeySyncMaterial** shall hold "AnnexI-HMAC-SHA1-PRF"; see I.12.

Gatekeeper H shall transmit the encrypted EK_{BH} to gatekeeper G. The enhanced OFB (EOFB) encryption mode (see B.2.5) shall be used with the secret, endpoint-specific salting key KS_{GH}. Applicable encryption algorithms are (see D.11):

- DES (56 bits) in EOFB mode using OID "Y1": optional;
- 3DES (168 bits) in outer-EOFB mode using OID "Z1": optional;
- AES (128 bits) in EOFB mode using OID "Z2": default and recommended;
- RC2-compatible (56 bits) in EOFB mode using OID "X1": optional.

For the EOFB encryption mode, GK H shall generate a random initial value IV. For OID "X1", OID "Y1" and OID "Z1" the IV has 64 bits and shall be conveyed within **iv8** of **params** within **V3KeySyncMaterial;** whereas the IV has 128 bits for OID "Z2" and shall be conveyed within **iv16** of **params** within **V3KeySyncMaterial**.

Gatekeeper H shall include $ENC_{K_{GH}, KS_{GH}, IV}(EK_{BH})$ in ClearToken CT_{HG} with **tokenOID** set to "I3". The obtained ciphertext $ENC_{K_{GH}, KS_{GH}, IV}(EK_{BH})$ shall be conveyed in the **h235key** data structure as part of **secureSharedSecret** where it shall be placed within the **encryptedSessionKey** of the **secureSharedSecret** data structure. The encryption algorithm shall be indicated in **algorithmOID** ("X1", "Y1", "Z1" or "Z2") within **V3KeySyncMaterial**. The **LCF** response shall hold the ClearToken CT_{HG}.

The gatekeeper G, recognizing that endpoints A and B support this annex, shall generate key material and ClearTokens as specified below.

The gatekeeper is able to calculate a call-based shared secret K_{AB}, besides the normal **ARQ** operation. This call-based shared secret is then propagated to both endpoints using ClearTokens. Those ClearTokens are conveyed within the **ACF** message and are sent back to the caller.

Two ClearTokens shall be included, one CT_A for the caller A and another one CT_B for the callee B. Each **ClearToken** shall contain an OID ("I1" or "I2") within **tokenOID** that indicates whether the token is destined for the caller (OID "I1" for CT_A) or for the callee (OID "I2" for CT_B).

The **ClearToken** as defined in this annex may be used in conjunction with other security profiles such as with Annex D or with Annex F that deploy **ClearTokens** as well. In such a case, Annex I ClearToken shall use those other **ClearToken** fields too. For example, in order to use Annex I in conjunction with Annex D, the fields **timeStamp**, **random**, **generalID**, **sendersID**, and **dhkey** shall be present and shall be used, as described by the Annex D security profiles.

The gatekeeper ID (GKID) shall be placed within **sendersID** whereas **generalID** shall hold either the endpoint identifier of endpoint A (CT_A) or of endpoint B (CT_B).

EK~~'~~ denotes the encryption key that is shared between an~~d~~ endpoint and its~~the~~ GK. The encryption keys EK~~'~~$_{AG}$ and EK~~'~~$_{H}$ for the encrypted end-to-end key K_{AB} shall be derived from the shared secret between the gatekeeper and the endpoints (K_{AG} or $K_{BH\cancel{G}}$) using the PRF-**based** key derivation procedure as defined in clause I.10 where **keyDerivationOID** in **V3KeySyncMaterial** shall hold "Annex I-HMAC-SHA1-PRF", see I.12.

The gatekeeper G shall generate a common shared session secret K_{AB}, which is shared between endpoint A and endpoint B.

This session secret K_{AB} shall be encrypted by EK~~'~~$_{AG}$ (for CT destined to endpoint A) or by EK~~'~~$_{BH\cancel{G}}$ (for the CT destined to endpoint B) using an encryption algorithm.

The enhanced OFB (EOFB) encryption mode (see B.2.5) shall be used with the secret, endpoint-specific salting key KS_{AG} resp. KS_{BG}. Applicable encryption algorithms are (see clause D.11):

- DES (56 bits) in EOFB mode using OID "Y1": optional;
- 3DES (168 bits) in outer-EOFB mode using OID "Z1": optional;
- AES (128 bits) in EOFB mode using OID "Z2": default and recommended;
- RC2-compatible (56 bits) in EOFB mode using OID "X1": optional.

For the EOFB encryption mode, the GK shall generate a random initial value IV. For OID "X1", OID "Y1" and OID "Z1" the IV has 64 bits and shall be conveyed within **iv8** of **paramS** within **V3KeySyncMaterial;** whereas the IV has 128 bits for OID "Z2" and shall be conveyed within **iv16** of **params** within **V3KeySyncMaterial**.

The obtained ciphertext $\mathrm{ENC}_{EK_{AG}, KS_{AG}, IV}$(~~{~~$K_{AB}$)~~}$_{K'_{AG}, KS_{AG}, IV}$~~ resp. $\mathrm{ENC}_{EK_{BG}, KS_{BG}, IV}$(~~{~~$K_{AB}$)~~}$_{K'_{BG}, KS_{BG}, IV}$~~ shall then be conveyed in the **h235key** data structure as part of **secureSharedSecret** where it shall be placed within the **encryptedSessionKey** of the **secureSharedSecret** data structure. The encryption algorithm shall be indicated in **algorithmOID** ("X1", "Y1", "Z1" or "Z2") within **V3KeySyncMaterial**.

For the ClearToken destined to endpoint A, the endpoint identifier of endpoint B ($EPID_B$) shall be placed within **generalID** of **V3KeySyncMaterial**. Likewise for the ClearToken destined to endpoint B, the endpoint identifier of endpoint A ($EPID_A$) shall be placed within **generalID** of **V3KeySyncMaterial**.

For the EOFB encryption algorithms, **encryptedSaltingKey** shall not be used.

The gatekeeper shall include both ClearTokens CT_A and CT_B in the **ACF** towards endpoint A.

Endpoint A shall identify CT_A by inspection of the **tokenOID** "I1" within ClearToken.

Endpoint A shall verify that the obtained CT_A is fresh by checking the **timestamp**. Further security checks shall verify the **generalID** and **sendersID** of the ClearToken and **generalID** within **V3KeySyncMaterial**. If the received CT_A was verified as being fresh, endpoint A shall retrieve the IV and compute EK~~$'$~~$_{AG}$ and KS_{AG} as described above for the gatekeeper G. Endpoint A shall decrypt the **encryptedSessionKey** information found within **V3KeySyncMaterial** of CT_A to obtain EK~~$'$~~$_{AB}$.

If the received CT_A was verified as being fresh, endpoint A is able to send a SETUP message to endpoint B. This SETUP message includes CT_B. The SETUP message shall be secured (authenticated and/or integrity protected) according to Annex D or according to Annex F using K_{AB} as the applied shared secret. For this, **generalID** in the Annex D hashed ClearToken (not CT_B!) shall not be used~~t~~ unless endpoint A has already an $EPID_B$ available (e.g., through configuration or memorized from former communication). If endpoint A uses an $EPID_B$ value for **generalID** in SETUP then endpoint A shall accept the value of the **sendersID** in the returned call signaling message as the true~~to~~ $EPID_B$.

Endpoint B shall identify CT_B by inspection of the **tokenOID** "I2" within ClearToken.

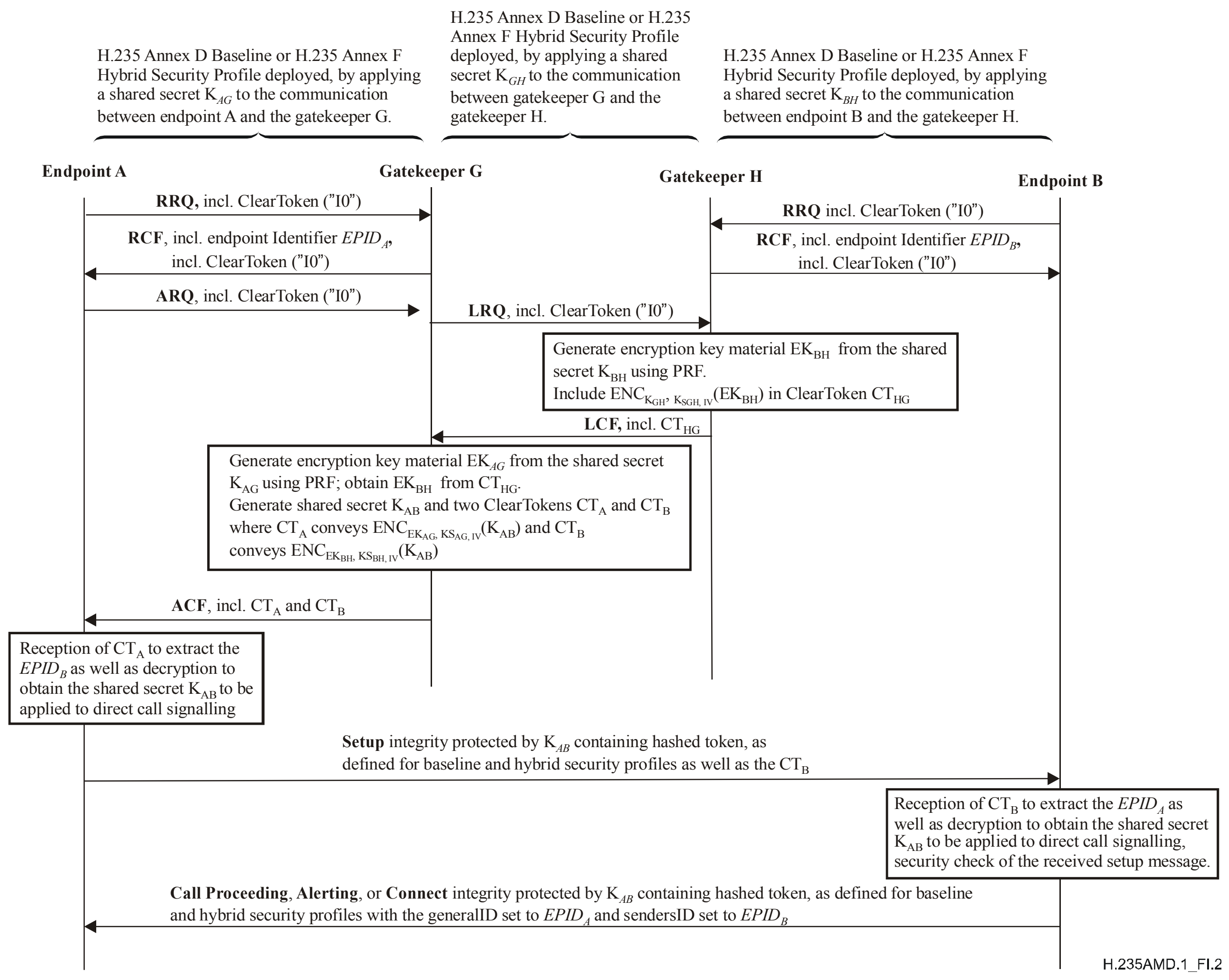
Endpoint B shall verify that the obtained CT_B is fresh by checking the **timestamp**. Further security checks shall verify the ~~**generalID** and~~ **sendersID** of the ClearToken and **generalID** within **V3KeySyncMaterial**. If the received CT_B was verified as being fresh, endpoint B shall retrieve the IV and compute EK~~$'$~~$_{BG}$ and KS_{BG} as described above for the gatekeeper. Endpoint B shall decrypt the **encryptedSessionKey** information found within **V3KeySyncMaterial** of CT_B to obtain EK~~$'$~~$_{AB}$.

In case CT_B was verified as being fresh, endpoint B is able to proceed the call signalling by replying with CALL-PROCEEDING, ALERT~~ER~~ING or CONNECT etc., as appropriate. In case CT_B was found not to be fresh or the security verification of the SETUP message failed, endpoint B shall reply with RELEASE-COMPLETE and the **ReleaseCompleteReason** set to a security error as defined by B.2.2.

When media security is to be deployed (see clause D.7), endpoint A and endpoint B shall exchange Diffie-Hellman half-keys according to D.7.1 and establish a dynamic session-based master key from which media-specific session keys can then be derived.

Endpoint B shall include generalID set to EPID$_A$ and sendersID set to EPID$_B$ for protection of any H.225.0 Call signaling message destined to EP A (e.g., Call Proceeding, Alerting or Connect).

Figure I.2 shows the basic communication flow:



H.235 Annex D Baseline or H.235 Annex F Hybrid Security Profile deployed, by applying a shared secret K_{AG} to the communication between endpoint A and the gatekeeper G.
H.235 Annex D Baseline or H.235 Annex F Hybrid Security Profile deployed, by applying a shared secret K_{GH} to the communication between gatekeeper G and the gatekeeper H.
H.235 Annex D Baseline or H.235 Annex F Hybrid Security Profile deployed, by applying a shared secret K_{BH} to the communication between endpoint B and the gatekeeper H.
Endpoint A
Gatekeeper G
Gatekeeper H
Endpoint B
RRQ, incl. ClearToken ("I0")
RRQ incl. ClearToken ("I0")
RCF, incl. endpoint Identifier $EPID_A$, incl. ClearToken ("I0")
RCF, incl. endpoint Identifier $EPID_B$, incl. ClearToken ("I0")
ARQ, incl. ClearToken ("I0")
LRQ, incl. ClearToken ("I0")
Generate encryption key material EK_{BH} from the shared secret K_{BH} using PRF. Include $ENC_{K_{GH}, K_{SGH, IV}}(EK_{BH})$ in ClearToken CT_{HG}
LCF, incl. CT_{HG}
Generate encryption key material EK_{AG} from the shared secret K_{AG} using PRF; obtain EK_{BH} from CT_{HG}. Generate shared secret K_{AB} and two ClearTokens CT_A and CT_B where CT_A conveys $ENC_{EK_{AG}, KS_{AG, IV}}(K_{AB})$ and CT_B conveys $ENC_{EK_{BH}, KS_{BH, IV}}(K_{AB})$
ACF, incl. CT_A and CT_B
Reception of CT_A to extract the $EPID_B$ as well as decryption to obtain the shared secret K_{AB} to be applied to direct call signalling
Setup integrity protected by K_{AB} containing hashed token, as defined for baseline and hybrid security profiles as well as the CT_B
Reception of CT_B to extract the $EPID_A$ as well as decryption to obtain the shared secret K_{AB} to be applied to direct call signalling, security check of the received setup message.
Call Proceeding, Alerting, or Connect integrity protected by K_{AB} containing hashed token, as defined for baseline and hybrid security profiles with the generalID set to $EPID_A$ and sendersID set to $EPID_B$
H.235AMD.1_FI.2

Figure I.2/H.235 – Basic communication flow

I.10 PRF-based key derivation procedure

This clause describes a procedure that defines how to derive key material from the shared secret and other parameters.

The encryption key EK~~'~~$_{AG}$ shall be computed using the PRF (see clause B.7) with the *inkey* parameter set to K_{AG} and *label* shall be set to the constant 0x2AD01C64 || **challenge**.

Likewise, the encryption key EK~~'~~$_{BG}$ shall be computed using that PRF with the *inkey* parameter set to K_{BH}~~$_G$~~ and *label* shall be set to the constant 0x1B5C7973 || **challenge**. In both cases, *outkey_len* shall be set to the length of the required length of the encryption key for the chosen encryption algorithm.

Using that same PRF, a secret, shared salting key shall be generated by the gatekeeper and by each endpoint. The salting key, when being used in conjunction with the EOFB encryption mode, guards against known-plaintext attacks of the CT_B by EP A where EP A might otherwise attempt to discover $K_{B\underline{H}\cancel{G}}$.

KS_{AG} denotes the secret, shared salting key that is shared between EP A and the GK G. KS_{AG} shall be computed using the PRF with the *inkey* parameter set to K_{AG} and *label* shall be set to the constant 0x150533E1 || **challenge**. $KS_{B\underline{H}\cancel{G}}$ shall be computed using PRF with the *inkey* parameter set to $K_{B\underline{H}\cancel{G}}$ and *label* shall be set to the constant 0x39A2C14B || **challenge**.

…

Table I.1/H.235 – Object identifiers used by H.235 Annex I

Object identifier reference	Object identifier value	Description
…	…	…
"I2"	{itu-t (0) recommendation (0) h (8) 235 version (0) 3 50}	Used in procedure DRC for the ClearToken tokenOID indicating that the ClearToken holds an end-to-end key for the callee.
"I3"	{itu-t (0) recommendation (0) h (8) 235 version (0) 3 52}	Used in procedure DRC for the inter-gatekeeper ClearToken tokenOID indicating that the ClearToken holds an encryption key for the originating gatekeeper.
…	…	…

Appendix I

H.323 implementation details

…

I.4.5 IPSEC

In general, IPSEC ([IPSEC], [ESP]) and IKE [IKE] can be used to provide authentication and, optionally, confidentiality (i.e., encryption) at the IP layer transparent to whatever (application) protocol runs above. The application protocol does not have to be updated to allow this; only security policy at each end.

For example, to make maximum use of IPSEC for a simple point-to-point call, the following scenario could be followed:

1) The calling endpoint and its gatekeeper would set policy to require the use of IPSEC (authentication and, optionally, confidentiality) on the RAS protocol. Thus, before the first RAS message is sent from the endpoint to the gatekeeper, the ISAKMP [ISAKMP]/Oakley [RFC 2412] daemon on the endpoint will negotiate security services to be used on packets to and from the RAS channel's well-known port. Once negotiation is complete, the RAS channel will operate exactly as if it were not secured. Using this secure channel the gatekeeper will inform the endpoint of the address and port number of the call signalling channel in the called endpoint.

2) After obtaining the address and port number of the call signalling channel, the calling endpoint would dynamically update its security policy to require the desired IPSEC security on that address and protocol/port pair. Now, when the calling endpoint attempts to contact this address/port, the packets would be queued while an ISAKMP [ISAKMP]/Oakley [RFC 2412] negotiation is performed between the endpoints. Upon completion of this negotiation, an IPSEC Security Association (SA) for the address/port will exist and the Q.931 signalling can proceed.

3) On the Q.931 SETUP and CONNECT exchange, the endpoints can negotiate the use of IPSEC for the H.245 channel. This will allow the endpoints to again dynamically update their IPSEC policy databases to force the use of IPSEC on that connection.

4) As with the call signalling channel, a transparent ISAKMP [ISAKMP]/Oakley [RFC 2412] negotiation will take place before any H.245 packets are transmitted. The authentication performed by this ISAKMP [ISAKMP]/Oakley [RFC 2412] exchange will be the initial attempt at user-to-user authentication, and will set up a (probably) secure channel between the two users on which to negotiate the characteristics of the audio channel. If, after some person-to-person Q&A, either user is not satisfied with the authentication, different certificates can be chosen and the ISAKMP [ISAKMP]/Oakley [RFC 2412] exchange repeated.

5) After each H.245 ISAKMP [ISAKMP]/Oakley [RFC 2412] authentication, new keying material is exchanged for the RTP audio channel. This keying material is distributed by the master on the secure H.245 channel. Because the H.245 protocol is defined for the master to distribute the media keying material on the H.245 channel (to allow for multipoint communication), it is not recommended that IPSEC be used for the RTP channel.

…

Appendix IV

Bibliography

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

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

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

[ISAKMP~~PSEC~~] IETF RFC 2408 (1998), ~~MAUGHAN (D.), SCHERTLER (M.), SCHNEIDER (M.), TURNER (J.),~~ *Internet Security Association and Key Management Protocol (ISAKMP)*~~, draft-ietf-ipsec-isakmp-08.text,~~ ~~*Internet Engineering Task Force*, 1997~~.

…

[MIKEY] ARKKO (J.), CARRARA (E.), LINDHOLM (F.), NASLUND (M.), NORRMAN (K.): MIKEY:Multimedia Internet KEYing, *Internet Draft <draft-ietf-msec-mikey-08~~6~~.txt>*, RFC xxxx, Work in Progress (MSEC WG), IETF, 1~~0~~2/2003.

{Editor's note: This RFC # will be included when available.}

…

[RTP]	IETF RFC 3550 (2003), ~~SCHULZRINNE (H.), CASNER (S.), FREDERICK (R.), JACOBSON (V.), RTP:~~ *RTP: A transport Protocol for Real-Time Applications*~~, RFC 3550, *Internet Engineering Task Force*, 2003~~.

…

[SRTP]	IETF RFC 3711 (2004), ~~Baugher, McGrew, Oran et al:~~ *The Secure Real-time Transport Protocol (SRTP)*~~; draft-ietf-avt-srtp-09.txt, RFC xxxx; Internet Engineering Task Force, 2003~~.
	~~{Editor's note: This RFC# will be included when available}~~
~~[TLS]~~	~~DIEKS (T.), ALLEN (C.): *The TLS Protocol Version 1.0*, RFC 2246, *Internet Engineering Task Force*, 1999.~~

SERIES OF ITU-T RECOMMENDATIONS

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

Printed in Switzerland
Geneva, 2004