International Telecommunication Union

ITU-T

TELECOMMUNICATION
STANDARDIZATION SECTOR
OF ITU

H.235.5

(09/2005)

SERIES H: AUDIOVISUAL AND MULTIMEDIA SYSTEMS

Infrastructure of audiovisual services – Systems aspects

H.323 security: Framework for secure authentication in RAS using weak shared secrets

ITU-T Recommendation H.235.5

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.235.5

H.323 security: Framework for secure authentication in RAS using weak shared secrets

Summary

This Recommendation 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.

In earlier versions of the H.235 sub-series, this profile was contained in H.235 Annex H. Appendices IV, V, VI to H.235.0 show the complete clause, figure, and table mapping between H.235 versions 3 and 4.

Source

ITU-T Recommendation H.235.5 was approved on 13 September 2005 by ITU-T Study Group 16 (2005-2008) under the ITU-T Recommendation A.8 procedure.

Keywords

Authentication, passwords, security.

FOREWORD

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

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

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

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

NOTE

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

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

INTELLECTUAL PROPERTY RIGHTS

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

As of the date of approval of this Recommendation, ITU had received notice of intellectual property, protected by patents, which may be required to implement this Recommendation. However, implementors are cautioned that this may not represent the latest information and are therefore strongly urged to consult the TSB patent database.

CONTENTS

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 Simple Password Exponential Key Exchange (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.

ITU-T Recommendation H.235.5

H.323 security: Framework for secure authentication in RAS using weak shared secrets

1 Scope

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

2 References

2.1 Normative references

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

– ITU-T Recommendation H.225.0 (2003), *Call signalling protocols and media stream packetization for packet-based multimedia communication systems*.

– ITU-T Recommendation H.235.0 (2005), *H.323 security: Framework for security in H-series (H.323 and other H.245-based) multimedia systems*.

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

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

– ITU-T Recommendation H.323 (2003), *Packet-based multimedia communications systems*.

– Federal Information Processing Standard FIPS PUB 180-2, *Secure Hash Standard*, U.S. Department of Commerce, Technology Administration, National Institute of Standards and Technology, 1 August 2002.

– 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.

2.2 Informative references

[AES] IETF RFC 3268 (2002), *Advanced Encryption Standard (AES) Ciphersuites for Transport Layer Security*.

[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 SP 800-57] NIST Draft Special Publication 800-57 (2005), *Recommendation for Key Management, Part 1: General Guideline*. http://www.csrc.nist.gov/publications/drafts/draft-800-57-Part1-April2005.pdf

[RFC2104] IETF RFC 2104 (1997), *HMAC: Keyed-Hashing for Message Authentication*.

[RFC2412] IETF RFC 2412 (1998), *The OAKLEY Key Determination Protocol*.

[RFC2246] IETF RFC 2246 (1999), *The TLS Protocol Version 1.0*.

[RFC3546] IETF RFC 3546 (2003), *Transport Layer Security (TLS) Extensions*.

3 Definitions

None.

4 Abbreviations

This Recommendation uses the following abbreviations:

ACF Admission Confirm

AES Advanced Encryption Standard

ARJ Admission Reject

ARQ Admission Request

CBC Cipher Block Chaining

CTR Counter Mode (see NIST SP 800-38A)

D-H Diffie-Hellman

EKE Encrypted Key Exchange

GCF Gatekeeper Confirm

GK Gatekeeper

GRJ Gatekeeper Reject

GRQ Gatekeeper Request

HMAC Hashed Message Authentication Code

ICV Integrity Check Value

ID Identifier

LCF Location Confirm

LRJ Location Reject

LRQ Location Request

MIM Man-in-the-middle

OID Object Identifier

PIN Personal Identification Number

PRF Pseudo-Random Function

RAS Registration, Admission and Status

RCF Registration Confirm

RFC Request for Comments

RRJ Registration Reject

RRQ Registration Request

SHA1 Secure Hash Algorithm 1

SPEKE Simple Password Exponential Key Exchange

TLS Transport Layer Security

UDP User Datagram Protocol

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.

For further conventions, refer to clause 5/H.235.0.

6 Basic framework

6.1 Improved negotiation capabilities in H.235.0

ITU-T Rec. H.235.0 provides support for this security framework via the inclusion 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.

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**ReQ**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 clause 10/H.235.0), message authentication code (e.g., HMAC-SHA1-96 [RFC2104]), 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 ciphersuite (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 ITU-T Rec. H.235.1.

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.

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.

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**.

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 SP 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 [FIPS PUB 180-1] 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 12-octet initialization vector, IV, and a 2-octet counter field, C, such that counter = D || IV || C, and C = 0 initially. See [NIST SP 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 defined in clause 10/H.235.0, 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 1. These elements are carried in a **ClearToken.profileInfo** element sequence as defined in ITU-T Rec. H.235.0.

Table 1/H.235.5 – 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 a 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 a 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 10.5.

– 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 6.3).

8 An improved security profile (SP2)

This clause defines a new security profile based on the original profile, SP1. It is identified informally as SP2 and formally with OID {itu-t (0) recommendation (0) h (8) 235 version (0) 4 62}. This profile is identical to SP1, except as specified in the following subclauses. The specific improvements over SP1 include:

– Improvements in the call signalling message sequence numbering to counter replay attacks.

– Salting of the generation of the password-based encryption key using the endpoint alias in order to counter dictionary attacks.

– The nonce size is increased and made variable.

– A salting key is derived for use with the encryption initialization vector.

– A more efficient transport of the profile **ClearToken** is provided using **genericData.**

SP2 uses the Profile Elements of Table 1, as well as the additional Profile Elements described in Table 2:

Table 2/H.235.5 – Additional profile elements for SP2

Element name (used in text)	ElementID value	Element choice (length)	Element description
seqNumber	7	Octets (4)	32-bit sequence number in network byte order
connectID	8	Octets (2)	Signalling Connection Identifier. (Optional, default = 0)
endpointID	9	Octets (variable)	ASN.1-encoded AliasAddress associated with the endpoint and its password. (Optional)

8.1 Call Signalling sequence number

H.225.0 Call Signalling messages do not contain a sequence number because they are transported over a reliable connection (TCP) which is responsible for sequencing. Nevertheless, lack of a unique message identifier at the application level does expose call signalling to replay or reflection attacks. This problem can be countered by adding a sequence number, and an optional connection identifier, to each call signalling message. Note that this technique does not completely prevent a replay or reflection attacks, but it severely reduces the attacker's chances of success.

Sequence numbers must be unique in each direction to prevent reflection. This can be accomplished, within practical limits, by requiring the issuer of the GRQ (endpoint) or LRQ (gatekeeper) message to start its call signalling transmit sequence number at 0 (zero) and its receive sequence number at 2^{31}, with corresponding behaviour by the receiving gatekeeper. This provides a very large time before any overlap can occur (almost 600 hours at the quite unusual rate of one message per millisecond). Successive calls using the same SessionID should transmit with the next unused sequence number in each direction. (To allow for lost messages over failed connections, the receiver should accept messages within a small (e.g., 5-10) window following the last received sequence number, and proceed from there.) Devices that support multiple simultaneous call signalling connections under the same session ID, may use an optional **connectID** to identify separate sequence number spaces for the calls. If not specified, the **connectID** is presumed to be 0 (zero).

8.2 Generation of Weak Encryption Key from password

In order to prevent dictionary attacks in which a PIN is guessed, used to encrypt a D-H public key, and then applied successively to all known endpoint aliases, it is desirable to "salt" the encryption key with the alias itself. In particular, the password-based key, K_p, shall be computed from the concatenation of the password and the supplied **endpointID**:

$$K_p = \text{Trunc}(\text{SHA1}(\text{user password} \,||\, \textbf{endpointID}), 16)$$

Typically, the **AliasAddress** in **endpointID** will be one of the aliases included in the GRQ **endpointAlias** element, but this is not necessary. For example, the **endpointID** may identify a gateway supporting many endpoints whose aliases are listed in **endpointType**.

8.3 Nonce size

Security Profile 1 requires each party to supply a 4-octet (32-bit) nonce as part of the key negotiation protocol. When supplied during the initial key negotiation, 32 bits is perhaps sufficient to insure freshness in those cases in which the responding gatekeeper reuses the same Diffie-Hellman public key but the requestor generates a fresh key. However, when negotiating new session keys from a previously-negotiated master key, the total of 64 unique bits may not provide sufficient difference between each set of derived keys. It is proposed that the nonce size be variable, with a minimum of 4 octets, and a maximum of 16.

8.4 Initialization vector salting

As an added obfuscation measure, a 112-bit session salting key, K_s, is derived from the negotiated master key as:

$$K_s = \text{PRF}(K_m, \text{"salting\_key"} \,||\, R_e \,||\, R_g, 112).$$

Construction of the initial AES-128-CM counter for encryption and decryption is then carried out as follows:

Counter = (K_s ^ (D || IV)) || C, where C is the 16-bit counter field, initially zero.

8.5 ClearToken encoding

Security profile 1 uses the **clearToken** sequence to carry the parameters of the profile. Every H.225.0 message contains a sequence of **ClearToken**s, except the **empty** choice of **h323-message-body**; all messages carry **genericData**. The structure of the SP1 procedures permit a rather regular structure for the **ClearToken**, which permits it to be ASN.1-encoded ahead of time, and carried as a **raw** parameter with **id.standard** set to 1 in a **GenericData** element identified by the SP2 OID. This form permits identification of the **clearToken** by the "null" OID {0,0}. Most importantly, it makes it easier to locate the token, and the check value within it, because the encoded form of the **ClearToken** alone becomes available as part of the normal encoding and decoding process. Thus it is faster to locate the integrityCheck element within the encoded clear token than within the entire encoded message.

9 Extensions to the framework (Informative)

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

9.1 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 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.

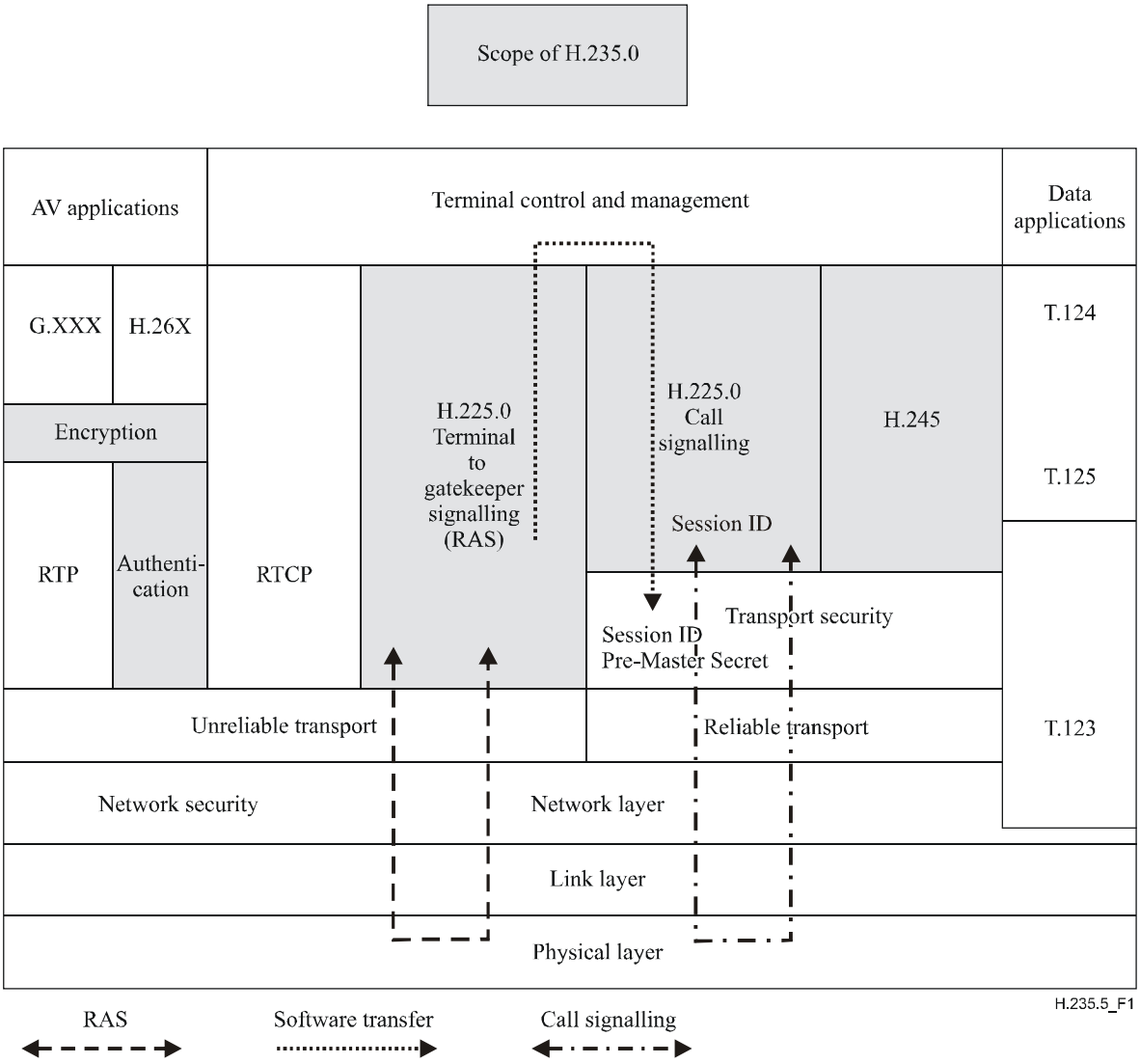

Figure 1/H.235.5 – Information flow for security profile and TLS

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

9.1.1 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.

9.2 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.

9.3 Use of alternative signalling security mechanisms

The parameters negotiated as part of a security profile under this Recommendation 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.

10 Threats (Informative)

10.1 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.

10.2 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.

10.3 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.

10.4 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.

10.5 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.

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	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, 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, 2006