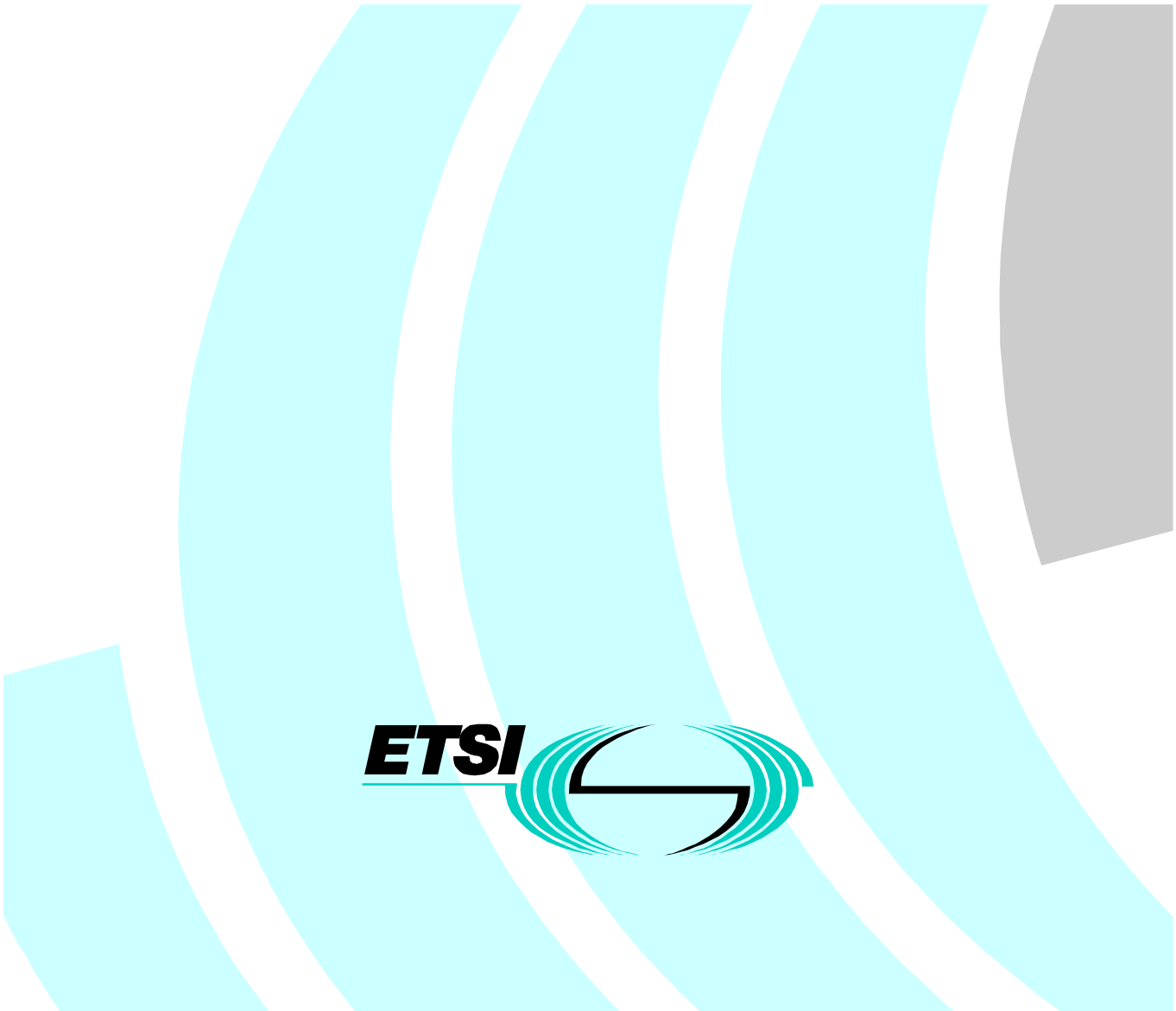
# ETSI TS 101 909-11 V1.1.1 (2001-07)

*Technical Specification*

**Access and Terminals (AT);**
**Digital Broadband Cable Access to the Public Telecommunications Network;**
**IP Multimedia Time Critical Services;**
**Part 11: Security**



Reference
DTS/AT-020020-11

Keywords
access, broadband, cable, IP, multimedia, PSTN, security

***ETSI***

650 Route des Lucioles
F-06921 Sophia Antipolis Cedex - FRANCE

Tel.: +33 4 92 94 42 00 Fax: +33 4 93 65 47 16

Siret N° 348 623 562 00017 - NAF 742 C
Association à but non lucratif enregistrée à la
Sous-Préfecture de Grasse (06) N° 7803/88

# Contents

# Intellectual Property Rights

IPRs essential or potentially essential to the present document may have been declared to ETSI. The information pertaining to these essential IPRs, if any, is publicly available for **ETSI members and non-members**, and can be found in ETSI SR 000 314: *"Intellectual Property Rights (IPRs); Essential, or potentially Essential, IPRs notified to ETSI in respect of ETSI standards"*, which is available from the ETSI Secretariat. Latest updates are available on the ETSI Web server (http://www.etsi.org/ipr).

Pursuant to the ETSI IPR Policy, no investigation, including IPR searches, has been carried out by ETSI. No guarantee can be given as to the existence of other IPRs not referenced in ETSI SR 000 314 (or the updates on the ETSI Web server) which are, or may be, or may become, essential to the present document.

# Foreword

This Technical Specification (TS) has been produced by ETSI Technical Committee Access and Terminals (AT).

The present document is part 11 of a multi-part deliverable supporting real-time multimedia services, as identified below:

Part 1: "General";

Part 2: "Architectural framework for the delivery of time critical services over cable Television networks using cable modems";

Part 3: "Audio Codec Requirements for the Provision of Bi-Directional Audio Service over Cable Television Networks using Cable Modems";

Part 4: "Network Call signalling Protocol";

Part 5: "Dynamic Quality of Service for the Provision of Real Time Services over Cable Television Networks using Cable Modems";

Part 6: "Media Terminal Adapter (MTA) device provisioning";

Part 7: "Management Information Base (MIB) Framework";

Part 8: "Media Terminal Adapter (MTA) Management Information Base (MIB)";

Part 9: "Network Call Signalling (NCS) MIB Requirements";

Part 10: "Event Message Requirements for the Provision of Real Time Services over Cable Television Networks using Cable Modems";

**Part 11: "Security";**

Part 12: "Internet Signalling Transport Protocol";

Part 13: "Trunking Gateway Control Protocol";

Part 14: "Operation System Support".

NOTE 1: The above list is complete for the first version of this Technical Specification (TS) (V1.1.1 2001-07). Additional parts are being proposed and these will be added to the list in future versions.

The present part is part 11 of the above-mentioned series of ETSI deliverables and describes the IPCablecom Security architecture, protocols, algorithms, associated functional requirements and any technological requirements that can provide for the security of the system for the IPCablecom network. Authentication, access control, message and bearer content integrity, confidentiality and non-repudiation security services must be provided as defined herein for each of the network element interfaces. The present set of documents specify IPCablecom, a set of protocols and associated element functional requirements. These have been developed to deliver Quality-of-Service (QoS), enhanced secure IP multimedia time critical communication services, using packetized data transmission technology to a consumer's home over a cable television Hybrid Fibre/Coaxial (HFC) data network.

NOTE 2: The choice of a multi-part format for this deliverable is to facilitate maintenance and future enhancements.

NOTE 3: The term **MUST** or **MUST NOT** is used as a convention in the present document part to denote an absolutely mandatory aspect of the specification.

NOTE 4: The provisions of this part of the specification have a potential impact on the provisions and operation of a later part detailing Lawful Interception. It is therefore possible that this part may need to be revised as work progress on the latter part. This is in order to align both parts of this specification to facilitate compliance with the requirements of both parts.

# Introduction

The cable industry in Europe and across other Global regions have already deployed broadband cable television hybrid fibre coax (HFC) data networks running the Cable Modem Protocol. The cable industry is in the rapid stages of deploying IP Voice and other time critical multimedia services over these broadband cable television networks.

The cable industry has recognized the urgent need to develop ETSI Technical Specifications aimed at developing interoperable interface specifications and mechanisms for the delivery of end to end advanced real time IP multimedia time critical services over bi-directional broadband cable networks.

IPCablecom is a set of protocols and associated element functional requirements developed to deliver Quality-of-Service (QoS) enhanced secure IP multimedia time critical communications services using packetized data transmission technology to a consumer's home over the broadband cable television Hybrid Fibre/Coaxial (HFC) data network running the Cable Modem protocol. IPCablecom utilizes a network superstructure that overlays the two-way data-ready cable television network. While the initial service offerings in the IPCablecom product line are anticipated to be Packet Voice, the long-term project vision encompasses packet video and a large family of other packet-based services.

The cable industry is a global market and therefore the ETSI standards are developed to align with standards either already developed or under development in other regions. The ETSI Specifications are consistent with the CableLabs/PacketCable set of specifications as published by the SCTE. An agreement has been established between ETSI and SCTE in the US to ensure, where appropriate, that the release of PacketCable and IPCablecom set of specifications are aligned and to avoid unnecessary duplication. The set of IPCablecom ETSI specifications also refers to ITU-SG9 draft and published recommendations relating to IP Cable Communication.

The whole set of multi-part ETSI deliverables to which the present document belongs specify a Cable Communication Service for the delivery of IP Multimedia Time Critical Services over a HFC Broadband Cable Network to the consumers home cable telecom terminal. 'IPCablecom' also refers to the ETSI working group program that shall define and develop these ETSI deliverables.

IPCablecom is a set of protocols and associated element functional requirements developed to deliver Quality of Service (QoS) enhanced secure communications services using packetized data transmission technology to a consumer's home over the cable television Hybrid Fibre/Coaxial (HFC) data network. IPCablecom utilizes a network superstructure that overlays the two-way data-ready cable television network. While the initial service offerings in the IPCablecom product line are anticipated to be Packet Voice and Packet Video, the long-term project vision encompasses a large family of packet-based services.

The purpose of any security technology is to protect value, whether a revenue stream, or a purchasable information asset of some type. Threats to this revenue stream exist when a user of the network perceives the value, expends effort and money, and invents a technique to get around the necessary payments. Some network users will go to extreme lengths to steal when they perceive extreme value. The addition of security technology to protect value has an associated cost; the more expended, the more secure one can be. The proper engineering task is to employ a reasonable costing security technology to force any user with the intent to steal or disrupt network services to spend an unreasonable amount of money to circumvent it. Security effectiveness is thus basic economics.

In addition, a IPCablecom network used to offer voice communications must be at least as secure as the Public Switched Telephone Network (PSTN) networks are today. Much of the PSTN security depends on the fact that each telephone is connected to a dedicated line. In order to provide the same level of privacy and resistance to denial of service attacks when a IPCablecom network is used for voice communications, where both voice and signalling data is transmitted over a shared HFC network and over a shared Internet Protocol (IP) backbone, appropriate cryptography-based security mechanisms have been specified.

IPCablecom security spans all of the IPCablecom specifications. The IPCablecom Architecture Specification defines the overall IPCablecom architecture, the system elements, interfaces, and functional requirements for the entire IPCablecom network.

The present document describes the security relationships between all of the above listed specifications. The general goals of the IPCablecom network security Specification and any implementations that encompass the requirements defined herein should be:

- **Secure network communications**. The IPCablecom network security must define a security architecture, methods, algorithms and protocols that meet the stated security service requirement. All media packets and all sensitive signalling communication across the network must be safe from eavesdropping. Unauthorized message modification, insertion, deletion and replays anywhere in the network must be easily detectable and must not affect proper network operation.

- **Reasonable cost**. The IPCablecom network security must define security methods, algorithms and protocols that meet the stated security service requirements such that a reasonable implementation can be manifested with reasonable cost and implementation complexity.

- **Network element interoperability**. All of the security services for any of the IPCablecom network elements must inter-operate with the security services for all of the other IPCablecom network elements. Multiple vendors may implement each of the IPCablecom network elements as well as multiple vendors for a single IPCablecom network element.

- **Extensibility**. The IPCablecom security architecture, methods, algorithms and protocols must provide a framework into which new security methods and algorithms may be incorporated as necessary.

The following assumptions are made relative to the scope of the IPCablecom Security Specification:

- Embedded MTAs are within the scope of IPCablecom. Standalone MTAs will be addressed in later phases and security issues for Standalone MTAs are outside the scope of the present document.

- All MTAs must use J.112 compliant CMs and must implement MAC layer security.

- NCSv2 is the only call signalling method addressed in the present document.

- IPCablecom specifies requirements for a single administrative domain. CMS to CMS communication is proprietary in 1.0, therefore security mechanisms for inter-CMS communication are also assumed to be proprietary and are not specified in the present document.

- A single Certificate Authority hierarchy is assumed for the purposes of the present document since IPCablecom is restricted to a single administrative domain. Cross-administrative domain security is outside the scope of the present document.

- Anonymity of IP addresses is out of scope of IPCablecom and therefore outside the scope of the present document.

- Security for chained Radius servers is outside the scope of the IPCablecom Security Specification.

- A subset of DQoS may be included in IPCablecom.

- Exportability of encryption algorithms is not addressed in the present document.

The present document also does not include requirements for associated security operational issues (e.g. site security), back office or inter/intra back office security, service authorization policies or secure database handing. Record Keeping Servers (RKS), Network Management Systems, File Transfer Protocol (FTP) servers and Dynamic Host Control Protocol (DHCP) servers are all considered to be unique to any service providers implementation and are beyond the scope of the present document.

Document Overview

The present document covers security for the entire IPCablecom architecture. The present document describes the IPCablecom architecture, identifies security risks and specifies mechanisms to secure the architecture. The document is structured as follows:

- Architectural Overview of IPCablecom
  - The initial clause describes the IPCablecom architecture as a point of reference for the remainder of the document. Refer to the Architecture Framework Document TS 101 909-2 and each individual Specification for full details.
- Security Threats are described in the context of the reference architecture.
- The overall security architecture and security assumptions are described.
- Security Mechanisms - This clause specifies how public domain security mechanisms are to be implemented in IPCablecom including Internet Protocol Security (IPSec), Internet Key Exchange (IKE), Kerberos with PKINIT, media stream security, MAC layer security and Radius.
- Security Profile - This clause profiles the security for each major area of the IPCablecom architecture. The profile includes a description of the security requirements as well as the Specifications for securing at-risk interfaces. Refer to the individual Specifications for details about each IPCablecom area. Security Specifications are provided for each of the following areas:
  - Device and Service Provisioning.
  - Quality of Service Signalling.
  - OSS Interfaces.
  - Billing System Interfaces.
  - Call Signalling.
  - PSTN Gateway Interfaces.
  - Media Stream.
- IPCablecom X.509 Certificate Profile and Management - X.509 Certificates are specified for a number of devices and functions within the IPCablecom architecture. This clause describes the format of the Certificates as well as the trust hierarchy for Certificate management within IPCablecom.
- Cryptographic Algorithms - This clause specifies the details of cryptographic algorithms specified in the IPCablecom security architecture.
- Physical Security - This clause documents assumptions about the physical security of the Media Terminal Adapter (MTA keys).
- Secure Software Upgrade - This clause specifies the secure loading and upgrading of software to the MTAs.

# 1 Scope

The present set of documents specify IPCablecom, a set of protocols and associated element functional requirements. These have been developed to deliver Quality-of-Service (QoS), enhanced secure IP multimedia time critical communication services, using packetized data transmission technology to a consumer's home over a cable television Hybrid Fibre/Coaxial (HFC) data network.

NOTE 1: IPCablecom set of documents utilize a network superstructure that overlays the two-way data-ready cable television network, e.g. as specified within ES 201 488 and ES 200 800.

While the initial service offerings in the IPCablecom product line are anticipated to be Packet Voice and Packet Video, the long-term project vision encompasses a large family of packet-based services. This may require in the future, not only careful maintenance control, but also an extension of the present set of documents.

NOTE 2: The present set of documents aims for global acceptance and applicability. It is therefore developed in alignment with standards either already existing or under development in other regions and in International Telecommunications Union (ITU).

# 2 References

The following documents contain provisions which, through reference in this text, constitute provisions of the present document.

- References are either specific (identified by date of publication and/or edition number or version number) or non-specific.
- For a specific reference, subsequent revisions do not apply.
- For a non-specific reference, the latest version applies.

FIPS-81, Federal Information Processing Standards Publication (FIPS PUB) 81 (1980): "DES Modes of Operation".

RFC 1750: "Randomness Recommendations for Security".

RFC 1889 (1996): "RTP: A Transport Protocol for Real-Time Applications".

RFC 1890: "RTP Profile for Audio and Video Conferences with Minimal Control".

RFC 2104 (1996): "HMAC: Keyed-Hashing for Message Authentication".

RFC 2139 (1997): "RADIUS Accounting".

RFC 2367 (1998): "PF_KEY Key Management API, Version 2".

RFC 2401 (1998): "Security Architecture for the Internet Protocol".

RFC 2406 (1998): "IP Encapsulating Security Payload (ESP)".

RFC 2407 (1998): "The Internet IP Security Domain of Interpretation for ISAKMP".

RFC 2409 (1998): "The Internet Key Exchange (IKE)".

RFC 2451 (1998): "The ESP CBC-Mode Cipher Algorithms".

RFC 2574 (1999): "User-based Security Model (USM) for version 3 of the Simple Network Management Protocol (SNMPv3)".

IETF draft (March 2001): "The Kerberos Network Authentication Service (V5)", http://www.ietf.org/internet-drafts/draft-ietf-cat-kerberos-revisions-08.txt.

ETSI ES 201 488: "Data-Over-Cable Service Interface Specifications Radio Frequency Interface Specification".

ETSI ES 200 800: "Digital Video Broadcasting (DVB); DVB interaction channel for Cable TV distribution systems (CATV)".

ETSI TS 101 909-2: "Access and Terminals (AT); Digital Broadband Cable Access to the Public Telecommunications Network; IP Multimedia Time Critical Services; Part 2: Architectural framework for the delivery of time critical services over cable Television networks using cable modems".

ETSI TS 101 909-4: "Access and Terminals (AT); Digital Broadband Cable Access to the Public Telecommunications Network; IP Multimedia Time Critical Services; Part 4: Network-Based Call Signalling Protocol".

ETSI TS 101 909-5: "Access and Terminals (AT); Digital Broadband Cable Access to the Public Telecommunications Network; IP Multimedia Time Critical Services; Part 5: Dynamic Quality of Service for the Provision of Real Time Services over Cable Television Networks using Cable Modems".

ETSI TS 101 909-6: "Access and Terminals (AT); Digital Broadband Cable Access to the Public Telecommunications Network; IP Multimedia Time Critical Services; Part 6: Media Terminal Adapter (MTA) device provisioning".

ETSI TS 101 909-7: "Access and Terminals (AT); Digital Broadband Cable Access to the Public Telecommunications Network; IP Multimedia Time Critical Services; Part 7: Management Information Base (MIB) Framework".

ITU-T Recommendation J.112: "Transmission systems for interactive television services".

ITU-T Recommendation X.500: "Information technology - Open Systems Interconnection - The Directory: Overview of concepts, models and services".

ITU-T Recommendation X.509: "Information technology - Open Systems Interconnection - The Directory: Public-key and attribute certificate frameworks".

ITU-T Recommendation X.520: "Information technology - Open Systems Interconnection - The directory: Selected attribute types".

PKCS #1 (1993): "RSA Encryption Standard", Version 1.5, RSA Laboratories.

PKCS #1 (1998): "RSA Cryptography Standard", Version 2.0, RSA Laboratories.

PKCS #7 (1993): "Cryptographic Message Syntax Standard", RSA Laboratories.

PKCS #9 (1993): "Selected Attribute Types", Version 1.1, RSA Laboratories.

Secure Hash Standard, Department of Commerce, NIST, FIPS 180-1, April, 1995.

# 3 Definitions and abbreviations

## 3.1 Definitions

For the purposes of the present document, the following terms and definitions apply:

**Access Control:** limiting the flow of information from the resources of a system only to authorized persons, programs, processes or other system resources on a network

**Access Node:** layer two termination device that terminates the network end of the ITU-T Recommendation J.112 connection

NOTE 1: It is technology specific. In ITU-T Recommendation J.112, annex A, it is called the INA while in annex B it is the CMTS.

**active:** J.112 Flow is said to be "active" when it is permitted to forward data packets

NOTE 2: A J.112 Flow must first be admitted before it is active.

**authentication:** process of verifying the claimed identity of an entity to another entity

**authenticity:** ability to ensure that the given information is without modification or forgery and was in fact produced by the entity who claims to have given the information

**authorization:** act of giving access to a service or device if one has the permission to have the access

**Cable Modem:** cable modem is a layer two termination device that terminates the customer end of the J.112 connection

**cipher:** algorithm that transforms data between plaintext and ciphertext

**ciphersuite:** set which must contain both an encryption algorithm and a message authentication algorithm (e.g. a MAC or an HMAC)

NOTE 3: In general, it may also contain a key management algorithm, which does not apply in the context of IPCablecom.

**confidentiality:** way to ensure that information is not disclosed to any one other then the intended parties

NOTE 4: Information is encrypted to provide confidentiality. Also known as privacy.

**downstream:** direction from the headend toward the subscriber location

**encryption:** method used to translate information in plaintext into ciphertext

**endpoint:** Terminal, Gateway or MCU

**IPCablecom:** ETSI working group project that includes an architecture and a series of Specifications that enable the delivery of real time services (such as telephony) over the cable television networks using cable modems

**Event Message:** message capturing a single portion of a connection

**gateway:** devices bridging between the IPCablecom IP Voice Communication world and the PSTN

NOTE 5: Examples are the Media Gateway which provides the bearer circuit interfaces to the PSTN and transcodes the media stream, and the Signalling Gateway which sends and receives circuit switched network signalling to the edge of the IPCablecom network.

**header:** protocol control information located at the beginning of a protocol data unit

**integrity:** way to ensure that information is not modified except by those who are authorized to do so

**Kerberos:** secret-key network authentication protocol that uses a choice of cryptographic algorithms for encryption and a centralized key database for authentication

**key:** mathematical value input into the selected cryptographic algorithm

**Key Exchange:** swapping of public keys between entities to be used to encrypt communication between the entities

**Key Management:** process of distributing shared symmetric keys needed to run a security protocol

**non-repudiation:** ability to prevent a sender from denying later that he or she sent a message or performed an action

**privacy:** way to ensure that information is not disclosed to any one other than the intended parties

NOTE 6: Information is usually encrypted to provide confidentiality. Also known as confidentiality.

**Private Key:** key used in public key cryptography that belongs to an individual entity and must be kept secret

**proxy:** facility that indirectly provides some service or acts as a representative in delivering information there by eliminating a host from having to support the services themselves

**Public Key:** key used in public key cryptography that belongs to an individual entity and is distributed publicly

NOTE 7: Other entities use this key to encrypt data to be sent to the owner of the key.

**Public Key Certificate:** binding between an entity's public key and one or more attributes relating to its identity, also known as a digital certificate

**Public Key Cryptography:** procedure that uses a pair of keys, a public key and a private key for encryption and decryption, also known as asymmetric algorithm

NOTE 8: A user's public key is publicly available for others to use to send a message to the owner of the key. A user's private key is kept secret and is the only key which can decrypt messages sent encrypted by the users public key.

**Root Private Key:** private signing key of the highest level Certification Authority

NOTE 9: It is normally used to sign public key certificates for lower-level Certification Authorities or other entities.

**Root Public Key:** public key of the highest level Certification Authority, normally used to verify digital signatures that it generated with the corresponding root private key

**X.509 certificate:** public key certificate specification developed as part of the ITU-T Recommendation X.500 standards directory

## 3.2 Abbreviations

For the purposes of the present document, the following abbreviations apply:

| | |
|---|---|
| AH | Authentication Header |
| AN | Access Node |
| BPI+ | Baseline Privacy Interface Plus |
| C7 | Signalling System Number 7 |
| CBC | Cipher Block Chaining Mode |
| CM | Cable Modem |
| CMS | Cryptographic Message Syntax |
| CMS | Call Management Server |
| DNS | Domain Name Server |
| DQoS | Dynamic Quality of Service |
| ESP | IPSec Encapsulation Security |
| FQDN | Fully Qualified Domain Name |
| HFC | Hybrid Fibre/Coaxial [cable] |
| HMAC | Hashed Message Authentication Code |
| INA | Interactive Network Adapter |
| IETF | Internet Engineering Task Force |
| IKE | Internet Key Exchange |
| IP | Internet Protocol |
| IPSec | Internet Protocol Security |
| LNP | Local Number Portability |
| MAC | Message Authentication Code |
| MAC | Media Access Control |
| MD5 | Message Digest 5 |
| MG | Media Gateway |
| MGC | Media Gateway Controller |
| MIB | Management Information Base |
| MTA | Media Terminal Adapter |
| NCS | Network Call Signalling |
| PKI | Public Key Infrastructure |
| PKINIT | Public Key Cryptography Initial Authentication |
| PSTN | Public Switched Telephone Network |
| QoS | Quality of Service |
| RADIUS | Remote Access Dial-In User Service |
| RC4 | a variable key length stream cipher offered in the ciphersuite, used to encrypt the media traffic in IPCablecom |
| RFC | Request for Comments |
| RKS | Record Keeping Server |
| RSVP | Resource Reservation Protocol |
| RTCP | Real Time Control Protocol |
| RTP | Real Time Protocol |

| | |
|---|---|
| SA | Security Association |
| SDP | Session Description Protocol |
| SG | Signalling Gateway |
| SHA-1 | Secure Hash Algorithm 1 |
| SIP | Session Initiation Protocol |
| SIP+ | Session Initiation Protocol Plus |
| SNMP | Simple Network Management Protocol |
| SPI | Security Parameters Index. |
| TCAP | Transaction Capabilities Application Protocol |
| TFTP | Trivial File Transfer Protocol |
| TGS | Ticket Granting Server |
| UDP | User Datagram Protocol |
| VoIP | Voice Over IP |

# 4 Void

# 5 Void

# 6 Architectural overview of IPCablecom security

## 6.1 IPCablecom Reference Architecture

Security requirements had been defined for every signalling and media link within the IPCablecom IP network. Thus, in order to understand the security requirements and Specifications for IPCablecom, one must first understand the overall architecture. This clause presents a brief overview of the IPCablecom architecture. For a more detailed Specification, refer to the IPCablecom Architecture Framework.

Figure 1, taken from the Architecture Framework, describes the IPCablecom system architecture:

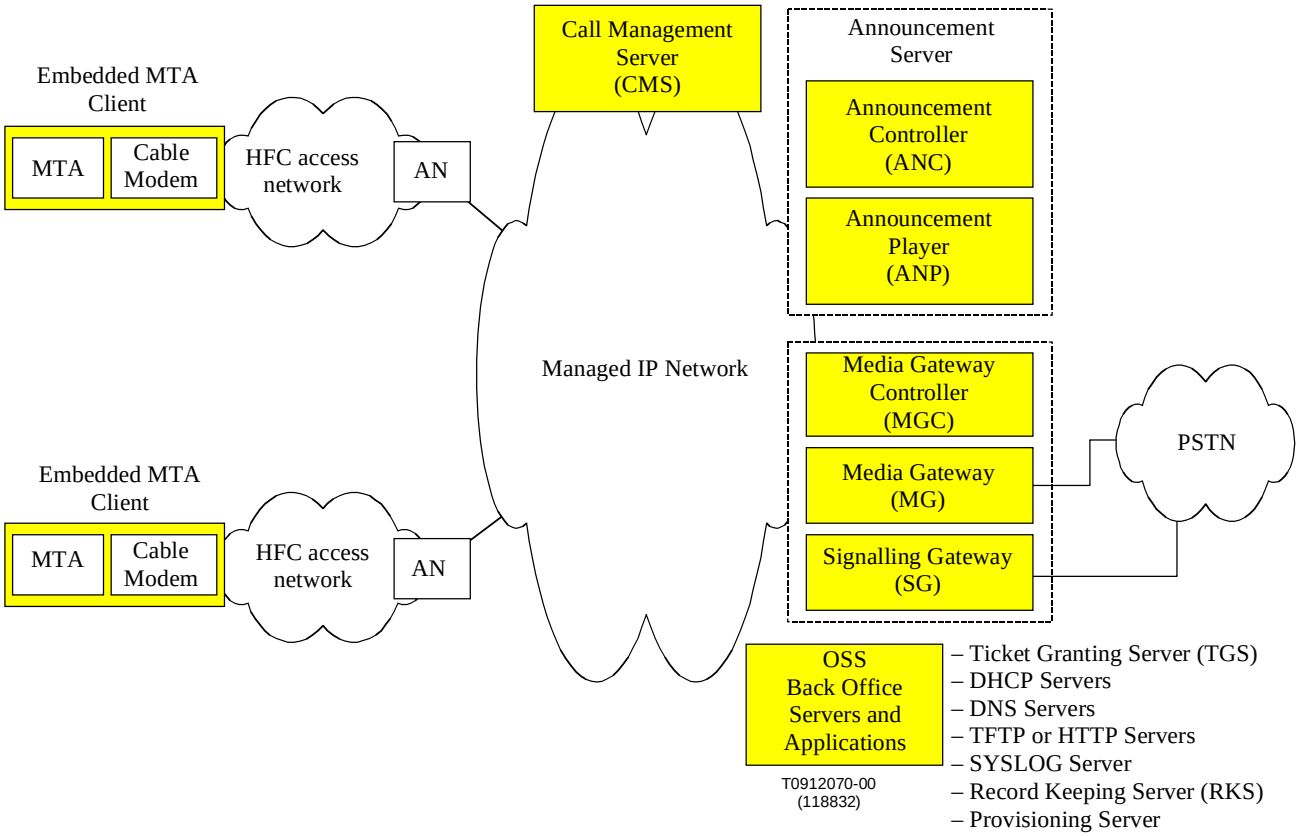



**Figure 1: IPCablecom Network Component Reference Model (partial)**

### 6.1.1 HFC Network

In figure 1, the Access Network between the MTAs and the AN is an HFC network, which employs J.112 physical layer and Media Access Control (MAC) layer protocols. J.112 MAC layer security and QoS protocols are enabled over this link.

There is signalling between the MTA and the AN, for the purpose of Dynamic QoS. This includes both custom User Datagram Protocol (UDP) messages defined specifically for IPCablecom DQoS, as well as standard Resource Reservation Protocol (RSVP) messages, which run directly over the IP layer in the protocol stack (see note).

NOTE: RSVP is not part of IPCablecom.

Since MAC layer security runs between the CM and the AN, it does not help in authenticating the identity of the MTA. Therefore, all protocols that run between the MTA and the AN have additional security requirements that cannot be met by MAC layer security.

### 6.1.2 Call Management Server

In the context of voice communications applications, a central component of the system is the Call Management Server (CMS). It is involved in both call signalling and the Dynamic Quality of Service (DQoS) establishment. The CMS also performs queries at the PSTN Gateway for LNP (Local Number Portability) and other services necessary for voice communications, including interface with the PSTN.

As described in the IPCablecom Architecture Framework J.160, the CMS is divided into the following functional components:

- Call Agent (CA) - The Call Agent maintains network intelligence and call state and controls the media gateway. Most of the time Call Agent is synonymous for Call Management Server.

- Gate Controller (GC) - The Gate Controller is a logical QoS management component that is typically part of the CMS. The GC coordinates all quality of service authorization and control on behalf of the application service - e.g. voice communications.
- Announcement Controller (ANC) - The ANC initiates and manages all announcement services provided by the Announcement Player. The ANC accepts requests from the CMS and arranges for the ANP to provide the announcement in the appropriate stream so that the user hears the announcement.
- Media Gateway Controller (MGC) - The Media Gateway Controller maintains the gateway's portion of call state for communications traversing the Gateway. A particular CMS can contain any subset of the above listed functional components.

### 6.1.3 Functional Categories

The IPCablecom Architecture Framework identifies two functional layers, a core service layer and an application layer, and the functional categories within each of these two layers.

**Core Service Layer:**

- Device and service provisioning.
- Quality of service (HFC access network and managed IP backbone).
- Security (specified herein).
- OSS functions and interfaces.
- Billing system interfaces.

**Application Layer:**

- Network call signalling (NCS).
- PSTN interconnectivity.
- CODEC functionality and media stream mapping.
- Announcement Servers.

In most cases, each functional category corresponds to a particular IPCablecom Specification document.

#### 6.1.3.1 Core Service Layer

##### 6.1.3.1.1 Device and Service Provisioning

During MTA provisioning and service provisioning (also called customer enrolment), the MTA gets its configuration with the help of the DHCP and Trivial File Transfer Protocol (TFTP) servers, as well as the OSS.

Provisioning interfaces need to be secured and have to configure the MTA with the appropriate security parameters (e.g. customer X.509 certificate signed by the Service Provider). The present document specifies the steps in both the MTA provisioning and customer enrolment, with the detailed Specifications given only to the security parameters. For a full Specification on MTA provisioning and customer enrolment, refer to TS 101 909-6.

##### 6.1.3.1.2 Quality of Service

IPCablecom provides a guaranteed Quality of Service (QoS) for each voice communication with Dynamic QoS or DQoS.

DQoS is controlled by the Gate Controller function within the CMS and can guarantee Quality of Service end-to-end. The Gate Controller utilizes the COPS protocol to download QoS policy into the AN. After that, the QoS reservation is established via layer 2 signalling or J.112 QoS between the MTA and the AN on both sides of the connection. QoS reservations are also forwarded to the IP Backbone between the ANs, but the Specifications of the Backbone reservations are outside of the scope of IPCablecom. Therefore, the corresponding security Specifications are also out of scope.

#### 6.1.3.1.3 OSS Functions and Interfaces

The OSS system is used to provision and configure each MTA and the corresponding subscriber. The dynamic configuration of an MTA occurs via the SNMPv3 protocol.

During the MTA and subscriber provisioning, the MTA gets its configuration from the DHCP and TFTP servers.

Interfaces between the MTA and the OSS, Provisioning and TFTP servers all need to be protected and the corresponding security requirements and Specifications are defined in the present document.

There are additional interfaces between the OSS system and the DHCP and TFTP servers, as well as the CMS. IPCablecom currently considers these interfaces out of scope. The present document defines security requirements and some recommended solutions for these links, but considers the corresponding security Specifications out of scope.

#### 6.1.3.1.4 Billing System Interfaces

The CMS, AN and the PSTN Gateway are all required to send out billing event messages to the RKS - Record Keeping Server. This interface is currently specified to be Radius. Billing information should be checked for integrity and authenticity as well as kept private. The present document defines security requirements and Specifications for the communication with RKS.

### 6.1.3.2 Application Layer

#### 6.1.3.2.1 Call Signalling

The call signalling architecture defined within IPCablecom is Network Call Signalling (NCS). The Call Management Server (CMS) is used to control call set-up, termination and most other call signalling functions. In the NCS architecture (J.162), the Call Agent function within the CMS is used in call signalling, and it utilizes the MGCP protocol.

#### 6.1.3.2.2 PSTN Interconnectivity

The PSTN interface to the voice communications capabilities of the IPCablecom network is through the Signalling and Media Gateways (SG and MG). Both of these gateways are controlled with the MGC (Media Gateway Controller). The MGC may be standalone or combined with a CMS.

All communications between the MGC and the SG and MG may be over the same shared IP network and is subject to similar threats (e.g. privacy, masquerade, denial of service) that are encountered in other links in the same network. The present document defines both security requirements and Specifications for the PSTN Gateway links.

When communications from an MTA to a PSTN phone is made, bearer channel traffic is passed directly between an MTA and a MG. The protocols used in this case are Real Time Protocol (RTP) and Real Time Control Protocol (RTCP), the same as in the MTA-to-MTA case. Both security requirements and Specifications are very similar to the MTA-to-MTA bearer requirements and are fully defined in the present document. After a voice communication enters the PSTN, the security requirements as well as Specifications are based on existing PSTN standards and are out of scope of the present document.

#### 6.1.3.2.3 CODEC Functionality and Media Stream Mapping

The media stream between two MTAs or between an MTA and a PSTN Gateway utilizes the RTP protocol. Although MAC layer security provides privacy over the HFC network, the potential threats within the rest of the voice communications network require that the RTP packets be encrypted end-to-end (see note).

In addition to RTP, there is an accompanying RTCP protocol, primarily used for reporting of RTCP statistics. In addition, RTCP packets may carry CNAME - a unique identifier of the sender of RTP packets. RTCP also defines a BYE message (see note) that can be used to terminate an RTP session. These two additional RTCP functions raise privacy and denial of service threats. Due to these threats, RTCP security requirements are the same as the requirements for all other end-to-end Session Initiation Protocol Plus (SIP+) signalling and are addressed in the same manner.

NOTE: In general, it is possible for an MTA-to-MTA or MTA-to-PSTN connection to cross networks of several different Service Providers. In the process, this path may cross a PSTN network. This is an exception to the rule, where all RTP packets are encrypted end-to-end. The media traffic inside an PSTN network does not utilize RTP and has its own security requirements. Thus, in this case the encryption would not be end-to-end and would terminate at the PSTN Gateway on both sides of the intermediate PSTN network.

In addition to MTAs and PSTN Gateways, Media Servers may also participate in the media stream flows. Media Servers represent network-based components that operate on media flows to support various voice communications service options. Media servers perform audio bridging, play

terminating announcements, provide interactive voice response services, etc. Both media stream and signalling interfaces to a Media Server are the same as the interfaces to an MTA (both in the cases of DCS and NCS signalling).

#### 6.1.3.2.4 Announcement Server Interfaces

For future study.

## 6.2 Threats

Figure 2 contains the interfaces that were analysed for security in IPCablecom.

There are additional interfaces that are identified in IPCablecom but for which protocols are not specified. In those cases, the corresponding security protocols are also not specified, and those interfaces are not listed in figure 2.

There are also interfaces to the DHCP and Domain Name Server (DNS) servers for which security is not required in IPCablecom. Those interfaces are also not listed in figure 2.

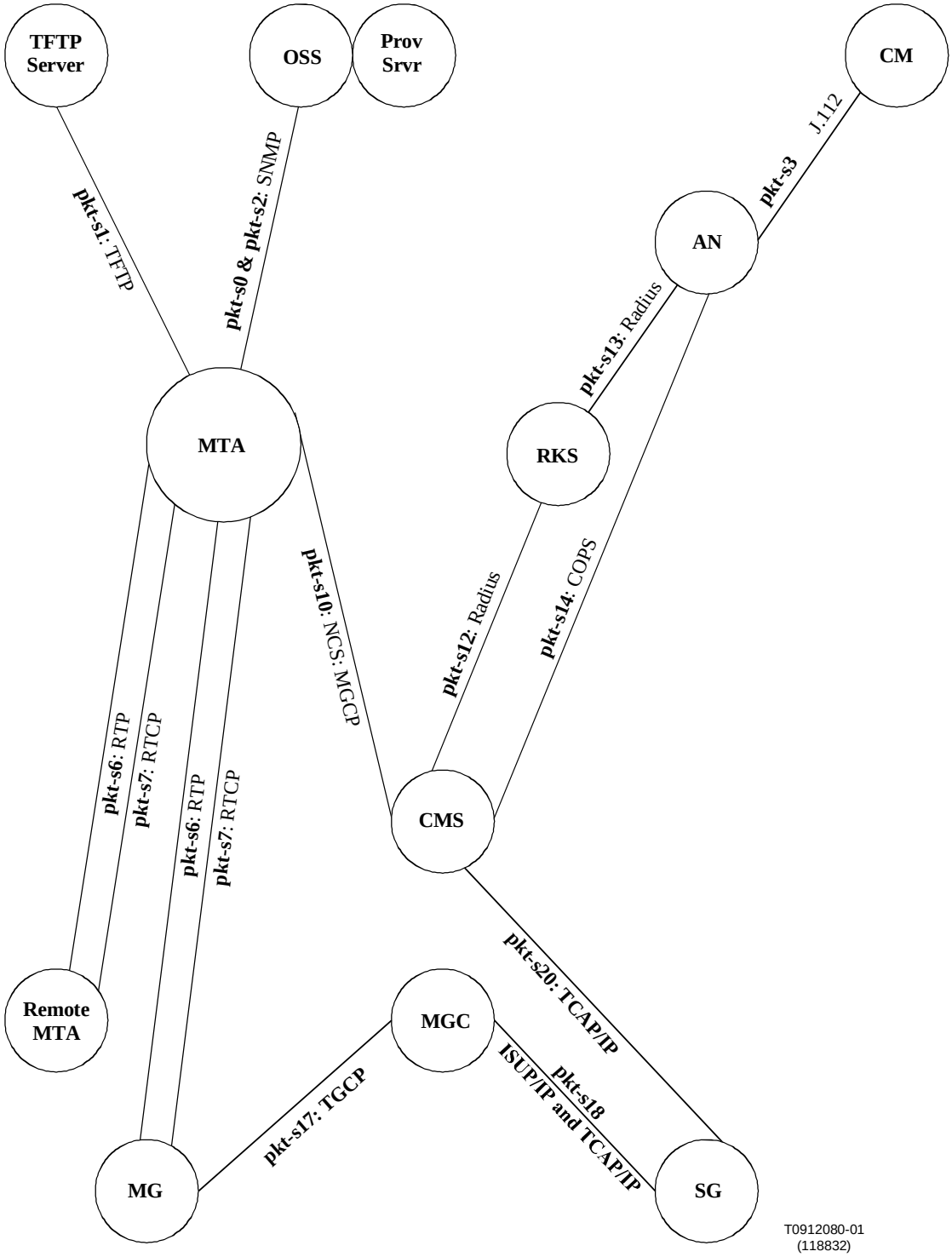



**Figure 2: Subset of IPCablecom Interfaces**

Below is a summary of general threats and the corresponding attacks that are relevant in the context of IP voice communications. This list of threats is not based on the knowledge of the specific protocols or security mechanisms employed in the network. A more specific summary of threats that are based on the functionality of each network element is listed in clause 6.2.6.

Some of the outlined threats cannot be addressed purely by cryptographic means - physical security and/or fraud management should also be used. These threats may be important, but cannot be fully addressed within the scope of IPCablecom. How each vendor and each operator implement fraud management and physical security will differ and in this case a Specification is not required for interoperability.

## 6.2.1 Theft of Service

### 6.2.1.1 Theft of Network Services

In the context of voice communications, the main services that may be stolen are:

- Long-distance service.
- Local (subscription) voice communications service.
- Videoconferencing.
- Network-based three-way calling.
- Quality of Service.

#### 6.2.1.1.1 Attacks

##### 6.2.1.1.1.1 MTA Clones

One or more MTAs can masquerade as another MTA by duplicating its permanent identity and keys. The secret cryptographic keys may be obtained by either breaking the physical security of the MTA or by employing cryptoanalysis.

When an MTA is broken into by someone other than the owner, the perpetrator can steal voice communications service and charge it all to the original owner. The feasibility of such an attack depends on where an MTA is located. This attack must be seriously considered in the cases when an MTA is located in an office or apartment building, or on a street corner.

An owner might break into his or her own MTA in at least one instance - after a false account with the operator providing the voice communications service had been set up. The customer name, address, or account number may all be invalid or belong to someone else. The provided Credit Card Number may be stolen. In that case, the owner of the MTA would not mind giving out the MTA cryptographic identity to others - he or she would not have to pay for service anyway.

In addition to cloning of the permanent cryptographic keys, temporary (usually symmetric) keys may also be cloned. Such an attack is more complex, since the temporary keys expire more often and have to be frequently redistributed. The only reason why someone would attempt this attack is if the permanent cryptographic keys are protected much better than the temporary ones, or if the temporary keys are particular easy to steal or discover with cryptoanalysis.

##### 6.2.1.1.1.2 Other Clones

It is conceivable, that the cryptographic identity of another network element, such as a AN or a CMS may be cloned. Such an attack is most likely to be mounted by an insider - a corrupt or disgruntled employee.

##### 6.2.1.1.1.3 Subscription Fraud

A customer sets up an account under false information.

##### 6.2.1.1.1.4 Non-Payment for Voice Communications Services

A customer stops paying his or her bill, but continues to use the MTA for voice communications service. This can happen if the network does not have an automated way to revoke customer's access to the network.

##### 6.2.1.1.1.5 Protocol Attacks against an MTA

A weakness in the protocol can be manipulated to allow an MTA to authenticate to a network server with a false identity or hijack an existing voice communication. This includes replay and man-in-the-middle attacks.

#### 6.2.1.1.1.6 Protocol Attacks against Other Network Elements

A perpetrator might employ similar protocol attacks to masquerade as a different Network Element, such as a AN or a CMS. Such an attack may be used in collaboration with the cooperating MTAs to steal service.

### 6.2.1.2 Theft of Services Provided by the MTA

Services such as the support for multiple MTA ports, three-way calling and call waiting may be implemented entirely in the MTA, without any required interaction with the network.

#### 6.2.1.2.1 Attacks

MTA code to support these services may be downloaded illegally by an MTA clone, in which case the clone has to interact with the network to get the download. In that case, this threat is no different from the network service theft described in the previous clause.

Alternatively, downloading an illegal code image using some illegal out-of-band means can also enable these services. Such service theft is much harder to prevent (a secure software environment within the MTA may be required). On the other hand, in order for an adversary to go through this trouble, the price for these MTA-based services has to make the theft worthwhile.

An implication of this threat is that valuable services cannot be implemented entirely inside the MTA without a secure software environment in addition to tamperproof protection for the cryptographic keys.

NOTE: While a secure software environment within an MTA adds significant complexity, it is an achievable task.

### 6.2.1.3 MTA Moved to Another Network

A leased MTA may be reconfigured and registered with another network, contrary to the intent and property rights of the leasing company.

## 6.2.2 Bearer Channel Information Threats

This class of threats is concerned with the breaking of privacy of voice communications over the IP bearer channel. Threats against non-VoIP communications are not considered here and assumed to require additional security at the application layer.

### 6.2.2.1 Attacks

Clones of MTAs and other Network Elements, as well as protocol manipulation attacks, also apply in the case of the Bearer Channel Information threats. These attacks were already described under the Service Theft threats.

MTA cloning attacks mounted by the actual owner of the MTA are less likely in this case, but not completely inapplicable. An owner of an MTA may distribute clones to the unsuspecting victims, so that he or she can later spy on them.

#### 6.2.2.1.1 Off-line Cryptanalysis

Bearer channel information may be recorded and then analysed over a period of time, until the encryption keys are discovered with cryptoanalysis. The discovered information may be of value even after a relatively long time has passed.

## 6.2.3 Signalling Channel Information Threats

Signalling information, such as the caller identity and the services to which each customer subscribes may be collected for marketing purposes. The caller identity may also be used illegally to locate a customer that wishes to keep his or her location private.

#### 6.2.3.1 Attacks

Clones of MTAs and other Network Elements, as well as protocol manipulation attacks, also apply in the case of the Signalling Channel Information threats. These attacks were already described under the Service Theft threats.

MTA cloning attacks mounted by the actual owner of the MTA are theoretically possible in this case. An owner of an MTA may distribute clones to the unsuspecting victims, so that he or she can monitor their signalling messages (e.g. for information with marketing value). The potential benefits of such an attack seem unjustified, however.

##### 6.2.3.1.1 Caller ID

A number of a party initiating a voice communication is revealed, even though a number is not generally available (i.e. is "unlisted") and the owner of that number enabled ID blocking.

##### 6.2.3.1.2 Information with Marketing Value

Dialled numbers and the type of service customers use may be gathered for marketing purposes by other corporations.

### 6.2.4 Service Disruption Threats

This class of threats is aimed at disrupting the normal operation of the voice communications. The motives for denial of service attacks may be malicious intent against a particular individual or against the service provider. Or, perhaps a competitor wishes to degrade the performance of another service provider and use the resulting problems in an advertising campaign.

#### 6.2.4.1 Attacks

##### 6.2.4.1.1 Remote Interference

A perpetrator is able to manipulate the protocol to close down ongoing voice communications. This might be achieved by masquerading as an MTA that is involved in such an ongoing communication. The same effect may be achieved if the perpetrator impersonates another Network Element, such as a Gate Controller or an Edge Router during either call set-up or voice packet routing.

Depending on the signalling protocol security, it might be possible for the perpetrator to mount this attack from the MTA, in the privacy of his or her own home.

Clones of MTAs and other Network Elements, as well as protocol manipulation attacks, also apply in the case of the Service Disruption threats. These attacks were already described under the Service Theft threats.

MTA cloning attacks mounted by the actual owner of the MTA can be theoretically used in service disruption against unsuspecting clone owners. However, since there are so many other ways to cause service disruption, such an attack cannot be taken seriously in this context.

### 6.2.5 Repudiation

In a network where masquerade (using the above-mentioned cloning and protocol manipulation techniques) is common or is easily achievable, a customer may repudiate a particular communication (and, e.g. deny responsibility for paying for it) on that basis.

In addition, unless public key-based digital signatures are employed on each message, the source of each message cannot be absolutely proven. If a signature over a message that originated at an MTA is based on a symmetric key that is shared between that MTA and a network server (e.g. the CMS), it is unclear if the owner of the MTA can claim that the Service Provider somehow falsified the message.

However, even if each message were to carry a public key-based digital signature and if each MTA were to employ stringent physical security, the customer can still claim in court that someone else initiated that communication without his or her knowledge, just as a customer of a telecommunications carrier on the PSTN can claim, e.g. that particular long-distance calls made from the customer's telephone were not authorized by the customer. Such telecommunications carriers commonly address this situation by establishing contractual and/or tariffed relationships with customers in which customers assume liability for unauthorized use of the customer's service. These same contractual principles are typically implemented in service contracts between information services providers such as ISPs and their subscribers. For these reasons, the benefits of non-repudiation seem dubious at best and do not appear to justify the performance penalty of carrying a public key-based digital signature on every message.

### 6.2.6 Threat Summary

This clause provides a summary of the above of threats and attacks and a brief assessment of their relative importance.

- Primary Threats
  - **Theft of Service**. Attacks are:
    - **Subscription Fraud**. This attack is prevalent today's telephony systems (i.e. the PSTN) and requires little economic investment. It can only be addressed with a Fraud Management system.
    - **Non-payment for services**. Within the PSTN, telecommunications carriers usually do not prosecute the offenders, but simply shut down their accounts. Because prosecution is expensive and not always successful, it is a poor counter to this attack. Methods such as debit-based billing and device authorization (pay as you play), increasingly common in the wireless sector of the PSTN, might be a possible solution for this attack in the IPCablecom context. This threat can also be minimized with effective Fraud Management systems.
    - **MTA clones**. This threat requires more technical knowledge than the previous two threats. A technically knowledgeable adversary or underground organization might offer cloning services for profit. This threat is most effective, when combined with subscription fraud, where an MTA registered under a fraudulent account is cloned. This threat can be addressed with both Fraud Management and physical security inside the MTA, or a combination of both.
    - **Impersonate a network server.** With proper cryptographic mechanisms, authorization and procedural security in place, this attack is unlikely, but has a potential for great damage.
    - **Protocol manipulation.** Can occur only when security protocols are flawed or when not enough cryptographic strength is in place.
  - Bearer Channel Information Disclosure. Attacks are:
    - **Simple Snooping**. This would happen if voice packets were sent in the clear over some segment of the network. Even if that segment appears to be protected, it may still be compromised by an insider. This is the only major attack on privacy. The bearer channel privacy attacks listed below are possible but are all of secondary importance.
    - **MTA clones**. Again, this threat requires more technical knowledge but can be offered as a service by an underground organization. A most likely variation of this attack, is when a publicly accessible MTA (e.g. in an office or apartment building) is cloned.
    - **Protocol manipulation**. A flawed protocol may somehow be exploited to discover bearer channel encryption keys.
    - **Off-line cryptoanalysis**. Even when media packets are protected with encryption, they can be stored and analysed for long periods of time, until the decryption key is finally discovered. Such an attack is not likely to be prevalent, since it is justified only for particularly valuable customer-provided information (IPCablecom security is not required to protect data). This attack is more difficult to perform on voice packets (as opposed to data). Still, customers are very sensitive to this threat and it can serve as the basis for a negative publicity campaign by competitors.

- **Signalling Information Disclosure.** This threat is listed as primary only due to potential for bad publicity and customer sensitivity to keeping their numbers and location private. All of the attacks listed below are similar to the ones for bearer channel privacy and are not described here:
  - Simple snooping.
  - MTA clones.
  - Protocol manipulation.
  - Off-line cryptoanalysis.
  - Service disruption.
  - Secondary Threats
- **Theft of MTA-based services**. Based on the voice communications services that are planned for the near future, this threat does not appear to have potential for significant economic damage. This could possibly change with the introduction of new value-added services in the future.
- **Illegally registering a leased MTA with a different Service Provider**. Leased MTAs can normally be tracked. Most likely, this threat is combined with the actual theft of a leased MTA. Thus, this threat does not appear to have potential for widespread damage.

## 6.3 Security Architecture

### 6.3.1 Overview of Security Interfaces

Figure 3 summarizes all of the IPCablecom security interfaces, including key management.

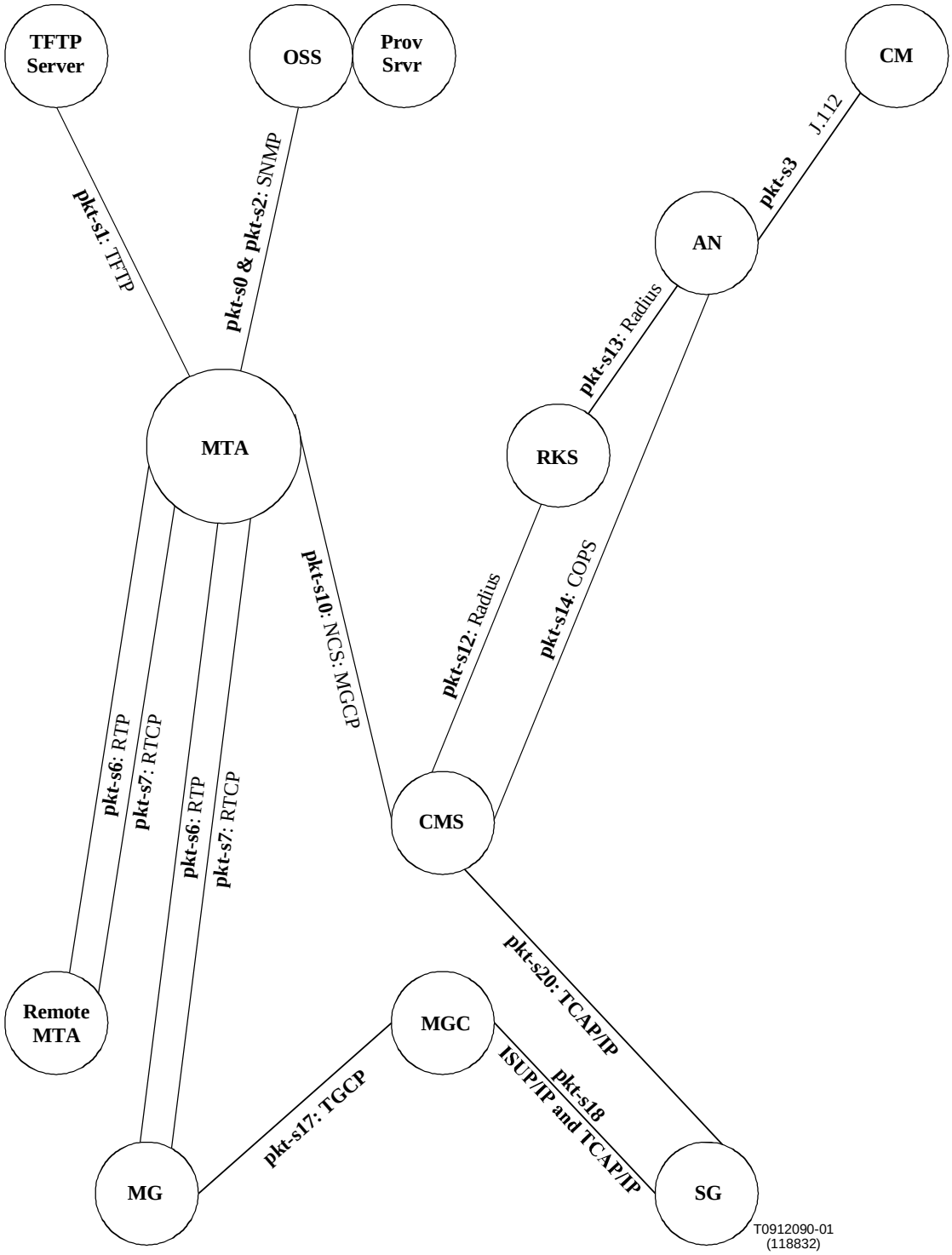



**Figure 3: IPCablecom Security Interfaces**

In figure 3, each interface label looks like:

<label>: <protocol> { <security protocol>/<key management protocol> }

If the key management protocol is missing, that means it is not needed for that interface. IPCablecom interfaces that do not require security are not shown on this figure.

The following abbreviations were used in the above figure:

IKE– IKE with pre-shared keys.

IKE+ IKE requires public key certificates.
CMS-based KM Keys randomly generated and distributed by CMS.

Here is a table that briefly describes each of the interfaces shown in figure 3.

NOTE: The number is not consecutive because interfaces that are out of scope for IPCablecom are not included here.

**Table 1: Summary of Security Interfaces**

| Interface | Description |
|---|---|
| pkt-s0 | SNMPv3 INFORM from the MTA to the Simple Network Management Protocol (SNMP) Manager, followed by optional SNMP GET(s) by the SNMP Manager are used to query MTA device capabilities. This occurs at the time where SNMPv3 keys may not be available, and security is provided with an RSA signature, formatted according to CMS (Cryptographic Message Syntax). |
| pkt-s1 | MTA Configuration file download. The MTA downloads a configuration file (with TFTP-get) that is signed by the TFTP server and sealed with the MTA public key, with a CMS (Cryptographic Message Syntax) wrapper. This flow occurs right after an SNMPv3 INFORM followed by an optional SNMP GET(s) - see flow pkt-s0. |
| pkt-s2 | Standard SNMPv3 security. The SNMPv3 keys are downloaded with the MTA configuration file, using interface pkt-s1. |
| pkt-s3 | MAC layer security on the HFC link. Both security and key management are defined by J.112. |
| pkt-s6 | End-to-end media packets between two MTAs, or between MTA and MG. RTP packets are encrypted directly with RC4, without any additional security layers. Message integrity is optionally provided by an MMH-based MAC (Message Authentication Code). Keys are distributed by the CMS to the two endpoints. |
| pkt-s7 | RTCP control protocol for RTP, defined above. Message integrity and encryption provided with IPSEC. Key management is same as for RTP - keys are distributed by CMS. |
| pkt-s10 | MTA-CMS signalling for NCS. Message integrity and privacy via IPSEC. Key management is with Kerberos with PKINIT (public key initial authentication) extension. |
| pkt-s12 | Radius billing events sent by the CMS to the RKS. Radius authentication keys are hardcoded to 0. Instead, IPSEC is used for message integrity, as well as privacy. Key management is IKE. |
| pkt-s13 | Radius events sent by the AN to the RKS. Radius authentication keys are hardcoded to 0. Instead, IPSEC is used for message integrity, as well as privacy. Key management is IKE. |
| pkt-s14 | COPS protocol between the GC and the AN, used to download QoS authorization to the AN. Message integrity and privacy provided with IPSEC. Key management is IKE. |
| pkt-s15 | Gate Coordination messages for DQoS. Message integrity is provided with an application-layer (Radius) authenticator. Keys are distributed by local CMS over COPS. |
| pkt-s17 | IPCablecom interface to the PSTN Media Gateway. IPSEC is used for both message integrity and privacy. Key management is IKE. |
| pkt-s18 | IPCablecom interface to the PSTN Signalling Gateway. IPSEC is used for both message integrity and privacy. Key management is IKE. |
| pkt-s19 | Kerberos/PKINIT key management protocol, where the TGS issues CMS tickets to the MTAs. |
| pkt-s20 | CMS queries the PSTN Gateway for LNP (Local Number Portability) and other voice communications services. IPSEC is used for both message integrity and privacy. Key management is IKE. |

## 6.3.2 Security Assumptions

### 6.3.2.1 AN Downstream Messages Are Trusted

As mentioned previously, it is assumed that AN downstream messages cannot be easily modified in transit and an AN can be impersonated only at a great expense.

Most messages secured in the present document either transit over the shared IP network in addition to the J.112 path, or do not go over J.112 at all. In those cases, this assumption does not apply.

#### 6.3.2.2 Non-Repudiation Not Supported

Non-repudiation service means that a sender of a message cannot deny that he or she sent that message. Public key cryptography and digital signatures make non-repudiation possible, but at the expense of significant overhead.

In this voice communications architecture addressed in the present document, non-repudiation is not supported for most messages, with the exception of the top key management layer and the MTA code download. This decision was based on the performance penalty that is incurred with each public key operation. The most important use for non-repudiation would have been during communications set-up - to prove that a particular party had initiated that particular communication. However, due to very strict requirements on the set-up time, it is not possible to perform public key operations for each communication.

#### 6.3.2.3 Root Private Key Will Not Be Compromised

The cryptographic mechanisms defined in the present document are based on a Public Key Infrastructure (PKI). As is the case with most other architectures that are based on a PKI, there is no automated recovery path from a compromise of a Root Private Key.

The corresponding Root Public Key is stored as a read-only parameter in many components of this architecture. Once the corresponding Root Private Key is compromised, each element must be manually reconfigured.

Due to this limitation of a PKI, the Root Private Key must be very carefully guarded with procedural and physical security. And, it must be sufficiently long, so that its value cannot be discovered with cryptographic attacks within the expected lifetime of the system.

#### 6.3.2.4 Root Public Keys Will Not Be Compromised

This PKI limitation is closely related to the one in the previous clause. As already mentioned, most components in the system include a Root Public Key as a read-only parameter. The security of the public key is also very important. While its value is well known, it must be very carefully guarded (with physical and procedural security) against modification.

Once the value of a Root Public Key in a particular network element is illegitimately modified, it cannot reliably authenticate any other element, and there is no automated way to recover. The compromised Network Element must be reconfigured again with a proper Root Public Key through some manual means.

#### 6.3.2.5 Limited Prevention of Denial of Service Attacks

The present document does not attempt to address all or even most of denial of service attacks. The cryptographic mechanisms defined in the present document prevent some denial of service attacks that are particularly easy to mount and are hard to detect. For example, they will prevent a compromised MTA from masquerading as other MTAs in the same upstream HFC segment and interrupting ongoing communications with illicit HANGUP messages.

The present document will also prevent more serious denial of service attacks, such as an MTA masquerading as a CMS in a different network domain that causes all communications set-up requests to fail.

On the other hand, denial of service attacks where a router is taken out of service or is bombarded with bad IP packets are not addressed. In general, denial of service attacks that are based on damaging one of the network components can only be solved with procedural and physical security, which is out of scope of the present document.

Denial of service attacks where network traffic is overburdened with bad packets cannot be prevented in a large network (although procedural and physical security helps), but can usually be detected. Detection of such an attack and of its cause is out of scope of the present document.

For example, denial of service attacks where a router is taken out of order or is bombarded with bogus IP packets cannot be prevented.

### 6.3.3 Susceptibility of Network Elements to Attack

This clause describes the amount and the type of trust that can be assumed for each element of the voice communications network. It also describes specific threats that are possible if each network component is compromised. These threats are based on the functionality specified for each component. The general categories of threats were described in clause 6.2.

Both the trust and the specific threats are described with the assumption that no cryptographic or physical security has been employed in the system, with the exception of the MAC layer security that is assumed on the J.112 links. The goal of this security Specification is to address threats that are relevant to this voice communications system.

#### 6.3.3.1 Managed IP Network

It is assumed that the same IP network may be shared between multiple, possibly competing service providers. It is also assumed that the service provider may provide multiple services on the same IP network, e.g. Internet connectivity. No assumptions can be made about the physical security of each link in this IP network. An intruder can pop up at any location with the ability to monitor traffic, perform message modification and to reroute messages.

#### 6.3.3.2 MTA

The MTA is considered to be an untrusted network element. It is operating inside customer premises, considered to be a hostile environment. It is assumed that a hostile adversary has the ability to open up the MTA and make software and even hardware modifications to fit his or her needs, and this would be done in the privacy of the customer's home.

The MTA communicates with the AN over the shared J.112 path and has access to downstream and upstream messages from other MTAs within the same HFC segment.

An MTA is responsible for:

- Initiating and receiving communications to/from another MTA or the PSTN.
- Negotiating QoS.

A compromise of an MTA can result in:

- MTA clones that are capable of:
  - charging services to someone else's account;
  - violating privacy;
  - identity fraud;
  - illegally downloading a code image that enables enhanced features within the MTA.
- MTA running a bad code image that disrupts communications made by other MTAs or degrades network performance.

#### 6.3.3.3 AN

The AN communicates both over the J.112 path and over the shared IP network. When the AN sends downstream messages over the J.112 path, it is assumed that a perpetrator cannot modify them or impersonate the AN. And, MAC layer security over that path provides privacy.

However, when the AN is communicating over the shared IP network (e.g. with the CMS or another AN), no such assumptions can be made.

While the AN, as well as voice communications network servers are more trusted than the MTAs, they cannot be trusted completely. There is always a possibility of an insider attack.

Insider attacks at the AN should be addressed by cryptographic authentication and authorization of the AN operators, as well as by physical and procedural security, which are all out of scope of the IPCablecom Specifications.

An AN is responsible for:

- Reporting billing related statistics to the RKS.
- QoS allocation for MTAs over the J.112 path.
- Implementation of MAC layer security and corresponding key management.

A compromise of an AN may result in:

- Service theft by reporting invalid information to the RKS.
- Unauthorized levels of QoS.
- Loss of privacy, since the AN holds MAC layer security keys. This may not happen, if additional encryption is provided above the MAC layer.
- Degraded performance of some or all MTAs in that HFC segment.
- Some or all of the MTAs in one HFC segment completely taken out of service.

#### 6.3.3.4 Voice Communications Network Servers

Servers used for voice communications (e.g. CMS, RKS, Provisioning, OSS, DHCP and TFTP Servers) reside on the network and can potentially be impersonated or subjected to insider attacks. The main difference would be in the damage that can be incurred in the case a particular server is impersonated or compromised.

Threats that are associated with each network element are discussed in the following clauses. To summarize those threats, a compromise or impersonation of each of these servers can result in a wide-scale service theft, loss of privacy, and in highly damaging denial of service attacks.

In addition to authentication of all messages to and from these servers (specified in the present document), care should be taken to minimize the likelihood of insider attacks. They should be addressed by cryptographic authentication and authorization of the operators, as well as by stringent physical and procedural security, which are all out of scope of the IPCablecom Specifications.

##### 6.3.3.4.1 CMS

The Call Management Server is responsible for:

- Authorizing individual voice communications by subscribers.
- QoS allocation.
- Initializing the billing information in the AN.
- Distributing per communication keys for MTA-MTA signalling, bearer channel, and DQoS messages on the MTA-AN and AN-AN links.
- Interface to PTSN gateway.

A compromised CMS can result in:

- Free voice communications service to all of the MTAs that are located in the same network domain (up to 100 000). This may be accomplished by:
  - allowing unauthorized MTAs to create communications;
  - downloading invalid or wrong billing information to the AN;
  - combination of both of the above.
- Loss of privacy, since the CMS distributes a bearer channel keys.
- Unauthorized allocation of QoS.
- Unauthorized disclosure of customer identities, location (e.g. IP address), communication patterns and a list of services to which the customer subscribes.

#### 6.3.3.4.2 RKS

The RKS is responsible for collecting billing events and reporting them to the billing system. A compromised RKS may result in:

- Free or reduced-rate service due to improper reporting of statistics.
- Billing to a wrong account.
- Billing customers for communications that were never made, i.e. fabricating communications.
- Unauthorized disclosure of customer identities, personal information, service usage patterns and a list of services to which the customer subscribes.

#### 6.3.3.4.3 OSS, DHCP & TFTP Servers

The OSS system is responsible for:

- MTA and service provisioning.
- MTA code downloads and upgrades.
- Handling service change requests and dynamic reconfiguration of MTAs.

A compromise of the OSS, DHCP or TFTP server can result in:

- MTAs running illegal code, which may:
  - intentionally introduce bugs or render MTA completely inoperable;
  - degrade voice communications performance on the IPCablecom network or on the HFC network;
  - enable the MTA with features to which the customer is not entitled.
- MTAs configured with an identity and keys of another customer.
- MTAs configured with service options for which the customer did not pay.
- MTAs provisioned with a bad set of parameters that would make them perform badly or not perform at all.

### 6.3.3.5 PSTN Gateways

#### 6.3.3.5.1 Media Gateway

The MG is responsible for:

- Passing media packets between the IPCablecom network and the PSTN.
- Reporting statistics to the RKS.

A compromise of the MG may result in:

- Service theft by reporting invalid information to the RKS.
- Loss of privacy on communications to/from the PSTN.

#### 6.3.3.5.2 Signalling Gateway

The SG is responsible for:

- Translating call signalling between the IPCablecom network and the PSTN.

A compromise of the SG may result in:

- Incorrect MTA identity reported to the PSTN.

- Unauthorized services enabled within the PSTN.
- Loss of PSTN connectivity.
- Unauthorized disclosure of customer identities, location (e.g. IP address), usage patterns and a list of services to which the customer subscribes.

# 7 Security Mechanisms

## 7.1 IPSec

### 7.1.1 Overview

IPSEC provides network layer security that runs immediately above the IP layer in the protocol stack. It provides security for the TCP or UDP layer and above. It consists of two protocols: Internet Protocol Security Encapsulation Security Payload (IPSEC ESP) and Internet Protocol Security Authentication Header (IPSEC AH), as specified in RFC 2401.

IPSEC ESP provides both confidentiality and message integrity, IP header not included. IPSEC AH provides only message integrity, but that includes most of the IP header (with the exception of some IP header parameters that can change with each hop). IPCablecom utilizes only the IPSEC ESP protocol (RFC 2406) since authentication of the IP header does significantly improve security within the IPCablecom architecture.

In general, IPSEC ESP may run in tunnel mode, where it is applied to all traffic that traverses a particular path between two hosts. In this mode, security is not applied end-to-end. It can also run in transport mode, where a Security Association (SA) is established end-to-end. IPCablecom security architecture utilizes only end-to-end Security Associations and thus only the IPSEC ESP transport mode. For more details on the two IPSEC ESP modes, refer to RFC 2406.

### 7.1.2 IPCablecom Profile for IPSEC ESP (Transport Mode)

#### 7.1.2.1 IPSEC ESP Transform Identifiers

IPSEC Transform Identifier (1 byte) is used by IKE to negotiate an encryption algorithm that is used by IPSEC. A list of available IPSEC Transform Identifiers is specified in RFC 2407. Within IPCablecom, the same Transform Identifiers are used by all IPSEC key management protocols: IKE, Kerberos and application layer (embedded in IP signalling messages).

For IPCablecom, the following IPSEC Transform Identifiers are supported (all of which are specified in RFC 2451):

**Table 2: IPSEC ESP Transform Identifiers**

| Transform ID | Value | Key Size (in bits) | Support REQUIRED | Description |
|---|---|---|---|---|
| ESP_3DES | 3 | 192 | yes | 3-DES in CBC mode. |
| ESP_RC5 | 4 | 128 | no | RC5 in CBC mode. |
| ESP_IDEA | 5 | 128 | no | IDEA in CBC mode. |
| ESP_CAST | 6 | 128 | no | CAST in CBC mode. |
| ESP_BLOWFISH | 7 | 128 | no | BLOWFISH in CBC mode. |
| ESP_NULL | 11 | 0 | yes | Encryption turned off. |

The ESP_3DES and ESP_NULL Transform Ids MUST be supported. For all of the above transforms, the cipher block chaining (CBC) Initialization Vector (IV) is carried in the clear inside each ESP packet payload.

IKE allows the negotiation of the encryption key size. Other IPSEC Key Management protocols used by IPCablecom do not allow key size negotiation, and so for consistency a single key size is listed for each Transform ID. If in the future it is desired to increase the key size for one of the above algorithms, IKE will use the built in key size negotiation, while other key management protocols will utilize a new Transform ID for the larger key size.

Security Specification for each IPCablecom interface lists whether or not encryption (confidentiality) is required. On each interface that requires confidentiality, the ESP_NULL transform MUST NOT be used.

#### 7.1.2.2 IPSEC ESP Authentication Algorithms

IPSEC Authentication Algorithm (1 byte) is used by IKE to negotiate a packet authentication algorithm that is used by IPSEC. A list of available IPSEC Authentication Algorithms is specified in RFC 2407. Within IPCablecom, the same Authentication Algorithms are used by all IPSEC key management protocols: IKE, Kerberos and application layer (embedded in IP signalling messages).

For IPCablecom, the following IPSEC Authentication Algorithms are supported (all of which are specified in RFC 2451):

**Table 3: IPSEC Authentication Algorithms**

| Authentication Algorithm | Value | Key Size (in bits) | Support REQUIRED | Description |
|---|---|---|---|---|
| HMAC-MD5 | 1 | 128 | yes (also required by RFC 2407) | MD5 HMAC |
| HMAC-SHA | 2 | 160 | yes | SHA-1 HMAC |

Security Specification for each IPCablecom interface lists whether or not message integrity is required. The Hashed Message Authentication Code Message Digest 5 (HMAC-MD5) and Hashed Message Authentication Code Secure Hash Algorithm (HMAC-SHA) authentication algorithms MUST be supported.

#### 7.1.2.3 Replay Protection

In general, IPSEC provides an optional service of replay protection (anti-replay service). An IPSEC sequence number outside of the current anti-replay window is flagged as a replay and the packet is rejected. When anti-replay service is turned on, an IPSEC sequence number cannot overflow and roll over to 0. Before that happens, a new IPSEC Security Association must be created, as specified in RFC 2406.

Within IPCablecom Security Specification, the IPSEC anti-replay service MUST be turned on at all times. This is regardless of which key management mechanism is used with the particular IPSEC interface.

#### 7.1.2.4 Key Management Requirements

Because within IPCablecom, IPSEC is used on a number of different interfaces with different security and performance requirements, several different key management protocols have been chosen for different IPCablecom interfaces. On some interfaces it is IKE (see clause 7.2), on other interfaces it is Kerberos/PKINIT (see clause 7.3), and in some cases IPSEC keys are distributed over protected signalling interfaces (see clause 8.7.3).

When IKE is not used for key management, an alternative key management protocol needs an interface to the IPSEC layer, in order to create/update/delete IPSEC Security Associations. The IPSEC layer also needs a way to signal a key management application when a new Security Association needs to be set up (e.g. the old one is about to expire or there is not one on a particular interface).

In addition, some network elements are required to run multiple key management protocols. In particular, the CMS and the MTA MUST support multiple key management protocols. The MTA has to support Kerberos/PKINIT on the MTA-CMS signalling interface, in addition to IKE on the CMS-AN and CMS-RKS interfaces. The MTA has to support Kerberos/PKINIT on the MTA-CMS signalling interface, as well get its IPSEC keys for RTCP packets from NCS signalling messages.

The PF_KEY interface (see RFC 2367) SHOULD be used for IPSEC key management within IPCablecom and would satisfy the above listed requirements. For example, PF_KEY permits multiple key management applications to register for rekeying events. When the IPSEC layer detects a missing Security Association, it would signal the event to all registered key management applications. Based on the Identity Extension associated with that Security Association, each key management application would decide if it should handle the event.

# 7.2 Internet Key Exchange (IKE)

## 7.2.1 Overview

IPCablecom utilizes IKE as one of the key management protocols for IPSEC (RFC 2409). It is utilized on the interfaces, where:

- There is not a very large number of connections (in the order of 100 000 or above).
- The endpoints on each connection know about each other in advance.

Within IPCablecom, IKE key management is completely asynchronous to call signalling messages and does not contribute to any delays during communications set-up. The only exception would be some unexpected error, where IPSEC SA is unexpectedly lost by one of the endpoints.

IKE is a peer-to-peer key management protocol. It consists of two phases. In the 1st phase, a shared secret is negotiated via a Diffie-Hellman key exchange. It is then used to authenticate the 2nd IKE phase. The 2nd IKE phase negotiates another secret, used to derive keys for the IPSEC ESP or IPSEC AH protocol.

## 7.2.2 IPCablecom Profile for IKE

### 7.2.2.1 1st IKE Phase

There are multiple modes defined for authentication during the 1st IKE phase.

#### 7.2.2.1.1 IKE Authentication with Signatures

In this mode, both peers are authenticated with X.509 certificates and digital signatures. IPCablecom utilizes this IKE authentication mode on some IPSEC interfaces. Whenever this mode is utilized, both sides MUST exchange X.509 certificates (although this is optional in RFC 2409).

#### 7.2.2.1.2 IKE Authentication with Public Key Encryption

In this mode, both peers are authenticated with encrypted nonces. Each party's ability to reconstruct a hash (proving that the other party decrypted the nonce) authenticates the exchange. In order to perform the public key encryption, the initiator must already have the responder's public key. For this reason, IPCablecom does not utilize this IKE authentication mode.

#### 7.2.2.1.3 IKE Authentication with Pre-Shared Keys

A key derived by some out-of-band (e.g. manual) mechanism is used to authenticate the exchange. IPCablecom utilizes this IKE authentication mode on some IPSEC interfaces. IPCablecom does not specify the out-of-band method for deriving pre-shared keys.

### 7.2.2.2 2nd IKE Phase

In the 2nd IKE phase, an IPSEC ESP SA is established, including the IPSEC ESP keys and ciphersuites. It is also possible to establish multiple IPSEC SAs with a single 2nd phase IKE exchange.

First, a shared 2nd phase secret is established, and then all the IPSEC keying material is derived from it using a one-way function, specified in RFC 2409.

The 2nd phase secret is built from encrypted nonces that are exchanged by the two parties. Another Diffie-Hellman exchange may be used in addition to the encrypted nonces. Within IPCablecom, IKE MUST NOT perform a Diffie-Hellman exchange in the 2nd IKE phase (in order to avoid the associated performance penalties).

The 2nd IKE phase is authenticated using a shared secret that was established in the 1st phase. Supported authentication algorithms are the same as the ones specified for IPSEC in clause 7.1.2.2.

#### 7.2.2.3 Encryption Algorithms for IKE Exchanges

Both phase 1 and 2 IKE exchanges include some symmetrically encrypted messages. The corresponding encryption algorithms supported within IPCablecom are the same as the ones specified for IPSEC in clause 7.1.2.1.

#### 7.2.2.4 Diffie-Hellman Groups

IKE defines specific sets of Diffie-Hellman parameters (i.e. prime and generator) that may be used for the phase 1 IKE exchanges. These are called groups in RFC 2409. The use of Diffie-Hellman groups within IPCablecom is exactly how it is specified in RFC 2409.

## 7.3 Kerberos/PKINIT

### 7.3.1 Overview

The Kerberos protocol with the public key PKINIT extension (draft RFC: "The Kerberos Network Authentication Service (V5)") is utilized for key management on the MTA-CMS interface (see note). Here is a summary of PKINIT and Kerberos:

- This protocol is based on Kerberos tickets, which are cookies, encrypted with the particular server's key. A Kerberos ticket is used to both authenticate a client to a server and to establish a session key, which is contained in the ticket.
- Two-way authentication with public key certificates (with PKINIT) is used by the MTA to obtain a CMS Ticket from the TGS (Ticket Granting Service). (The authentication in each direction is accomplished with a digital signature over some known information (that includes a nonce for replay protection) and with a public key certificate chain. The digital signature is a proof that each party possesses its private key, while the certificate chain authenticates the corresponding public key and associates it with a known identity.)
- The corresponding session key is delivered to the MTA sealed with either the MTA's RSA public key or with a secret derived from a Diffie-Hellman exchange.
- The CMS ticket is kept for a relatively long period of time, days or weeks. The length of this period can be adjusted, based on the performance requirements and based on the clock skew rate at the MTAs. (PKINIT exchange provides a mechanism to synchronize clocks.)
- The CMS ticket contains a symmetric session key, which is in turn used to establish a set of keys for use with the IPSEC ESP mode. The keys used by IPSEC will time out after some configurable time-out period (e.g. 10 minutes). Normally, the same CMS ticket is used to automatically establish a new IPSEC SA. However, there are instances where it is desirable to drop IPSEC sessions after a SA time out and establish them on-demand later. This allows for improved system scalability, where a CMS does not need to maintain a SA for every MTA that it controls. It also is possible that a group (e.g. NCS cluster) of CMSs controls the same subset of MTAs for load balancing. In this case, the MTA does not need to maintain a SA with each CMS in that group. This clause provides Specifications for how to automatically establish a new IPSEC SA right before a time out of the old one and how to establish IPSEC SA on-demand, when a signalling message needs to be sent.
- The IPSEC keys are not derived from the session key itself. Rather, another random key - a subkey - is generated and then used to derive the IPSEC keys.

NOTE: In the present document, we use the term TGS (Ticket Granting Service) for a Kerberos server. The referenced PKINIT and Kerberos specifications often refer to it as the KDC (Key Distribution Centre). A KDC is usually a combination of both the TGS and the Authentication Server. The Authentication Server authenticates the clients and grants them a TGT - Ticket Granting Ticket that can later be used to obtain tickets for specific servers (from the TGS). Since we do not use TGTs within IPCablecom, we do not use the term KDC.

### 7.3.2 PKINIT Exchange

Figure 4 illustrates how the MTA uses PKINIT to obtain a Kerberos ticket for the CMS. The Kerberos ticket is later used by the MTA to set up an IPSEC security association with the CMS, described in the following clause.

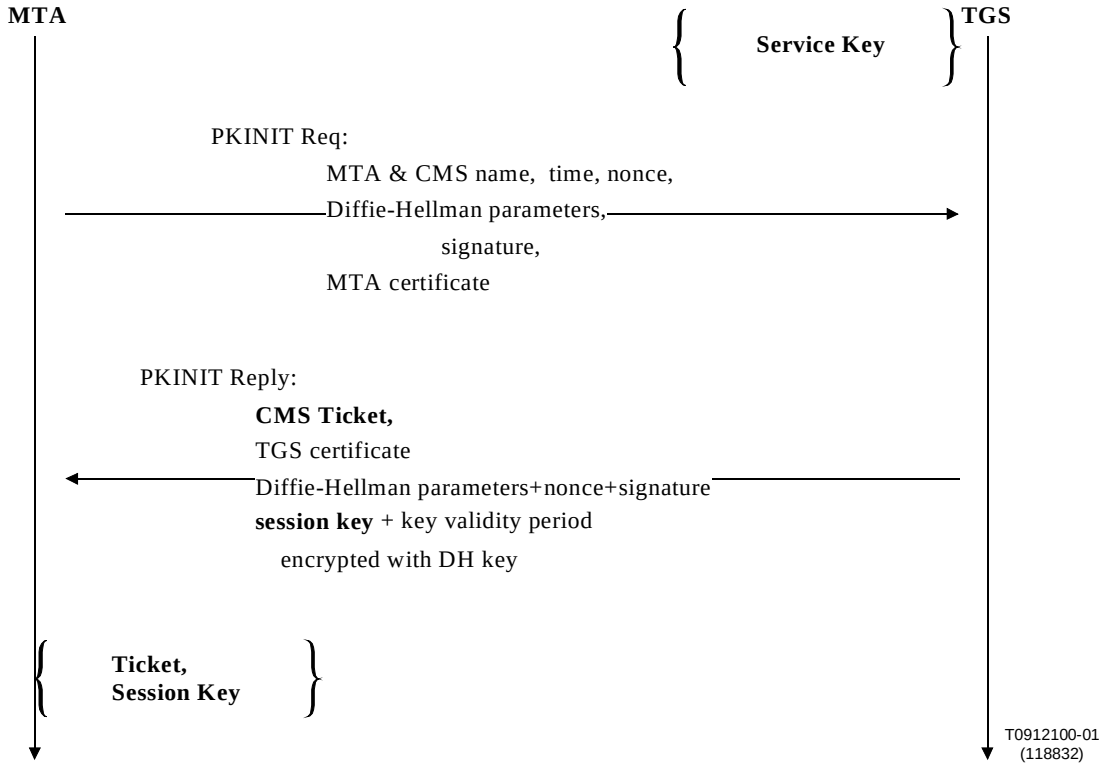



**Figure 4: PKINIT Exchange**

Figure 4 lists several important parameters in the PKINIT Request and Reply messages. These parameters are:

- PKINIT Request:
  - MTA (Kerberos principal) name - found in the KDC-REQ-BODY Kerberos structure. Its format is based on the MTA's X.500 name in the certificate, as specified in draft RFC: "The Kerberos Network Authentication Service (V5)".
  - CMS (Kerberos principal) name - found in the KDC-REQ-BODY Kerberos structure. For the format used in IPCablecom, see clause 7.3.8.
  - time - found in the PKAuthenticator structure.
  - nonce - found in the PKAuthenticator structure. There is also a 2nd nonce in the KDC-REQ-BODY Kerberos structure.
  - Diffie-Hellman parameters, signature and MTA certificate - these are all specified by PKINIT (draft RFC: "The Kerberos Network Authentication Service (V5)") and their use in IPCablecom is specified in clause 7.3.2.1.1.
- PKINIT Reply:
  - CMS Ticket - found in the KDC-REP Kerberos structure.
  - TGS Certificate, Diffie-Hellman parameters, signature - these are all specified by PKINIT and their use in IPCablecom is specified in clause 7.3.2.1.2.
  - nonce - found in the KdcDHKeyInfo structure, specified by PKINIT. This nonce must be the same as the one found in the PKAuthenticator structure of the PKINIT Request. There is another nonce in EncKDCRepPart Kerberos structure. This nonce must be the same as the one found in the KDC-REQ-BODY of the PKINIT Request.
  - session key, key validity period - found in the EncKDCRepPart Kerberos structure.

In this figure, the PKINIT exchange is performed at long intervals, in order to obtain an (intermediate) symmetric session key. This session key is shared between the MTA and the CMS (via the CMS Ticket).

This exchange occurs independent of the signalling protocol, based on the current Ticket Expiration Time $\mathbf{Ticket_{EXP}}$ and on the PKINIT Grace Period $\mathbf{PKINIT_{GP}}$. The MTA initiates the PKINIT exchange at the time: $\mathbf{Ticket_{EXP}}$ - $\mathbf{PKINIT_{GP}}$.

The use of the grace period accounts for a possible clock skew between the MTA and the CMS - if the MTA is late with the PKINIT exchange, it still has until $\mathbf{Ticket_{EXP}}$ before the CMS starts rejecting the ticket.

The PKINIT exchange stops after the MTA obtains a new ticket and therefore does not affect an existing IPSEC Security Association between the MTA and the CMS. We also do not have to worry about synchronizing the PKINIT exchange with the AP Request/Reply exchange, as long as the AP Request/AP Reply exchange is guaranteed a valid, non-expired Kerberos ticket.

The PKINIT Request/Reply messages contain public key certificates, which makes them longer than a normal size of a UDP packet. They will be sent over large UDP packets, requiring fragmentation.

#### 7.3.2.1 PKINIT Profile for IPCablecom

The PKINIT Request is actually carried as a Kerberos pre-authenticator field inside an **AS Request** and the PKINIT Reply is a pre-authenticator inside the **AS Reply**. The syntax of the Kerberos **AS Request/Reply** messages and how pre-authenticators would plug in is specified in draft RFC: "The Kerberos Network Authentication Service (V5)".

Once an MTA receives an AS Reply (with the PKINIT Reply in it), it SHOULD save both the CMS ticket and the session key information (found in the **enc-part** member of the reply) in non-volatile memory (which is usually the case with existing Kerberos implementations). Thus, the MTA will be able to re-use the same Kerberos ticket after a reboot, avoiding the need to perform PKINIT again, with the associated overhead of public key operations.

Since an MTA is not required to save the CMS ticket, the MTAs that do not follow the above Specification should not adversely affect the performance of call signalling. Therefore, a TGS server SHOULD be implemented on a separate host, independent of the CMS. This would mean, that frequent PKINIT operations from some MTAs will not affect the performance of the CMS or the performance of those MTAs that do not require frequent PKINIT exchanges.

Kerberos Tickets MUST NOT be issued for a period of time that is longer than 7 days. The MTA clock MUST NOT drift more than 2,5 minutes within that period (7 days). The PKINIT Grace Period $\mathbf{PKINIT_{GP}}$ MUST be at least 15 minutes.

##### 7.3.2.1.1 PKINIT Request

The PKINIT Request message (PA-PK-AS-REQ) is defined as:

```
PA-PK-AS-REQ ::= SEQUENCE {
signedAuthPack    [0] SignedData
 trustedCertifiers   [1] SEQUENCE OF TrustedCas OPTIONAL,
 kdcCert             [2] IssuerAndSerialNumber OPTIONAL
 encryptionCert      [3] IssuerAndSerialNumber OPTIONAL
 }
```

The following fields MUST be present in PA-PK-AS-REQ for IPCablecom (and all other fields MUST NOT be present):

- **signedAuthPack** - a signed authenticator field, needed to authenticate the client. This structure uses a CMS (Cryptographic Message Syntax) data type SignedData, specified in RFC 2630 and is defined as:

```
SignedData ::= SEQUENCE {
version CMSVersion,
digestAlgorithms DigestAlgorithmIdentifiers,
encapContentInfo EncapsulatedContentInfo,
```

certificates [0] IMPLICIT CertificateSet OPTIONAL,

crls [1] IMPLICIT CertificateRevocationLists OPTIONAL,

signerInfos SignerInfos }

In this structure:

- **digestAlgorithms** - for now MUST contain an algorithm identifier for SHA-1. Other digest algorithms may optionally be supported in the future.
- **encapContentInfo** - is of type **EncapsulatedContentInfo** that is defined by CMS (Cryptographic Message Syntax) as:

EncapsulatedContentInfo ::= SEQUENCE {

eContentType ContentType,

eContent [0] EXPLICIT OCTET STRING OPTIONAL

}

Here **eContentType** indicates the type of data and MUST be set to **id-data** for IPCablecom. **eContent** is a data structure of type **PKAuthenticator** (defined by PKINIT) encoded inside an OCTET STRING. There are no optional parameters in this structure.

- **certificates** - required by IPCablecom. This field MUST contain a single MTA Telephony certificate. If the MTA Telephony certificate was issued by a Local System CA, then the corresponding Local System Certificate MUST also be present. This field MUST NOT contain any other certificates. All IPCablecom certificates are X.509 certificates for RSA Public keys.
- **crls** - MUST NOT be filled in by the MTA.
- **signerInfos** - must be a sequence with exactly one member that holds the MTA signature. This signature is a part of a **SignerInfo** data structure defined within the CMS (Cryptographic Message Syntax). All optional fields in this data structure are not used by IPCablecom and the signature algorithm for now MUST be RSA over a SHA-1 digest.

PKINIT allows an Ephemeral-Ephemeral Diffie-Hellman exchange as part of the PKINIT Request/Reply sequence. (Ephemeral-Ephemeral means that both parties during each exchange randomly generate the Diffie-Hellman private exponents.) The Kerberos session key is returned to the MTA in the PKINIT Reply, encrypted with a secret that is derived from the Diffie-Hellman exchange. Within IPCablecom, the Ephemeral-Ephemeral Diffie-Hellman MUST be supported. IPCablecom requirements for the Diffie-Hellman parameters (i.e. prime and generator) MUST follow the IKE Recommendation in RFC 2409.

Additionally, PKINIT supports a Static-Ephemeral Diffie-Hellman exchange, where the client is required to possess a Diffie-Hellman certificate in addition to an RSA certificate. This mode is not currently used by IPCablecom.

PKINIT also allows a single client RSA key to be used both for digital signatures and for encryption - wrapping the Kerberos session key in the PKINIT Reply. This mode is currently not used by IPCablecom.

PKINIT has an additional option for a client to use two separate RSA keys - one for digital signatures and one for encryption. This mode is not currently used by IPCablecom.

#### 7.3.2.1.2 PKINIT Reply

The PKINIT Reply message (PA-PK-AS-REP) is defined as follows:

PA-PK-AS-REP ::= CHOICE {

dhSignedData [0] SignedData,

encKeyPack [1] EnvelopedData,

}

IPCablecom utilizes only the **dhSignedData** choice, which is needed for a Diffie-Hellman exchange.

The value of the Kerberos session key is not present in PA-PK-AS-REP. It is found in the encrypted portion of the **AS Reply** message that is specified in draft RFC: "The Kerberos Network Authentication Service (V5)". AS Reply MUST be encrypted with 3-DES CBC, where the corresponding Kerberos **etype** value MUST be **3des-cbc-md5**. Other encryption types may be supported in the future.

The client MUST use PA-PK-AS-REP to determine the encryption key used on the **AS Reply**.

**dhSignedData** is of type SignedData, specified in clause 7.3.2.1.1. Within SignedData:

- **digestAlgorithms** - for now MUST contain an algorithm identifier for SHA-1. Other digest algorithms may optionally be supported in the future.
- **encapContentInfo** - is of type **id-data** and contains the following data structure:

```
KdcDHKeyInfo ::= SEQUENCE {
 nonce [0] INTEGER,
 subjectPublicKey [2] BIT STRING
}
```

Where the **nonce** MUST be the same nonce that was passed in by the client in the PKINIT Request and **subjectPublicKey** is the Diffie-Hellman public value generated by the TGS. The Diffie-Hellman-derived key is used to directly encrypt part of the **AS Reply**.

- **certificates** - required by IPCablecom. This field MUST contain a TGS certificate. If the TGS certificate was issued by a Local System CA, then the corresponding Local System Certificate MUST also be present. This field MUST NOT contain any other certificates.
- **crls** - this optional field MAY be filled in by the TGS.
- **signerInfos** - must be a sequence with exactly one member that holds the TGS signature. This signature is a part of a **SignerInfo** data structure defined within the CMS. All optional fields in this data structure MUST NOT be used by IPCablecom and the signature algorithm for now MUST be RSA over a SHA-1 digest.

#### 7.3.2.1.2.1 PKINIT Error Messages

In the case that a PKINIT Request is rejected, instead of a PKINIT Reply the TGS MUST return a Kerberos error message of type the **KRB-ERROR,** as defined in draft RFC: "The Kerberos Network Authentication Service (V5)". Any error code that is defined for PKINIT MAY be returned.

This error message MUST NOT include the optional **e-cksum** member that would contain a keyed checksum of the error reply. The use of this field is not possible during the PKINIT exchange, since the client and the TGS do not share a symmetric key.

##### 7.3.2.1.2.1.1 Clock Skew Error

When the TGS server clock and the client clock are off by more than the limit for a clock skew (usually 5 minutes), an error code KRB_AP_ERR_SKEW MUST be returned along with the difference (in seconds) between the two clocks. The client SHOULD store this clock difference in non-volatile memory and use it to adjust its clock in subsequent PKINIT and AP Request messages.

In the case that a PKINIT request failed due to a clock skew error, an MTA MUST immediately retry after adjusting its clock.

In addition, the MTA MUST validate the time offset returned in the clock skew error, to make sure that it does not exceed a maximum allowable amount. This maximum time offset MUST not exceed 1 hour. This MTA check against a maximum time offset protects against an attack, where a rogue TGS attempts to fool an MTA into accepting an expired TGS certificate.

#### 7.3.2.2 Profile for the Kerberos AS Request/AS Reply Messages

As mentioned earlier, the PKINIT Request and Reply are really pre-authenticator fields, embedded into the AS Request/AS Reply messages.

The client's Kerberos principal name and realm in these messages MUST be formed based on the client's X.500 name and on the client certificate issuer's name respectively.

The optional fields **enc-authorization-data, additional-tickets** and **rtime** in the **KDC-REQ-BODY** MUST NOT be present in the AS Request. All other optional fields in the AS Request MAY be present for IPCablecom. None of the Kerberos ticket flags are currently supported within IPCablecom.

In the AS Reply, **key-expiration**, **starttime**, **renew-till** and **caddr** optional fields MUST NOT be present.

The encrypted part of the AS Reply MUST be encrypted with the encryption type set to **3des-cbc-md5**. The following data MUST be concatenated and in that order before being encrypted with 3-DES CBC, IV=0:

1) 8-byte random byte sequence, called a **confounder**;
2) and MD5 checksum, calculated as specified in draft RFC: "The Kerberos Network Authentication Service (V5)";
3) AS Reply part that is to be encrypted;
4) random padding up to a multiple of 8.

#### 7.3.2.3 Profile for Kerberos Tickets

The following optional fields MUST NOT be present in the tickets: **caddr**, **authorization-data**, **starttime** and **renew-till**. None of the Kerberos ticket flags are currently supported within IPCablecom.

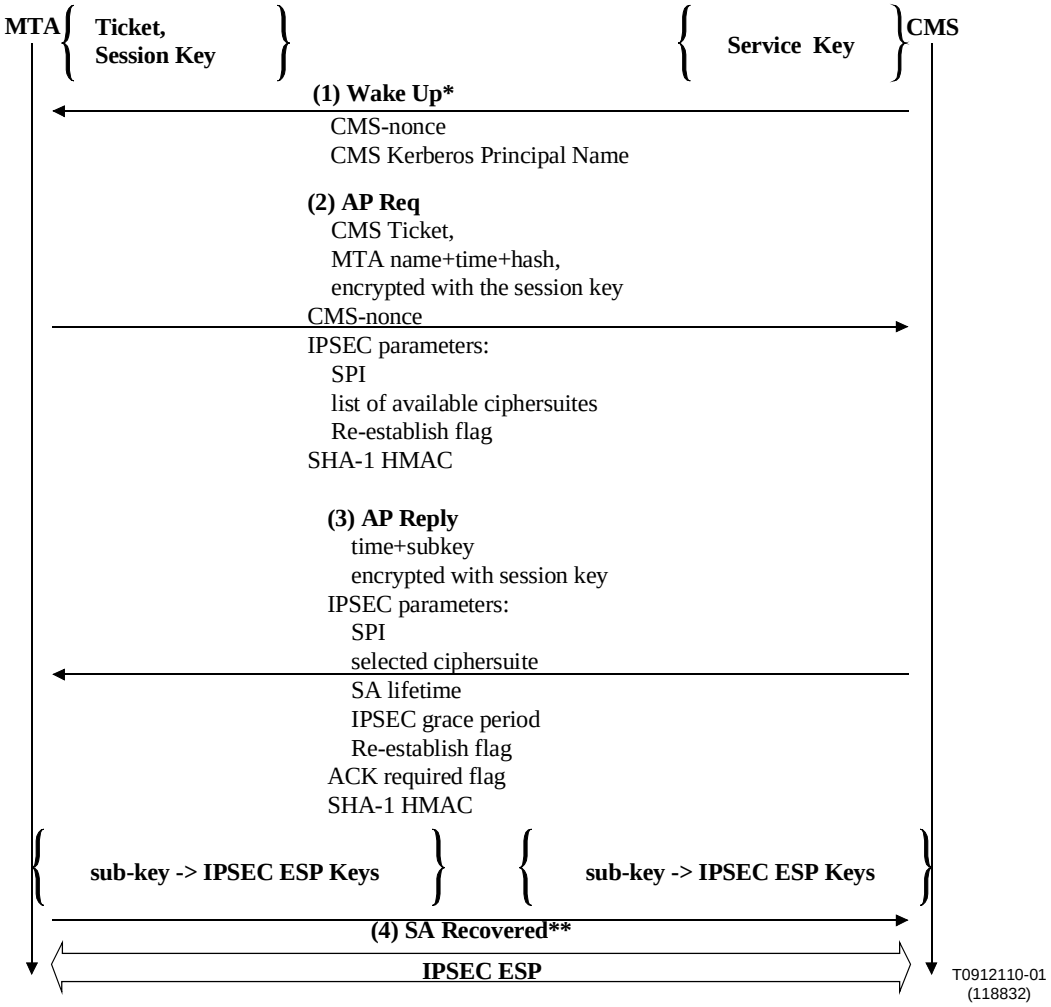
The encrypted part of the Kerberos ticket MUST be encrypted with the encryption type set to **des3-cbc-md5**, using the same procedure as described in the above clause 7.3.2.2.

### 7.3.3 Kerberos AP Request/AP Reply Exchange

The Kerberos session key MUST be used in the AP Request and AP Reply messages that are exchanged in order to re-establish IPSEC keys. This subkey is used to derive all of the IPSEC ESP keys used for both directions. The AP Request and AP Reply messages are small enough to fit into a standard UDP packet, not requiring fragmentation.

A Kerberos AP Request/Reply exchange MAY occur periodically, to insure that there is always a valid IPSEC SA between the MTA and the CMS. It MAY also occur on-demand, where IPSEC SAs are allowed to time out and are re-established the next time that a signalling message needs to be sent.

The following figure illustrates an AP Request/AP Reply exchange.




NOTE 1: This message is sent whenever key management is initiated by the CMS.
NOTE 2: This message is optional, sent whenever the ACK-required flag is set in the preceeding AP Reply.

**Figure 5: Kerberos AP Request/AP Reply Exchange**

1) **Wake Up** - this message is utilized when there is no IPSEC SA between that MTA and the CMS and the CMS decides to re-establish it.

To prevent denial of service attacks, this message includes a CMS-nonce field - a random value generated by the CMS. The MTA MUST include the exact value of this CMS-nonce in the subsequent AP Request.

This message also contains the CMS Kerberos principal name, used by the MTA to find or to obtain a correct Kerberos ticket for that CMS.

The Wake Up message MUST be formatted as the concatenation of the following fields:

- **Key Management Message ID** - 1 byte value. Always set to 0x01.
- **CMS-nonce** - a 4-byte random binary string. Its value MUST NOT be all 0s.
- **CMS Kerberos principal name** - a printable, null-terminated ASCII string, representing a fully qualified Kerberos Principal Name of the CMS, as specified in draft RFC: "The Kerberos Network Authentication Service (V5)".

Once the CMS has sent a Wake Up, it MUST save the CMS-nonce. The CMS MUST keep this nonce for **pktcCmsKeyMgmtTimeout1** in order to validate a matching AP Request. After pktcCmsKeyMgmtTimeout1 the CMS MUST discard this CMS-nonce, after which it will no longer accept a matching AP Request.

2) **AP Request** - MUST be sent by the MTA in order to establish a new IPSEC SA. Any time that the MTA receives a Wake Up message, it MUST respond with this AP Request.

In addition, the present document specifies the use of this message by the MTA to periodically establish a new IPSEC SA with the CMS - see clause 7.3.3.4. It also specifies the use of this message by the MTA to establish a new IPSEC SA with the CMS, when the MTA somehow lost the SA (e.g. after a reboot) - see clause 7.3.5.1.

The MTA starts out with a valid Kerberos ticket, previously obtained during a PKINIT exchange. The CMS starts out with its Service Key that it can use to decrypt and validate Kerberos tickets.

The MTA sends an AP Request that includes a ticket and an authenticator, encrypted with the session key. The CMS gets the session key out of the ticket and uses it to decrypt and then validate the authenticator.

The AP Request includes the Kerberos **KRB_AP_REQ** message along with some additional information, specific to IPCablecom. It MUST consist of the concatenation of the following fields:

- **Key Management Message ID** - 1 byte value. Always set to 0x02.
- **KRB_AP_REQ** - DER encoding of the **KRB_AP_REQ** Kerberos message, as specified in draft RFC: "The Kerberos Network Authentication Service (V5)".
- **CMS-nonce** - a 4-byte random binary string. If this AP Request is in response to a Wake Up, then the value MUST be identical to that of the CMS-nonce field in the Wake Up message. If this AP Request is in response to a Rekey, then the value MUST be identical to that of the CMS-nonce field in the Rekey message (see clause 3.3.3.1). Otherwise, the value MUST be all 0s.
- **SPI** - Security Parameters Index that uniquely identifies a new IPSEC SA for messages sent from the CMS to this MTA. It is a 4-byte integer value, MSB first.
- List of ciphersuites available for IPSEC:
  - number of entries in this list (1 byte);
  - each entry has the following format:

| Authentication Algorithm | Encryption Transform ID |
|---|---|
| (1 byte) | (1 byte) |

For the list of available transforms and their values, refer to clause 7.1 for IPSEC, and to clause 7.3.4 for the bearer channel security.

- **Re-establish flag** - a 1-byte Boolean value. When the value is TRUE (1), the MTA is making an attempt to automatically establish a new IPSEC SA before the old one expires (see clause 7.3.3.4). Otherwise the value is FALSE (0).
- **SHA-1 HMAC** (20 bytes) over the contents of this message, not including this field. The 20-byte key for this HMAC is determined by taking a SHA-1 hash of the session key.

Whenever the AP Request is received (by the CMS), it MUST verify the value of this HMAC. If this integrity check fails, the CMS MUST immediately discard the AP Request and proceed as if the message had never been received (e.g. if the CMS was waiting for a valid AP Request it should continue to do so).

Once the MTA has sent an AP Request, it MUST save the nonce value that was contained in the **seq-number** field (a different nonce from the CMS-nonce specified above) along with the CMS Kerberos principal name. If the MTA generated this AP Request on its own, it MUST keep this nonce and CMS Kerberos principal name for **pktcMtaDevKeyMgmtTimeout1** (a variable in the Media Terminal Adapter Management Information Base (MTA MIB) in order to validate a matching AP Reply. If the AP Request was generated in response to a message sent by the CMS (Wake Up or Rekey), then the MTA MUST keep the nonce and CMS Kerberos principal name for **pktcMtaDevKeyMgmtTimeout2** (a variable in the MTA MIB). After a timeout, the MTA MUST discard this (nonce, CMS Kerberos principal name) pair, after which it will no longer accept a matching AP Reply.

In the case that the CMS-nonce is 0 (not filled in), the CMS MUST verify that this AP Request is not a replay using the procedure specified in the Kerberos standard:

- If the timestamp in the AP Request differs from the current CMS time by more than **pktcCmsToMtaMaxClockSkew** then CMS MUST reply with an error message specified in the clock skew clause.

- If the realm, CMS name, along with the MTA name, time and microsecond fields from the Kerberos Authenticator (in the AP Request) match any recently-seen such tuples, the KRB_AP_ERR_REPEAT error is returned. The CMS MUST remember any authenticator presented within pktcCmsToMtaMaxClockSkew, so that a replay attempt is guaranteed to fail.

- If the CMS loses track of any authenticator presented within pktcCmsToMtaMaxClockSkew, it MUST reject all requests until the clock skew interval has passed.

In the case that the CMS-nonce is not 0, the CMS MAY follow the above procedure in order to fully conform with the Kerberos Recommendation draft RFC: "The Kerberos Network Authentication Service (V5)". In this case, the above procedure is not required because matching the CMS-nonce in the Wake Up or Rekey message against the CMS-nonce in the AP Request also prevents replays.

3) **AP Reply -** Sent by the CMS in response to AP Request.

The AP Reply MUST include a randomly generated subkey, encrypted with the same session key. For IPCablecom, the subkey is the 46-byte **MTA-CMS Secret** - used to derive all the necessary IPSEC keys for both the MTA and the CMS.

The AP Reply includes the Kerberos **KRB_AP_REP** message along with some additional information, specific to IPCablecom. It MUST consist of the concatenation of the following fields:

- **Key Management Message ID** - 1 byte value. Always set to 0x03.

- **KRB_AP_REP** - DER encoding of the **KRB_AP_REP** Kerberos message, as specified in draft RFC: "The Kerberos Network Authentication Service (V5)".

- **SPI** - Security Parameters Index that uniquely identifies a new IPSEC SA for messages sent to the CMS from the MTA. It is a 4-byte integer value, MSB first.

- **selected ciphersuite** for IPSEC, using the same format as defined for AP Request.

- **SA lifetime** - a 4-byte value, MSB first, indicating the number of seconds from now, when this IPSEC SA is due to expire.

- **IPSEC grace period** - a 4-byte value in seconds, MSB first. This indicates to the MTA to start creating a new IPSEC ESP association (with a new AP Request/AP Reply exchange) when the timer gets to within this period of the IPSEC SA expiration time.

- **Re-establish flag** - a 1-byte Boolean value. When the value is TRUE (1), a new IPSEC SA MUST be established before the old one expires as specified in clause 7.3.3.5. When the value is FALSE (0), the old IPSEC SA MUST expire as specified in clause 7.3.3.5.

- **ACK-required flag -** a 1-byte Boolean value. When the value is TRUE (1), the AP Reply message requires an acknowledgement, in the form of the **SA Recovered** message.

- **SHA-1 HMAC** (20 bytes) over the contents of this message, not including this field. The 20-byte key for this HMAC is determined by taking a SHA-1 hash of the session key.

Whenever the AP Reply is received (by the MTA), it MUST verify the value of this HMAC. If this integrity check fails, the MTA MUST immediately discard the AP Reply and proceed as if the message had never been received (e.g. if the MTA was waiting for a valid AP Reply it should continue to do so).

Once the CMS has sent an AP Reply with the ACK-required flag set, it MUST compute the expected value in the SA Recovered message and save it for **pktcCmsKeyMgmtTimeout3** in order to validate an SA Recovered response from the MTA. After pktcCmsKeyMgmtTimeout3 the CMS MUST discard this value, after which it will no longer accept a matching SA Recovered.

4) **SA Recovered** - Sent by the MTA to the CMS to acknowledge that it received an AP Reply and successfully set up new IPSEC SAs. This message is only sent during some error recovery scenarios, when ACK-required flag is set in the AP Reply - see clause 7.3.4.

This message MUST consist of the concatenation of the following:

- **Key Management Message ID** - 1 byte value. Always set to 0x04.

- **HMAC** - a 20-byte SHA-1 HMAC of the preceding AP Reply message. The key used for the hash is the same key that is used to authenticate IPSEC packets on the outgoing IPSEC SA that was just established.

If the receiver (CMS) gets a bad SA Recovered message that does not match an AP Reply, the CMS MUST discard it and proceed as if this SA Recovered message was never received.

#### 7.3.3.1 Rekey Messages

The **Rekey** message replaces the **Wake Up** message and provides better performance, whenever CMS wants to trigger the establishment of a Security Association with a specified MTA. **Rekey** message requires the availability of the shared **Server Authentication Key**, which is not always available. Thus, support for the **Wake Up** message is still required.

The **Rekey** message was added specifically for use with the NCS-based clustered Call Agents, potentially consisting of multiple IP addresses and multiple hosts. Any IP address or host within one cluster needs the ability to quickly establish a new IPSEC Security Association with an MTA, without a significant impact to the ongoing voice communication. For more details, see clause 8.5.1.

The use of the **Rekey** message eliminates the need for the AP Reply message, thus reducing the IPSEC SA establishment delay to a single roundtrip. This is illustrated in figure 6:

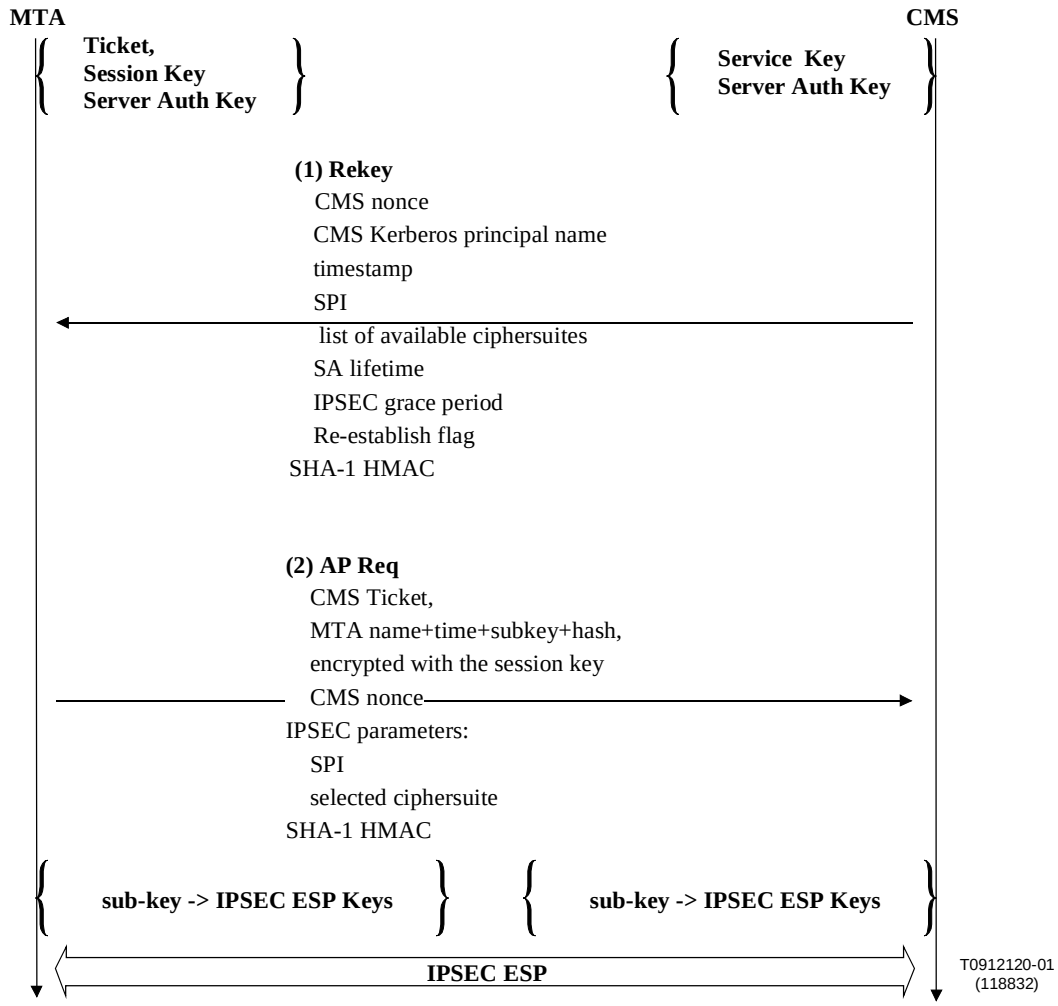



**Figure 6: Rekey Message to establish an IPSEC SA**

The messages listed in this figure are defined as follows:

1) **Rekey** - sent by the CMS to establish a new IPSEC SA. It is a concatenation of the following:

- **Key Management Message ID** - 1 byte value. Always set to 0x05.

- **CMS-nonce** - a 4-byte random binary string. Its value MUST NOT be all 0s.
- **CMS Kerberos Principal Name** - a printable, null-terminated ASCII string, representing a fully qualified Kerberos Principal Name of the CMS, as specified in draft RFC: "The Kerberos Network Authentication Service (V5)". This allows the MTA to both find the right Server Authentication Key and to pick the right Kerberos ticket for the subsequent AP Request message.
- **timestamp** - a string of the format YYMMDDhhmmssZ, representing UTC time. This string is not NULL-terminated.
- **SPI** - Security Parameters Index that uniquely identifies a new IPSEC SA for messages sent to the CMS from the MTA. It is a 4-byte integer value, MSB first.
- **List of ciphersuites** available for IPSEC - see above Recommendation for the **AP Request** message.
- **SA lifetime** - a 4-byte value, MSB first, indicating the number of seconds from now, when this IPSEC SA is due to expire.
- **IPSEC grace period** - a 4-byte value in seconds, MSB first. This indicates to the MTA to start creating a new IPSEC ESP association (with a new AP Request/AP Reply exchange) when the timer gets to within this period of the IPSEC SA expiration time.
- **Re-establish flag** - a 1-byte Boolean value. When the value is TRUE (1), a new IPSEC SA MUST be established before the old one expires, as specified in clause 7.3.3.4. When the value if FALSE (0), the old IPSEC SA MUST expire as specified in clause 7.3.3.5.
- **SHA-1 HMAC** over the concatenation of all of the above listed fields. The **Server Authentication Key** used for this HMAC is uniquely identified by a triple (Kerberos realm, Media Terminal Adapter Fully Qualified Domain Name [MTA FQDN], CMS principal name).
- This key is updated at the CMS right after it sends an AP Reply message. It is set to a (20-byte) SHA-1 HMAC of the Kerberos session key used in that AP Reply. The MTA also updates this key as soon as it receives the AP Reply. (Note that multiple AP Replies will continue using the same Kerberos session key, until it expires. That means that the derived Server Authentication Key may have the same value as the old one.) (see note)

NOTE: It is possible, that the CMS sends a Rekey message as soon as it sends an AP Reply (from another IP address), and before the MTA is able to derive the new Server Authentication Key. In that case, the MTA will not authenticate the **Rekey** message and the CMS will have to retry. Similarly, after sending an AP Reply the CMS might immediately send an IP packet using the just established IPSEC Security Association, when the MTA is not yet ready to receive it. In this case, the MTA will reject the packet and the CMS will have to retransmit. Both of these error cases could be completely avoided with a 3-way handshake (an MTA acknowledging and AP Reply with an SA Recovered message), which is not used in this case for performance reasons - to avoid an extra upstream message.

- Whenever the Rekey message is received (by the MTA), it MUST verify the value of this HMAC. If this integrity check fails, the MTA MUST immediately discard this message and proceed as if the message had never been received.

  Once the CMS has sent a Rekey, it MUST save the CMS-nonce. The CMS MUST keep this nonce for **pktcCmsKeyMgmtTimeout2** in order to validate a matching AP Request. After pktcCmsKeyMgmtTimeout2 the CMS MUST discard this CMS-nonce, after which it will no longer accept a matching AP Request.

  When this Rekey message is received and validated by the MTA, if this MTA previously had any outgoing IPSEC SAs with this CMS IP address, they MUST be removed at this time. If the MTA previously had a timer set for automatic refresh of IPSEC SAs with this CMS IP address, that automatic refresh MUST be disabled.

2) **AP Request** - sent by the MTA as a response to a **Rekey** message. Unlike the AP Request message described above, this one also includes the subkey (inside KRB_AP_REQ ASN.1 structure). KRB_AP_REQ will have a Kerberos flag set, indicating that an AP Reply must not follow.

There are additional, IPSEC-specific fields appended to the AP Request, as specified above in clause 7.3.2.1.2.1. The only difference is that the list of ciphersuites here must contain exactly one entry - the ciphersuite selected by the MTA from the list provided in the **Rekey** message.

Right before this AP Request is sent out by the MTA, it MUST establish both incoming and outgoing IPSEC SAs with the corresponding CMS IP address. If the corresponding Rekey message had the Re-establish flag set, the MTA MUST be prepared to automatically re-establish new IPSEC SAs, as specified in clause 7.3.3.4.

Once this AP Request is received and verified by the CMS, the CMS MUST establish both incoming and outgoing IPSEC SAs with the MTA.

#### 7.3.3.2 IPCablecom Profile for KRB_AP_REQ/KRB_AP_REP Messages

In the KRB_AP_REQ, the following options are supported:

- MUTUAL-REQUIRED - mutual authentication required. When this option is used, the server must respond with an AP Reply message. When this option is not set, the AP Reply must not follow.

All other options are not supported.

When MUTUAL-REQUIRED is set, the encrypted authenticator in the KRB_AP_REQ contains the following fields, which are optional in Kerberos:

- **seq-number**: random value generated by the CMS.

When MUTUAL-REQUIRED is not set, the encrypted authenticator contains the following fields which are optional in Kerberos:

- **subkey** - a 46-byte value of the MTA-CMS Secret.

All other optional fields within the encrypted authenticator are not supported within IPCablecom. The authenticator itself MUST be encrypted using 3-DES CBC with the following possible Kerberos **etype** values: **des3-cbc**. Other encryption types may be supported in the future, as they are added to future versions of Kerberos. Combined etypes that specify an encryption algorithm and a checksum are not used by IPCablecom, since a keyed HMAC (outside of the KRB_AP_REP) is used instead.

In the encrypted part of the KRB_AP_REP, the optional **subkey** field is REQUIRED for IPCablecom. It MUST be the 46-byte **MTA-CMS Secret** - used to derive all the necessary IPSEC keys for both the MTA and the CMS.

The optional **seq-number** MUST be present, and will echo the value that was sent by the client in the KRB_AP_REQ. In this context, the **seq-number** field is used as a random nonce. The encrypted part of the KRB_AP_REP MAY be encrypted with 3-DES CBC, using the following possible Kerberos **etype** values: **des3-cbc**.

##### 7.3.3.2.1 Error Reply

If the CMS is able to successfully parse the AP Request and the ticket that is inside of it, but the AP Request is rejected, it MUST return a Kerberos error message of type **KRB-ERROR,** as defined in draft RFC: "The Kerberos Network Authentication Service (V5)". The error message MUST include the optional **e-cksum** member,

which is the keyed hash over the **KRB-ERROR** message. The checksum type MUST be **rsa-md5-des3**, as it is specified in draft RFC: "The Kerberos Network Authentication Service (V5)". This keyed checksum is calculated by:

1) take an MD5 hash of the KRB-ERROR message;

2) prepend the hash with an 8-byte random byte sequence, called a confounder;

3) take the 3-DES session key from the ticket and XOR each byte with F0;

4) use 3-DES in CBC mode to encrypt the result of step (2), using the key in step (3) and with IV(initialization vector) = 0.

For IPCablecom, the first 4 bytes of the confounder MUST be set to the value of the **seq-number** field from the AP Request.

Upon receiving this error reply, the MTA MUST verify both the keyed checksum and the first four bytes of the confounder, to make sure that it matches the **seq-number** field from the AP Request.

If the CMS is not able to successfully parse the AP Request and the ticket, it MUST drop the request and it MUST NOT return any response to the MTA. In case of a line error, the MTA will time out and re-send its AP Request.

#### 7.3.3.2.2 Clock Skew Error

When the CMS clock and the client clock are off by more than the limit for a clock skew (usually five minutes), an error code KRB_AP_ERR_SKEW MUST be returned along with the difference (in seconds) between the two clocks. The client SHOULD store this clock difference in non-volatile memory and use it to adjust its clock in subsequent PKINIT and AP Request messages.

In the case that an AP Request failed due to a clock skew error, an MTA MUST immediately retry after adjusting its clock.

In addition, the MTA MUST validate the time offset returned in the clock skew error, to make sure that it does not exceed a maximum allowable amount. This maximum time offset MUST not exceed one hour. This MTA check against a maximum time offset protects against an attack, where a rogue TGS attempts to fool an MTA into accepting an expired TGS certificate (later, during the next PKINIT exchange).

### 7.3.3.3 Derivation of MTA-CMS Keys

After the CMS sends out an AP Reply message, it is ready to derive a new set of IPSEC keys. Similarly, after the MTA receives this AP Reply, it is ready to derive the same set of keys for IPSEC. This clause specifies how the IPSEC keys are derived from the MTA-CMS Secret (Kerberos subkey).

The size of the MTA-CMS Secret is 46 bytes (the same as with the SSL or TLS pre-master secret).

The derived IPSEC ESP keys are as follows, in the specified order:

a) message authentication key for MTA->CMS messages;

b) encryption key for MTA->CMS messages;

c) message authentication key for CMS->MTA messages;

d) encryption key for CMS->MTA messages.

For specific authentication and encryption algorithms that may be used by IPCablecom for IPSEC, refer to clause 7.1.

The derivation of the required keying material consists of running a one-way pseudo-random function **F(S, "MTA-CMS Signalling Security Association")** recursively until the right number of bits has been generated. Here, S is the MTA-CMS Secret and the string "MTA-CMS Signalling Security Association" is taken without quotes and without a terminating null character. F is defined in clause 10.6.

### 7.3.3.4 Periodic Re-establishment of IPSEC Security Associations

An IPSEC SA is defined with an expiration time $T_{EXP}$ and a grace period $GP_{IPSEC.}$ The clauses below specify how both the MTA and the CMS handle the re-establishment of IPSEC Security Associations (**renew SA flag** was TRUE in the AP Reply). This must be done in such a way that there is always at least one SA available for each direction and that there is no interruption to call signalling.

#### 7.3.3.4.1 Periodic Re-establishment of IPSEC SAs at the MTA

If the re-establish flag is set, the MTA MUST attempt to establish a new set of IPSEC SAs (one for each direction) starting at the time $T_{EXP}$ - $GP_{IPSEC}$. At this time, the MTA MUST send an AP Request as specified earlier in this clause. After the MTA receives an AP Reply, it MUST perform the following steps:

1) Create new IPSEC SAs, based on the negotiated ciphersuite, SPIs and on the established MTA-CMS Secret, from which the IPSEC keys are derived as specified in clause 7.3.3.2.1. The expiration time for the outgoing SA MUST be set to $T_{EXP}$, while the expiration time for the incoming SA MUST be set to $T_{EXP}$ + $GP_{IPSEC}$.

2) From this point forward, the new SA MUST be used for sending messages to the CMS. The old SA that the MTA used for sending signalling messages to the CMS MAY be explicitly removed at this time, or it MAY be allowed to expire (using an IPSEC timer) at the time $T_{EXP}$.

3) Continue accepting incoming signalling messages from the CMS on both the old and the new incoming SAs, until the time $T_{EXP}$ + $GP_{IPSEC}$. After this time, the old incoming SA MUST expire. If an MTA receives a signalling message from the CMS using a new incoming SA at an earlier time, it MAY at that time remove the old incoming SA.

#### 7.3.3.4.2 Periodic Re-establishment of IPSEC SAs at the CMS

1) When an AP Request message is received and right before an AP Reply is returned, create new IPSEC SAs, based on the negotiated ciphersuite, SPIs and on the established MTA-CMS Secret, from which the IPSEC keys are derived as specified in clause 7.3.3.2.1.

2) Send back an AP Reply.

3) Continue sending signalling messages to the MTA using an old outgoing SA until the time $T_{EXP}$. During the same period, accept incoming messages from either the old or the new incoming SA.

4) At the time $T_{EXP}$ both the old incoming and the old outgoing SAs MUST expire. Switch to the new SA for outgoing signalling messages to the MTA. If for some reason the new SAs were not established successfully, there would not be any IPSEC SAs that are available after this time.

### 7.3.3.5 Expiration of IPSEC SAs

An IPSEC SA is defined with an expiration time $T_{EXP}$ and a grace period $GP_{IPSEC.}$ The clause specifies how both the MTA and the CMS handle the expiration of IPSEC Security Associations (**renew SA flag** was FALSE in the AP Reply).

At the MTA:

- outgoing SA expires at $T_{EXP}$;
- incoming SA expires at $T_{EXP}$ + $GP_{IPSEC}$.

At the CMS:

- outgoing SA expires at $T_{EXP}$;
- incoming SA expires at $T_{EXP}$ + $GP_{IPSEC}$.

After an IPSEC SA had expired and a signalling message needs to be sent by either the MTA or the CMS, an IPSEC layer signals the key management layer to establish a new IPSEC SA. It is established using the same procedures as the ones specified in clause 7.3.5.

## 7.3.4 Initial Establishment of IPSEC SAs

When an MTA is rebooted, it does not have any current IPSEC SAs established with the CMS, since IPSEC SAs are not saved in non-volatile memory. In order to re-establish them, it goes through the recovery procedure that is described in clause 7.3.5.1.

## 7.3.5 Error Recovery

This clause describes the recovery steps that must be taken in the case that an IPSEC Security Association is somehow lost and needs to be re-established.

### 7.3.5.1 MTA Loses an Outgoing IPSEC SA

An MTA attempts to send a signalling message to the CMS. At that time, the IPSEC layer in the MTA realizes the SA is missing and returns an error back to the signalling application (see note). In this case, the following recovery steps MUST be taken at the key management layer:

1) The MTA first makes sure that it has a valid Kerberos ticket for the CMS. If not, it must first perform a PKINIT exchange as specified in clause 7.3.2.

2) MTA sends a new AP Request to the CMS and gets back an AP Reply, as specified in clause 7.3.3.4.

After the receipt of the AP Reply the MTA is prepared to use both of the newly created IPSEC SAs.

The CMS may set an ACK-required flag in the AP Reply. In that case, right after sending out an AP Reply, the CMS is prepared to receive messages on the incoming SA but cannot yet start using an outgoing SA for sending messages to the MTA. In this case, the IPSEC SA setup continues with the following steps 3 and 4.

The CMS may also not set the ACK-required flag in the AP Reply. In that case, right after sending out an AP Reply, the CMS is prepared to both send and receive messages on the newly created SAs. In this case, steps 3 and 4 below are skipped.

Also, after receiving this AP Request (with Re-establish flag = FALSE), the CMS MUST remove any existing outgoing IPSEC SAs that it might already have for this MTA.

3) Immediately after the MTA establishes the new IPSEC SAs, it sends an **SA Recovered** message to the CMS.

4) Upon receipt of this message, the CMS will immediately activate the new outgoing SA for sending signalling messages to the MTA.

NOTE: In this case, there are no actual messages exchanged between the MTA and the CMS.

The signalling application at the MTA MAY either retry a send after some period of time or get an explicit signal from the key management application running on the same MTA, when it completes the establishment of IPSEC SAs.

#### 7.3.5.2 MTA Loses an Incoming IPSEC SA

The MTA receives an IP packet from a CMS on an unrecognized IPSEC SA. This error MUST be ignored by the MTA and the packet MUST be dropped. In this case, any attempt at recovery (e.g. establishing a new IPSEC SA) is prone to denial of service attacks.

#### 7.3.5.3 CMS Loses an Outgoing IPSEC SA

A CMS attempts to send a signalling message to the MTA. At that time, the IPSEC layer in the CMS realizes the SA is missing and returns an error back to the signalling application (see note). In this case, the following recovery steps MUST be taken at the key management layer:

1) CMS sends a **Wake Up** message to the MTA.

2) The MTA makes sure that it has a valid Kerberos ticket for the CMS. If not, it must first perform a PKINIT exchange as specified in clause 7.3.2.

3) MTA sends a new AP Request to the CMS, as specified in clause 7.3.3.4. For each AP Request, the MTA generates a nonce and puts it into the **seq-number** field. As specified in clause 7.3.3.4, the MTA will save this nonce for a short period of time and wait for a matching AP Reply (this is not the same nonce as the CMS-nonce received in the Wake Up). However, after this timeout, the MTA MUST NOT retry and MUST abort an attempt to establish an IPSEC SA in response to a received Wake Up.

   Once the MTA gets back a matching AP Reply, it will be in the format specified in clause 7.3.3.4. The ACK-required flag in the AP Reply is set, to insure that the MTA replies with the SA Recovered message in the following step.

   If this MTA previously had any outgoing IPSEC SAs with this CMS IP address, they MUST be removed at this time. If the MTA previously had a timer set for automatic refresh of IPSEC SAs with this CMS IP address, that automatic refresh MUST be disabled.

   The MTA can start using both of the newly created SAs. If the AP Reply had the Re-establish flag set, the MTA MUST be prepared to automatically re-establish new IPSEC SAs, as specified in clause 7.3.3.4.

   The CMS can receive signalling messages from the MTA on the new incoming SA but cannot yet start using an outgoing SA for sending messages to the MTA.

4) Immediately after the MTA establishes the new IPSEC SAs, it sends an **SA Recovered** message to the CMS.

5) Upon receipt of this message, the CMS will immediately activate the new outgoing SA for sending signalling messages to the MTA.

NOTE: In this case, there are no actual messages exchanged between the MTA and the CMS.

The signalling application at the CMS MAY either retry a send after some period of time or get an explicit signal from the key management application running on the same CMS.

#### 7.3.5.4 CMS Loses an Incoming IPSEC SA

### 7.3.6 The CMS receives an IP packet from a MTA on an unrecognized IPSEC SA

This error must be ignored by the CMS and the packet must be dropped. In this case, any attempt at recovery (e.g. establishing a new IPSEC SA) is prone to denial of service attacks.

**Kerberos Server Locations and Naming Conventions**

#### 7.3.6.1 Kerberos Realms

In IPCablecom, a Kerberos Realm has a one-to-one correspondence with a zone. Each MTA MUST be configured with a single Kerberos realm name. A realm name MAY use the same syntax as a domain name. For a full Recommendation of Kerberos realms, refer to draft RFC: "The Kerberos Network Authentication Service (V5)".

### 7.3.7 TGS

An MTA MUST be configured with DNS names or IP addresses of each of the TGS servers that are available to it. All of the available TGS servers MUST belong to the same Kerberos realm. Regardless of the DNS name for a particular TGS server, its Kerberos principal name is always: **krbtgt@<realm>**, where <realm> is the Kerberos realm corresponding to the particular IPCablecom zone.

This order in which TGS servers are listed in the MTA configuration is the order in which the MTA MUST contact them. The MTA MUST always attempt to contact the first TGS in the list. If it is not available, it MUST try the next one, and so on.

Kerberos implementations usually require that a master (primary) Kerberos server is specified. When clients request a password change or any other updates to the Kerberos database, they must contact the primary Kerberos server. For IPCablecom, the TGS does not perform updates to the Kerberos database. Therefore, for IPCablecom there MUST NOT be a TGS marked as primary or master.

### 7.3.8 CMS

MTA configuration includes a list of CMS DNS names or IP addresses. In addition, each CMS entry MUST include a CMS Kerberos Principal Name, with the realm name not specified (since all CMS entries for one MTA are in the same realm).

Kerberos Recommendation in general allows for principal names to contain an unlimited number of components, of the form <comp1>/<comp2>/<comp3>@<realm>. For IPCablecom, each principal name MUST contain only a single component.

A Kerberos principal name for a CMS might be **CMS1@sandiego.company1.com**. In this example, the realm is **sandiego.company1.com** and it will be a separate MTA configuration parameter. The CMS name will be listed as **CMS1**.

In IPCablecom, a single CMS Kerberos principal name MAY be shared between multiple CMSs that MAY be located on different hosts. In that case, several CMS entries in an MTA configuration file will all have different DNS names or IP addresses but the same Kerberos principal name.

For full syntax of Kerberos principal names, refer to draft RFC: "The Kerberos Network Authentication Service (V5)".

#### 7.3.8.1 Service Key Versioning

The CMS service key that is shared between a TGS and CMS, to encrypt/decrypt CMS tickets, is a versioned key (refer to [17]). This key may be changed either due to a routine key refresh, or because it was compromised. When the CMS service key is changed, the CMS MUST retain the older key for a period of time that is at least as long as the ticket lifetime used when issuing CMS tickets (i.e. up to 7 days).

In the case of a routine service key change, the CMS MUST accept any ticket that is encrypted with an older key that it has retained and is still valid (not compromised). This key versioning on the CMS will prevent against many MTAs from suddenly flooding a TGS with PKINIT Requests for new tickets.

If a CMS service key is changed because it has been compromised, the CMS MUST flag all older key versions it has retained as invalid and reject any AP Request that contains a ticket that is encrypted with one of these invalid keys. When rejecting the AP Request, the CMS MUST respond as specified in [17] with a **KRB_AP_ERR_BADKEYVER** error. The CMS MUST still decrypt the rejected ticket, using the invalid service key, in order to extract the session key. This session key is needed to securely bind the **KRB_ERROR** reply message to the AP Request message using a keyed checksum (see clause 3.3.3.2.1). Note that this step is necessary in order to prevent denial of service attacks which could otherwise occur if the MTA was unable to verify the authenticity of the **KRB_ERROR** message.

Upon receiving this error reply, the MTA MUST discard the CMS ticket which is no longer valid and fetch a new one from its TGS.

The Kerberos Set/Change Password protocol (see [See Kerberos Set/Change Password]) SHOULD be used for setting and changing the CMS service keys. This protocol will provide for greater vendor inter-operability and facilitate key and password administration between the TGS and its principals. Routine CMS service key changes SHOULD be scheduled to occur autonomously using this protocol. Refer to annex D for Specifications on key refresh schedules.

## 7.4 End-to-End Security for RTP

RTP security is currently fully specified in clause 8.7. Key Management for RTP requires that both the (encryption) Transform ID and the Authentication Algorithm are specified, analogous to the IPSEC key management. This clause lists the Transform IDs and Authentication Algorithms that are available for RTP security.

**Table 4: RTP Packet Transform Identifiers (SEC-N-00028v2)**

| Transform ID | Value | Key Size (in bits) | Support REQUIRED | Description |
|---|---|---|---|---|
| Reserved | 0x50 | - | - | |
| RTP_RC4 | 0x51 | 128 | Yes | RC4 stream cipher |
| XDESX-ECB | 0x52 | 192 | No | DESX-XEX-ECB |
| XDESX-CBC | 0x53 | 192 | No | DESX-XEX-CBC |
| DES-CBC-PAD | 0x54 | 128 | No | DES-CBC-PAD |
| 3DES-ECB | 0x55 | 128 | No | 3DES-EDE-ECB |
| 3DES-CBC | 0x56 | 128 | No | 3DES-EDE-CBC |
| Reserved | 0x57-59 | - | - | |

The RTP_RC4 Transform ID MUST be supported.

**Table 5: RTP Packet Authentication Algorithms (SEC-N-00028v2)**

| Authentication Algorithm | Value | Key Size (in bits) | Support REQUIRED | Description |
|---|---|---|---|---|
| AUTH_NULL | 0x60 | 0 | Yes | Authentication turned off |
| Reserved | 0x61 | - | - | |
| RTP_MMH_2 | 0x62 | Variable (see clause 8.7) | Yes | 2-byte MMH MAC |
| Reserved | 0x63 | - | - | |
| RTP_MMH_4 | 0x64 | Variable (see clause 8.7) | Yes | 4-byte MMH MAC |
| Reserved | 0x65 | - | - | |

The Authentication Algorithms AUTH_NULL, RTP_MMH_2 and RTP_MMH_4 MUST be supported.

## 7.5 MAC layer security

All MTAs MUST use CMs compliant with J.112 and its associated security mechanism.

### 7.5.1 The role of MAC layer security within the IPCablecom Security Architecture

The MAC layer security does not provide any security services beyond the J.112 cable access network. The majority of IPCablecom's signalling and media traffic flows, however, take paths that traverse the managed IP "back haul" networks, which lie behind ANs. Since J.112 and IPCablecom service providers typically will not guarantee the security of their managed IP back haul networks, the IPCablecom security architecture defines end-to-end security mechanisms for all these flows. End-to-end security is provided at the Network layer through IPSec, or, in the case of MTA media flows, at the application/transport layer through RTP application layer security. Thus, IPCablecom does not rely on MAC layer security to provide security services to its component protocol interfaces.

## 7.6 Radius

Radius protocol requires an authenticator field for all messages, which provides message integrity. No other security services or key management are defined within the Radius standard RFC 2139.

A 16-byte Authenticator field is calculated as follows:

- **Request Authenticator**: MD5 hash calculated over a stream of octets consisting of the Request Code + Identifier + Length + 16 zero octets + request attributes + shared secret (where + indicates concatenation).
- **Response Authenticator**: MD5 hash calculated over a stream of octets consisting of the Response Code + Identifier + Length + Request Authenticator field from the Accounting-Request packet being replied to + the response attributes if any + shared secret.

The shared secrets for the Response and Request Authenticator fields do not have to be the same.

IPCablecom interfaces that utilize Radius require that the authentication algorithm (ciphersuite) be specified (see clause 7.1.2.2). Currently, only the standard Radius authentication mechanism (as described above) is supported and the ID for this authentication algorithm is **100** (decimal).

## 7.7 DNSSEC

This clause will in the future define a IPCablecom profile for DNSSEC - a specific subset of functionality that is specified in RFC 2137.

# 8 Security Profile

The IPCablecom architecture defines over half a dozen networked components and the protocol interfaces between them. These networked components include the media terminal adapter (MTA), call management server (CMS), signalling gateway (SG), media gateway (MG) and a variety of OSS systems (DHCP, TFTP and DNS servers, network management systems, provisioning servers, etc.). IPCablecom security addresses the security requirements of each constituent protocol interface by:

- Identifying the threat model specific to each constituent protocol interface.
- Identifying the security services (authentication, authorization, confidentiality, integrity, non-repudiation) required to address the identified threats.
- For each constituent protocol interface, specifying the particular security mechanism providing the required security services.

Clause 6.2 describes the threat models applicable to IPCablecom's protocol interfaces. In this clause, we identify the security service requirements of each protocol interface and security mechanisms providing those services.

The security mechanisms include both the security protocol (e.g. IPSec, RTP-layer security, SNMPv3 security) and the supporting key management protocol (e.g. IKE, PKINIT/Kerberos).

The per-protocol security analysis is organized by functional categories (see clause 6.1.3). For each functional category, we identify the constituent protocol interfaces, the security services required by each interface, and the particular security mechanism employed to deliver those security services. Each per-protocol security description includes the detailed information sufficient to ensure interoperability. This includes cryptographic algorithms and cryptographic parameters (e.g. key lengths).

As a convenient reference, each functional category's security analysis includes a summary security profile matrix of the following form (Media security profile matrix shown):

| | **RTP (MTA-MTA, MTA-PSTN GW)** | **RTCP (MTA-MTA, MTA-MG, MG-MG)** |
|---|---|---|
| Authentication | Optional (indirect) | Optional (indirect) |
| Access control | Optional | Optional |
| Integrity | Optional | Yes |
| Confidentiality | Yes | Yes |
| Non-repudiation | No | No |
| Security mechanisms | Application Layer Security via RTP IPCablecom Security Profile.<br>Keys distributed over secured MTA-CMS links.<br>RC4-128 encryption algorithm.<br>Optional 2-byte or 4-byte MAC based on MMH algorithm.<br>IPCablecom requires support for ciphersuite negotiation. | IPSec ESP in transport mode with both encryption and message integrity enabled. The UDP checksum may need to be turned off (set to 0), depending on how NAT is implemented.<br>Keys derived from bearer channel RTP SA. |

Each matrix column corresponds to a particular protocol interface. All but the last row corresponds to a particular security service; the cell contents in these rows indicate whether the a protocol interface requires the corresponding security service. The final row summarizes the security mechanisms selected to provide the required services.

NOTE: The protocol interface column headings not only identify the protocol, but also indicate the network components the protocols run between. Since a CMS can perform multiple functions, the security profile matrices indicate which of the CMS functional components is participating in the identified protocol interface.

## 8.1 Device and Service Provisioning

Device provisioning is the process by which an MTA is configured to support voice communications service. The MTA provisioning process is specified in J.mtadpv.

Figure 7 illustrates the flows involved with the provisioning processes. The provisioning Specification lays these flows out in detail. The flows involving security mechanisms are described in this clause of the document.

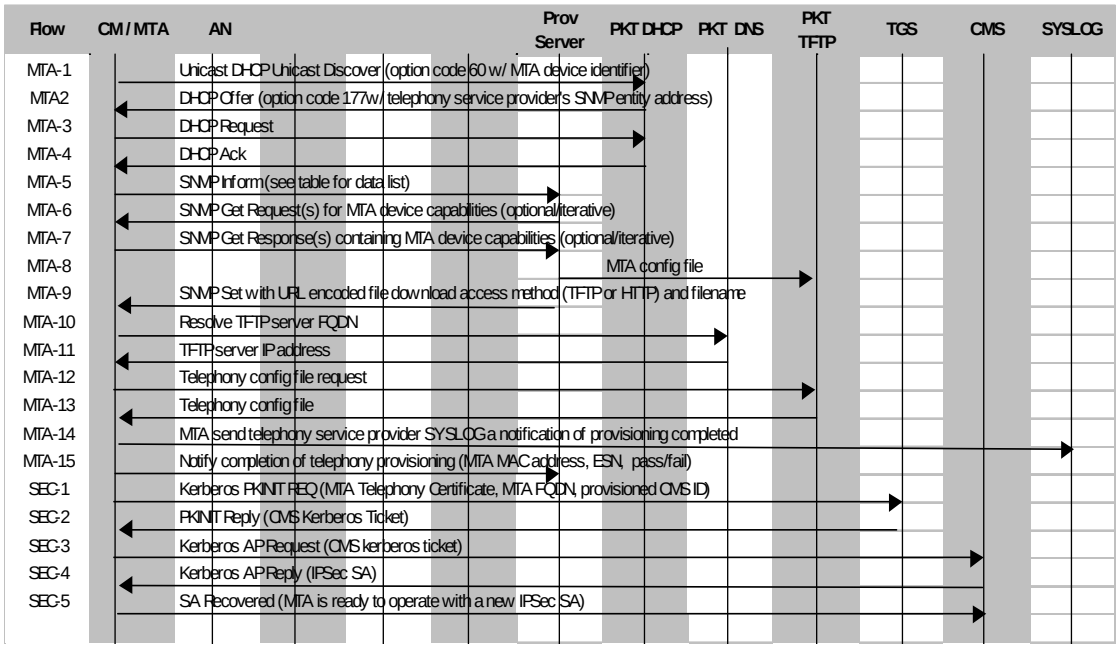



**Figure 7: Illustration of Provisioning Call Flows**

## 8.1.1 Device Provisioning

Device provisioning occurs when an MTA device is inserted into the network. A provisioned MTA device that is not yet associated with a billing record MAY have minimal voice communications service available.

Device provisioning involves the MTA making itself visible to the network, obtaining its IP configuration and downloading its configuration data.

### 8.1.1.1 Security Services

#### 8.1.1.1.1 MTA-DHCP Server

Authentication and Message Integrity is desirable on this interface, in order to prevent denial of service attacks, where an MTA is improperly configured. However, current version of this Specification does not support any security services on this interface.

#### 8.1.1.1.2 MTA-SNMP Manager

This clause applies only to commands that precede the establishment of SNMPv3 security. SNMPv3 is turned on after flow MTA-13.

*Authentication*: the identity of the MTA that is sending configuration parameters to the SNMP manager must be authenticated, to prevent denial of service attacks, where the OSS is tricked into continuously creating bogus configuration files. Or, where the SNMP Manager creates an MTA configuration file based on incorrect MTA capabilities, thus disabling that MTA.

This allows for the possibility of denial of service attacks, where the MTA is told to FTP the wrong configuration file. (The contents of the configuration file will still fail an authentication check - see next clause.) Also, anyone is able to read the MTA configuration parameters - confidentiality is not possible.

*Message Integrity*: is required to prevent denial of service attacks at the OSS and at the MTA - see the above description of the denial of service attacks under authentication.

*Confidentiality*: since confidentiality is not possible on SNMP GET commands prior to distribution of the SNMPv3 keys, the MTA will not respond to queries of sensitive MTA parameters during the device provisioning phase.

*Access Control*: access to the MTA configuration parameters is allowed only for the SNMP Manager that is specified with DHCP. Access Control cannot always be enforced, since DHCP is currently not authenticated.

*Non-Repudiation*: is supported for the identity of the MTA through the use of digital signatures, even though there is no clear need for this security service.

#### 8.1.1.1.3 MTA-Provisioning Server, via TFTP Server

*Authentication*: the identity of the OSS that generated the MTA configuration file is authenticated with a digital signature on that file. This is required to prevent denial of service attacks, where an MTA is improperly configured.

The identity of the MTA requesting the file is not authenticated. Authentication of the MTA is not required, since the configuration file is sealed with the MTA's public key and no one else will be able to use it.

*Message Integrity*: is required to prevent denial of service attacks where an MTA is either improperly configured or configured with old configuration data that was replayed.

*Confidentiality*: is required, because the configuration file contains SNMPv3 secret keys for that MTA.

*Access Control*: not required at the TFTP Server, since each MTA configuration file is encrypted with its public key.

*Non-Repudiation*: is supported for the identity of the OSS through the use of digital signatures, even though there is no clear need for this security service.

### 8.1.1.2 Cryptographic Mechanisms

#### 8.1.1.2.1 Call Flow MTA-5: MTA-SNMP Manager: SNMP INFORM

The SNMP INFORM message contains MTA device attributes and MUST be signed with the private RSA key of the MTA. The signature MUST be performed over the concatenation of the values of the SNMP variables (signature not included), in the order that they appear in the message. The reason this signature is included, is because the SNMPv3 security is not guaranteed until after the TFTP-get is completed and the SNMPv3 keys are established as described in clause 8.3.

The signature itself is included in the SNMP INFORM message as a MIB variable **pktcMtaDevSignature**. The format of the signature uses the ASN.1 DER encoding of **SignedData**, as it is defined by the CMS (Cryptographic Message Syntax) standard in the following table. The SNMP Manager MUST verify the MTA signature before taking any further action based on the SNMP INFORM message. It needs both the MTA Device certificate and the MTA Manufacturer certificate in order to verify this signature. If it does not already have them, it MAY issue subsequent SNMP GETs to query the information from the MTA (further described in the following clause).

The MTA MAC address and MTA Hardware Version reported in the SNMP INFORM message MUST match the MTA MAC address and MTA Hardware Version inside the MTA device certificate (see clause 9.2.4). Responses to any additional queries from the SNMP manager that include the MTA MAC address or the MTA Hardware Version MUST also match the MTA MAC address and MTA Hardware Version inside the MTA device certificate.

PktcDevMtaSignature in the SNMP INFORM includes a timestamp, that is used by the SNMP Manager to detect replays. SNMP messages that are too old MUST be rejected by the SNMP Manager. The SNMP Manager makes this determination based on the maximum allowable clock skew between it and an MTA (a parameter that is locally configured at the SNMP Manager).

#### 8.1.1.2.2 Call Flows MTA-6, 7: MTA-SNMP Mgr: SNMP GET Requests/Responses

Right after the SNMP INFORM, the SNMP Manager MAY follow up with one or more SNMP GET requests to the MTA, in order to query some additional device attributes. Each of these SNMP GETs MUST include the variable **pktcMtaDevSignature**, which will be filled in with a signature, calculated using the same procedure as for the SNMP INFORM. The SNMP Manager MUST verify the MTA signature before taking any further action based on the SNMP GET Reply.

Each of the SNMP GET Replies MUST contain a random nonce value (as part of the signature) that is identical to the one in the SNMP INFORM. The SNMP Manager MUST verify that this nonce value is the same as in the SNMP INFORM, to prevent replays of responses to old SNMP GETs.

As mentioned in the above clause, the SNMP Manager may need to issue SNMP Get commands for the MTA Device and Manufacturer certificates. Due to the large size of the certificates, each such SNMP GET command MUST be issued for just one certificate and with no other SNMP variables requested. This means that as an exception, the PKCS#7 signature in the **pktcMtaDevSignature** variable MUST not be requested. (Certificates already include a signature which MUST be verified by the SNMP Manager.)

Note that the while the SNMP GET Responses are verified, the SNMP GET Requests are not. This means that the SNMP GET Requests received by the MTA at this stage can come from anybody and thus confidentiality of the MTA configuration parameters is not possible. If the MTA contains any sensitive parameters that it does not want publicly disclosed, at this stage it MUST ignore any SNMP GET Requests that attempt to query those sensitive parameters. For a possible list of such parameters, refer to the MTA MIB J.mtamib.

**Table 6: MP signature (pktcMtaDevSignature variable) format using Cryptographic Message Syntax**

| CMS SignedData type Fields | Subfields | Value (Description) |
|---|---|---|
| Version | | **version = 1** (indicates syntax). |
| DigestAlgorithmIdentifiers | | **SHA-1** (collection of message-digest algorithm identifiers identifying the message-digest algorithm under which the content is digested for a signer). |
| EncapsulatedContentInfo | | This data structure contains the content that is signed. The format of this content is specified in the IPCablecom provisioning Specification. |
| | ContentType | data. |
| | Content | file-with-mac-address-name with contents as specified in the Provisioning Specification. |
| Certificates | | Not used (optional in Cryptographic Message Syntax). Because this structure is returned as a value of an SNMP variable, the size limitations do not allow for having certificates here. MTA Device and Manufacturer certificates may be queried separately via SNMP GET commands. |
| CRLs | | (A set of certificate-revocation lists not used in IPCablecom.) |
| SignerInfo | | This is a collection of per-signer information. In this case there is a single signer, the MTA that originates the message. |
| | Version | **version = 1** (syntax version number). |
| | IssuerAndSerialNumber | **MTA Certificate information** (specifies the signer's certificate by issuer distinguished name and issuer-specific serial number). |
| | DigestAlgorithm | **SHA-1** (message-digest algorithm under which the content and authenticated attributes (if present) are digested). |
| | SignedAttributes | **SigningTime** - a standard PKCS#9 attribute that holds the time that the digital signature was generated. See PKCS#9 for the exact format of this attribute.<br>**UniqueIdentifier** - an X.520 attribute, containing a 4-byte integer value, MSB first. It is used as a nonce - a random value that will later be echoed in the file retrieved with TFTP get. It will be used to prevent replays during TFTP get.<br>**ContentType** - data; content type of ContentInfo field.<br>**MessageDigest** - digest of the content plus authenticated attributes. |
| | SignatureAlgorithm | **PKCS#1v1.5 RSA** (Algorithm used to sign the digest and produce the signature). |
| | Signature | The signature. |
| | UnsignedAttrs | Not used (Optional in Cryptographic Message Syntax). |

#### 8.1.1.2.3 Call Flow MTA-8: Provisioning Server-TFTP Server: Create MTA Config File

In this flow, the Provisioning Server builds an MTA device configuration file. This file MUST contain SNMPv3 authentication and privacy keys for the MTA and MUST contain the following configuration info for each endpoint (port) in the MTA:

- CMS name (FQDN format).
- Kerberos Realm for this CMS (see clause 7.3.6.1).
- Kerberos Principal name for this CMS (see clause 7.3.8).
- Telephony Service Provider Certificate (see clause 9.3.2).
- MTA Telephony Certificate, containing the MTA's FQDN (see clause 9.3.4). When the MTA Telephony Certificate is signed by a Local System CA, the corresponding Local System Certificate MUST also be present.
- List of one or more TGS names (FQDN format) for requesting Kerberos tickets for this CMS (see clause 7.3.7).
- PKINIT Grace Period.

This file is signed with the Provisioning Server's private RSA key and sealed with the MTA's public RSA key. The file uses the ASN.1 encoding of **EnvelopedData** (applied to **SignedData**), as it is defined by the Cryptographic Message Syntax standard.

SignedData includes a certificate chain that is needed to verify the Provisioning Server's signature. This certificate chain starts with the Provisioning Server certificate, (optionally) followed by Local System Certificate, followed by the Telephony Service Provider. Telephony Service Provider is signed by the private key of the IP Telephony Root CA. Each MTA MUST be loaded with the public key of the IP Telephony Root CA at manufacture time (which allows the MTA to verify the signature on the Telephony Service Provider). For the Specification of this public key, refer to clause 5.3.1.

The **EnvelopedData** content type consists of encrypted content of any type and content-encryption keys encrypted with the recipient's public key.

Note that the MTA Telephony Certificate MUST accompany the CMS name, regardless of whether or not that MTA is associated with an enrolled subscriber. The MTA Telephony Certificate simply means that the MTA has registered with this network. In order to determine if the MTA is authorized for voice communications service (or is authorized on to communicate to the CSR, 911, etc.), the CMS MUST consult its authorization database that is updated base on the information received from the Back Office.

#### 8.1.1.2.4 Call Flows MTA-9, 10 and 11: Establish TFTP Server Location

This unauthenticated set of call flows is used to establish the IP address of the TFTP server from where the MTA will retrieve its configuration file.

This flow is not authenticated and thus allows for denial of attacks, where the MTA is pointed to a wrong file or TFTP server. The MTA cannot be fooled in accepting the wrong configuration file due to the signature on the file - this denial of service attack will result in failed MTA provisioning.

This threat is not currently addressed, because it is very similar to the denial of service threat, where the MTA is pointed to the wrong SNMP Manager during the DHCP exchanges. DHCP protocol is currently not authenticated within IPCablecom although there are current implementations (based on an Internet Engineering Task Force (IETF) draft) that support it. This threat may be addressed in the future versions of this security Specification.

#### 8.1.1.2.5 Call Flows MTA-12, 13: MTA-TFTP Server: TFTP Get/Get Response

The TFTP get request is not authenticated and thus anyone can request an MTA configuration file. Since this file is encrypted with the MTA's key, no one else can make use of this file.

This flow is open for a denial of service attack, where the TFTP server is made busy with useless TFTP-get requests. This denial of service attack is not addressed at this time.

The TFTP get response retrieves a configuration file from the TFTP server. The configuration file format is described in clause 8.1.1.2.3 above. After step MTA-14 (see figure 7), the SNMPv3 keys contained in the configuration file MUST be used to enable SNMPv3 security (authentication and privacy) on all SNMP messages.

#### 8.1.1.2.6 Notify Completion of Telephony Provisioning - MTA-15

This is an SNMP INFORM message from the MTA to the SNMP Manager that notifies it of the completion of the MTA provisioning process for this MTA. This message is not yet secured with the SNMPv3 keys, since the validation of the MTA configuration file in the previous step might have failed.

This message is protected with a digital signature inside the MIB variable pktcMtaDevSignature. For the format of the value of this variable, refer to table 6. The SNMP Manager MUST validate this signature with the MTA Device Certificate - see clause 8.1.1.2.2.

#### 8.1.1.2.7 Call Flows SEC-1, 2: Get a Kerberos Ticket for the CMS

The MTA uses PKINIT protocol to get a Kerberos Ticket for the specified CMS (see clause 7.3.2). The Kerberos Ticket is issued by the TGS for a group of one or more CMSs, uniquely identified with the pair (Kerberos Realm, CMS Principal Name).

In the event that different MTA ports are configured for a different group of CMSs, the MTA MUST obtain multiple Kerberos Tickets by repeating these call flows for each ticket. It is also possible, that the MTA is configured to request Kerberos Tickets from different TGS servers, depending on the CMS group.

**Table 7: EnvelopedData for TFTP-get file**

| Cryptographic Message Syntax EnvelopedData type Fields | Subfields | Value (Description) |
|---|---|---|
| Version | | **Version = 0** (indicates syntax). |
| OriginatorInfo | | Not used (optional in Cryptographic Message Syntax). |
| RecipientInfo | | (Collection of per-recipient information). There will be only one recipient - the MTA. Each RecipientInfo is of ASN.1 type CHOICE, from which **ktri** of type **KeyTransRecepientInfo** is selected. |
| | Version | **Version = 0** (indicates syntax). |
| | issuerAndSerialNumber | Identifies the certificate used to encrypt the content encryption key. In this case it is the MTA's device certificate. This identifier is the X.500 issuer name and Serial Number of the certificate. |
| | keyEncryptionAlgorithm | **PKCS1v.2 RSA** (algorithm under which the content encryption key is encrypted). |
| | encryptedKey | The content encryption key. |
| EncryptedContentInfo | | This data structure contains the encrypted content. |
| | ContentType | **SignedData** - format is specified in the following table. |
| | EncryptionAlgorithmIdentifier | **3-key 3-DES**. (Content encryption algorithm and the associated parameters.) |
| | EncryptedContent | TFTP-get file contents as specified in the provisioning spec J.mtadpv. |
| UnprotectedAttributes | | Not used (Optional in Cryptographic Message Syntax). |

**Table 8: SignedData embedded inside Enveloped Data for TFTP-get**

| Cryptographic Message Syntax SignedData type Fields | Subfields | Value (Description) |
|---|---|---|
| Version | | **Version = 1** (indicates syntax). |
| DigestAlgorithmIdentifier | | **SHA-1** (collection of message-digest algorithm identifiers identifying the message-digest algorithm under which the content is digested for a signer). |
| EncapsulatedContentInfo | | This data structure contains the signed content. |
| | ContentType | data. |
| | Content | TFTP-get file contents as specified in J.mtadpv. |
| Certificates | | **Provisioning Server Certificate** (used to verify the signature inside SignerInfos).<br>**Local System Certificate** - optional (used to verify the Provisioning Server Certificate).<br>**Telephony Service Provider Certificate** (used to verify either the optional Local System certificate or the Provisioning Server Certificate) (see note). |
| CRLS | | (A set of certificate-revocation lists, not used in IPCablecom.) |
| SIgnerInfos | | This is a collection of per-signer information. In this case there is a single signer, the Provisioning Server. |
| | Version | **Version = 1** (syntax version number). |
| | issuerAndSerialNumber | **Provisioning Server certificate information** (specifies the signer's certificate by issuer distinguished name and issuer-specific serial number). |
| | DigestAlgorithm | **SHA-1** (Message-digest algorithm under which the content and authenticated attributes (if present) are digested.) |
| | SignedAttrs | **UniqueIdentifier** - an X.520 attribute, containing a 4-byte integer value, MSB first. This is the same nonce that was in the preceding TFTP put.<br>**ContentType** - data: content type of the ContentInfo field.<br>**MessageDigest** - digest of the content plus authenticated attributes. |
| | SignatureAlgorithm | **PKCS#1v1.5 RSA** (Algorithm used to sign the digest and produce the signature.) |
| | Signature | The Signature. |
| | UnsignedAttrs | Not used (optional in Cryptographic Message Syntax). |
| NOTE: This certificate MAY be different to the Telephony Service Provider certificate used in authenticating the TGS in the PKINIT Reply. That is, a different Service Provider may operate the provisioning system. In order to activate a particular MTA port, a separate Telephony Service Provider Certificate for telephony signalling MUST be present in the MTA configuration file or set later via SNMP. | | |

#### 8.1.1.2.8 Call Flows SEC-3, 4, 5: Establish IPSEC SAs with the CMS

The MTA uses the Kerberos Ticket to establish a pair of simplex IPSEC Security Associations with the given CMS. In the event that different MTA ports are configured with different CMS (FQDN) names, multiple sets of SAs will be established (one set for each CMS).

Since a Kerberos Ticket is issued for a group of CMSs, it is possible that a single Kerberos Ticket is used to establish more than one set of IPSEC SAs.

In IPCablecom, a CMS FQDN may translate into a list of multiple IP addresses, as would be the case with the NCS clustered Call Agents. In those cases, the MTA MUST initially establish SAs with one of the IP addresses returned by the DNS Server. The MTA MAY also establish SAs with the additional CMS IP addresses.

Additional IPSEC SAs with the other IP addresses may be established later, as needed (e.g. the current CMS IP address does not respond).

#### 8.1.1.2.9 Expiration of MTA Telephony Certificates

When an MTA sends a PKINIT request with an expired certificate, it will get a PKINIT error code KDC_ERR_INVALID_CERTIFICATE from the TGS. In this case, the MTA MUST issue an SNMP INFORM message to the SNMP Manager containing the expired MTA Telephony certificate.

This is a signal for the SNMP Manager to re-issue a new MTA Telephony Certificate and send it to the MTA with the SNMP SET command. (If this subscriber had not paid the bills, the SNMP Manager might purposely refuse the request. In that case, the SNMP SET command with the new certificate would not be sent out.)

Figure 8a illustrates how a new MTA Telephony Certificate is issued.

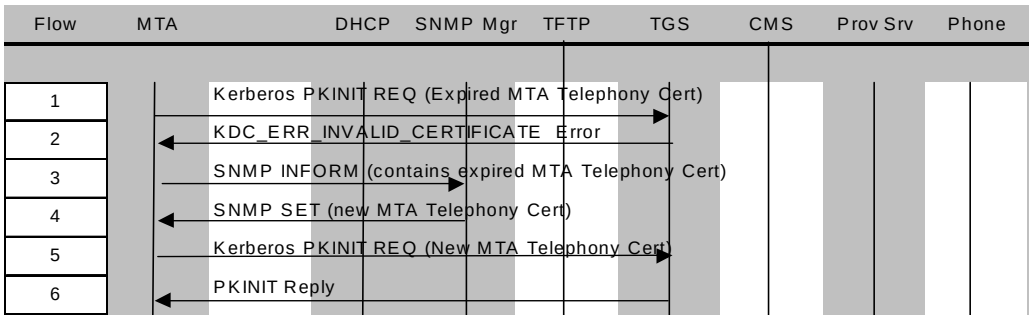



**Figure 8a: Provisioning Server Re-Issues a New MTA Telephony Certificate**

### 8.1.1.3 MTA Embedded Keys

The MTA device will be manufactured with a public/private RSA key pair and an X.509 device certificate that MUST be different from the CM device certificate.

### 8.1.1.4 Summary Security Profile Matrix - Device Provisioning

**Table 9: SMNP & TFTP Security Profile Matrix**

| | SNMP | TFTP (MTA-TFTP server) |
|---|---|---|
| Authentication | Yes for the MTA, no for the SNMP Manager. | Yes: authentication of source of configuration data. |
| Access control | Yes: access limited to a default MIB view until SNMPv3 security is set up. | No. |
| Integrity | Yes for messages initiated by the MTA.<br>No for the SNMP Manager. | Yes:<br>Integrity of configuration data received from authenticated source. |
| Confidentiality | No. | Yes:<br>Of MTA configuration information during the TFTP-get. |
| Non-repudiation | No. | Yes. |
| Security mechanisms | MTA parameters reported with the SNMP INFORM and SNMP Get Response messages are signed by the MTA using Cryptographic Message Syntax format (see note). | All MTA configuration data is signed by the OSS provisioning server and sealed in Cryptographic Message Syntax digital envelope with the MTA's public key. |
| NOTE: Cryptographic Message Syntax. | | |

## 8.1.2 Subscriber Enrolment

The subscriber enrolment process establishes a permanent customer billing account that uniquely identifies the MTA to the CMS via the MTA's MAC address. The billing account is also used to identify the services subscribed to by the customer for the MTA.

Subscriber enrolment MAY occur in-band or out-of-band. The actual Specification of the subscriber enrolment process is out of scope for IPCablecom and may be different for each Service Provider. The device provisioning procedure described in the previous clause allows the MTA to establish IPSEC Security Associations with one or more Call Agents, regardless of whether or not the corresponding subscriber had been enrolled.

As a result, when subscriber enrolment is performed in-band, a communication to a CSR (or to an automated subscriber enrolment system) is protected using the same security mechanisms that are used to secure all other voice communication.

During each communication setup (protected with IPSEC ESP), the CMS MUST check the identity of an MTA against its authorization database to see which voice communications services are permitted. If that MTA does not yet correspond to an enrolled subscriber, it will be restricted to permitting a customer to contact the service provider to establish service ("customer enrolment"). Some additional services, such as communications with emergency response organizations (e.g. 911, 112), may also be permitted in this case.

Since in-band customer enrolment is based on standard security provided for call signalling and media streams, no further details are provided in this clause. Refer to clause 8.5 and to clause 8.7 on media streams.

# 8.2 Quality of Service (QoS) Signalling

## 8.2.1 Dynamic Quality of Service (DQoS)

### 8.2.1.1 Reference architecture for embedded MTAs

Figure 8b shows DQoS connections for embedded MTAs, using an option that does not require RSVP (see note). This version of DQoS does not work for stand-alone MTAs.

NOTE: RSVP is out of scope for IPCablecom.

Figure 8b is also NCS-specific. There are additional DQoS interfaces required for DCS that are not shown on that figure, as they are out of scope for IPCablecom.

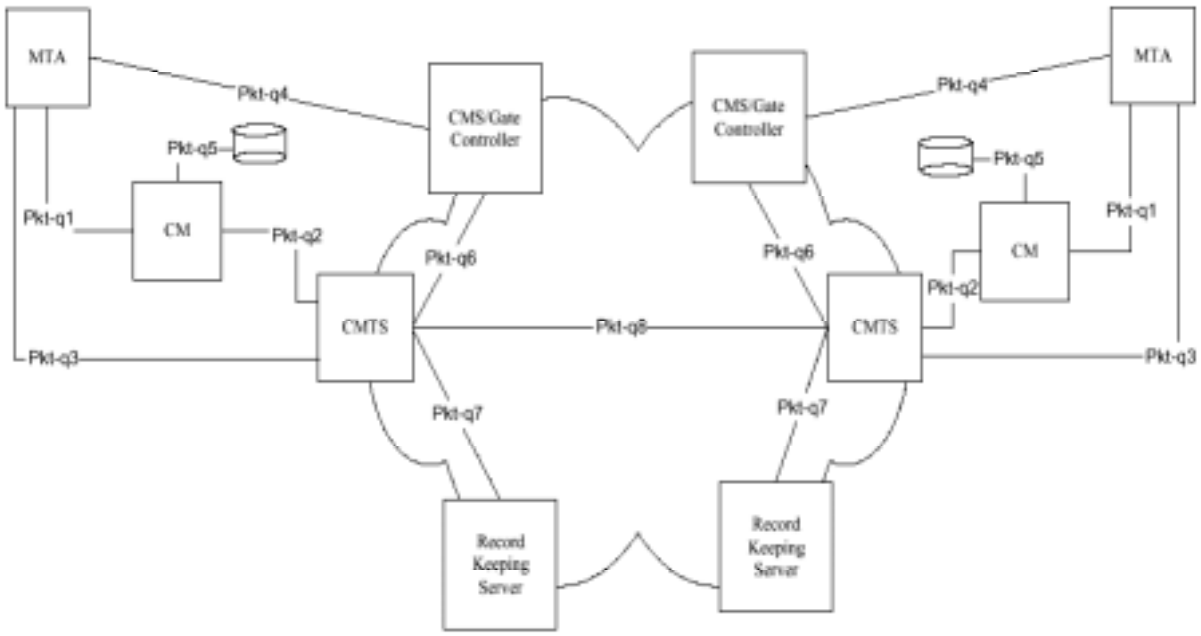



**Figure 8b: QoS Signalling Interfaces in IPCableCom Network**

#### 8.2.1.2 Security Services

##### 8.2.1.2.1 CM-AN J.112 QoS Messages

Refer to J.112.

##### 8.2.1.2.2 AN-CMS Gate Coordination Messages (over UDP)

The AN-CMS Gate Coordination messages are required in order to prevent some theft of service scenarios, described in TS 101 909-5.

*Authentication*: required to prevent theft of service and denial of service attacks. Direct authentication is not possible due to the large number of associations and performance requirements (for post-dial and post-pickup delays). Each end is indirectly authenticated, via a trusted third party - the CMS.

*Message Integrity*: required on this interface. Without it, the same service theft scenarios are still possible, along with the denial of service attacks.

*Per-Communication Security Associations*: although for IPCablecom, these messages are always between the local AN and CMS, this will not be the case in future IPCablecom versions, when DCS is introduced. In the future, the same messages may be exchanged between two ANs, or between a AN and a remote CMS or MGC. Thus, a potentially large and dynamic number of security associations is anticipated for the Gate Coordination messages, requiring that they are established on a per-communication basis.

##### 8.2.1.2.3 Gate Controller - AN COPS Messages

*Authentication, Access Control and Message Integrity*: required to prevent QoS theft and denial of service attacks.

*Confidentiality*: required to keep customer information private.

#### 8.2.1.3 Cryptographic Mechanisms

##### 8.2.1.3.1 CM-AN J.112 QoS Messages

The J.112 QoS messages are specified in the annex A of J.112.

###### 8.2.1.3.1.1 QoS for the J.112 Flow

A J.112 flow is a J.112 MAC-layer transport service that provides transport of packets between the CM and the AN. A J.112 Flow is characterized by a set of QoS Parameters such as latency, jitter, and throughput assurances. In order to standardize operation between the CM and AN, these attributes include details of how the CM requests slots and the expected behaviour of the AN upstream scheduler.

Applied to each packet entering the cable network is a set of matching criteria called a Classifier. A classifier consists of some packet matching criteria (IP source address, for example), a classifier priority, and a reference to a J.112 Flow. If a packet matches the specified packet matching criteria, it is then delivered on the referenced J.112 Flow.

Downstream Classifiers are applied by the AN to packets it is transmitting, and Upstream Classifiers are applied at the CM and may be applied at the AN to police the classification of upstream packets.

The network can be vulnerable to IP packet attacks; i.e. attacks stemming from an attacker using another MTA's IP source address and flooding the network with the packets intended for another MTA's destination address. An AN controlling downstream J.112 Flows will limit an MTA's downstream bandwidth according to QoS allocations. If the AN is flooded from the backbone network with extra packets intended for one of its MTAs, packets for that MTA may be dropped to limit the downstream packet rate to its QoS allocation. The influx of the attacker's packets may result in the dropping of good packets intended for the destination MTA.

To thwart this type of network attack, access to the backbone network should be controlled at the entry point. This can be accomplished using a variety of QoS classifiers, but is most effective when the packet source is verified by its source IP address. This will limit the ability of a rogue source from flooding the network with unauthorized IP packets.

To address this situation where an AN accesses the network, the AN SHOULD apply upstream classifiers to police upstream packets from its network; including the verification of the source IP address.

For more information regarding the use of packet classifiers, refer to J.112.

#### 8.2.1.3.2 AN-CMS Gate Coordination Messages (over UDP)

These are DQoS handshake messages between the two sides of the communication, to make sure that the QoS resources had been reserved on both sides. These messages MUST be formatted as Radius messages, which include an authenticator field. Security for this interface is provided solely by the Radius-specific authenticator, based on an MD5 hash, as defined by RFC 2139. This provides message integrity, but not privacy. The key for the Radius authenticator MUST be exactly 16 bytes long, and there will be a separate key used for each direction.

#### 8.2.1.3.3 Gate Controller - AN COPS Messages

The Gate Controller function in the CMS MUST send COPS messages to the AN, to download a QoS policy for a particular communications connection. These messages MUST be both authenticated and encrypted with IPSEC ESP. Refer to clause 7.1 on the details of how IPSEC ESP is used within IPCablecom and for the list of available ciphersuites.

### 8.2.1.4 Key Management

#### 8.2.1.4.1 AN-CMS Gate Coordination Messages (over UDP)

The keys for this interface are securely distributed by the local CMS over the existing IPSEC links. The key is included in the Gate Authorization message Common Open Policy Service Protocol (COPS) sent from the local CMS to the AN. For the on-net to on-net communications, the keys for Gate Coordination messages on each side of the communication are distinct (which will not be the case for DCS).

There is only one message that is involved in the key management for this interface: the Gate-Set Authorization message sent to the AN by the local CMS which MUST include the following parameters:

- Transaction ID (associates request an response together).
- Gate ID.
- Authentication algorithm (1 byte) - only Radius, MD5-based MAC is currently supported. See clause 7.6.
- Radius Key (16 bytes).
- IP address of the local CMS (listed as the IP address of the remote gate in the Gate-Set messages).

For the gate coordination exchange to proceed, the AN must accept this Radius Key and ciphersuite and use it to authenticate Gate Coordination messages in both directions for the specified Gate ID.

Per-communication keying provides replay protection. The same Radius key is used to authenticate only a few messages, consisting of the Gate-Open/Gate-Open-Ack and the Gate-Close/Gate-Close-Ack exchanges. Each message type is clearly identified and appears exactly once for each communication.

#### 8.2.1.4.2 Gate Controller - AN COPS Messages

Key management for this COPS interface MUST be implemented via IKE. IKE must used pre-shared keys. For more information on the IPCablecom use of IKE, refer to clause 7.2.2.

#### 8.2.1.5 Summary Security Profile Matrix

**Table 10: COPS & Gate Coordination Security Profile Matrix**

| | COPS (AN-CMS) | Gate Coordination (AN-CMS) |
|---|---|---|
| Authentication | Yes | Yes (through a third party) |
| Access control | Yes | No |
| Integrity | Yes | Yes |
| Confidentiality | Yes | No |
| Non-repudiation | No | No |
| Security Mechanisms | IPSec with encryption and message integrity.<br>IKE w/ pre-shared keys. | Radius authentication (MD5-based MAC).<br>CMS distributes key per communication. |

## 8.3 OSS Interfaces

This clause describes the use of SNMPv3 for querying the status of the MTAs and other network devices, as well as for their dynamic configuration. Standard SNMPv3 security is used.

Additionally, during the MTA device provisioning, SNMPv3 keys are not yet established and authentication is accomplished with a digital signature in one of the SNMP variables. For details of the use of SNMPv3 during device provisioning, refer to clause 8.1.1.

### 8.3.1 Security Services

*Authentication*: required for both the MTA and the SNMP Manager, to prevent denial of service attacks and disclosure of sensitive MTA parameters.

*Access Control*: required, to prevent unauthorized SNMP Manager from reading or updating MTA parameters.

*Message integrity*: required on this interface. It is needed to prevent denial of service attacks, where an MTA is misconfigured or the SNMP Manager is given wrong MTA status or configuration information.

*Confidentiality*: may be required for some sensitive MTA parameters. For a list of such MTA parameters, refer to the MTA MIB in J.mtamib.

### 8.3.2 Cryptographic Mechanisms

SNMPv3 supports HMAC MD5 and HMAC SHA-1 algorithms for authentication. As specified in RFC 2574, HMAC MD5 MUST be supported. In addition, HMAC SHA-1 authentication SHOULD also be supported.

If an SNMP command contains a value of a sensitive SNMP parameter, it MUST be encrypted. The encryption algorithm is DES-CBC, as specified in RFC 2574. However, SNMP implementations are encouraged to support stronger encryption algorithms in addition to DES, such as 3-DES CBC (see note).

NOTE: It is desirable to track the progress of the SNMPv3 standard. It appears that there will soon be a new RFC that provides stronger (than DES) encryption with SNMPv3. At that time, this part should be updated to require a new, stronger encryption with SNMPv3.

### 8.3.3 Key Management

SNMPv3 security Recommendation in RFC 2574 defines key change messages which are protected with the old value of the key. These key change messages MUST be supported as part of SNMPv3.

However, if an SNMPv3 authentication or privacy key is known to be compromised, this key change mechanism SHOULD NOT be used. During device provisioning (clause 8.1.1), the MTA receives its authentication and privacy keys in an encrypted configuration file. This is the method that SHOULD be used when an SNMPv3 is known or suspected to be compromised.

When the MTA receives its authentication and privacy keys in the configuration file, it still needs to localize them with the SNMP Engine ID. The procedure for SNMP key localization is specified in RFC 2574.

### 8.3.4 Summary Security Profile Matrix

**Table 11: SNMP Security Profile Matrix**

| | SNMP (SNMP Mgr-MTA) | SNMP (SNMP Mgr-network devices) |
|---|---|---|
| Authentication | Yes | Yes |
| Access control | Yes | Yes |
| Integrity | Yes | Yes |
| Confidentiality | Yes | Yes |
| Non-repudiation | No | No |
| Security mechanisms | SNMPv3 security.<br>USM for SNMPv3.<br>IPCablecom VACM profile.<br>Obtain auth and priv keys in CMS sealed object (config file). | SNMPv3 security.<br>USM for SNMPv3.<br>VACM for SNMPv3.<br>Pre-shared keys. |

## 8.4 Billing System Interfaces

### 8.4.1 Security Services

#### 8.4.1.1 CMS-RKS Interface

*Authentication, Access Control and Message Integrity*: required to prevent service theft and denial of service attacks. Want to insure that the billing events reported to the RKS are not falsified.

*Confidentiality*: required to protect subscriber information and communication patterns.

#### 8.4.1.2 AN-RKS Interface

*Authentication, Access Control and Message Integrity*: required to prevent service theft and denial of service attacks. Want to insure that the billing events reported to the RKS are not falsified.

*Confidentiality*: required to protect subscriber information and communication patterns. Also, effective QoS information and network performance is kept secret from competitors.

### 8.4.2 Cryptographic Mechanisms

Both message integrity and privacy MUST be provided by IPSEC ESP, using any of the ciphersuites that are listed in clause 7.1.

RADIUS itself defines MD5-based keyed MAC for message integrity at the application layer. And, there does not appear to be a way to turn off this additional integrity check at the application layer. For IPCablecom, the key for this RADIUS MAC MUST always be hardcoded to the value of 10 ASCII 0s. That is, the shared secret is "0000000000000000". This in effect turns the RADIUS keyed MAC into an MD5 hash that can be used to protect against transmission errors but does not provide message integrity. No key management is needed for RADIUS MACs.

Billing event messages contain an 8-octet binary **Element ID** of the CMS or the AN. The RKS MUST verify for each billing event that the specified Element ID correctly corresponds to the IP address. This check is done via a lookup into a map of IP addresses to Element IDs. Refer to clause 8.4.3 on how this map is maintained.

#### 8.4.2.1 RADIUS Server Chaining

RADIUS servers may be chained. This means that when the local RADIUS server that is directly talking to the CMS or AN client is not able to process a message, it forwards it to the next server in the chain.

IPCablecom specifies security mechanisms only on the links to the local RADIUS server. IPCablecom also requires authentication, access control, message integrity and privacy on the interfaces between the chained RADIUS servers, but the corresponding Specifications are outside of the scope of IPCablecom 1.0.

Key Management (in the following clause) applies to the local RADIUS Server/RKS only.

## 8.4.3 Key Management

### 8.4.3.1 CMS-RKS Interface

CMS and RKS will negotiate a shared secret (CMS-RKS Secret) using IKE. IKE may use one of the modes with pre-shared keys. Certificates may be used in future versions of IPCablecom. For details, refer to clause 7.1.2.3.

IKE will be running asynchronous to the billing event generation and will guarantee that there is always a valid, non-expired CMS-RKS Secret. This shared secret MUST be unique to this particular CMS and RKS.

At the RKS, CMS Element IDs MUST somehow be associated with the corresponding IP addresses. One possibility is to associate each pre-shared key directly with the Element ID. IKE negotiations will use an ISAKMP identity payload of type ID_KEY_ID to identify the pre-shared key. The value in that identity payload will be the Element ID used in billing event messages. For more details refer to RFC 2407.

Later, when a billing event arrives at the RKS, it will be able to query the database of IPSEC SAs and retrieve a source IP address, based on the Element ID. The RKS will make sure that it is the same as the source IP address in the IP packet header. One way to query this database is through SNMP, using an IPSEC Monitoring MIB.

### 8.4.3.2 AN-RKS Interface

AN and RKS will negotiate a shared secret (AN-RKS Secret) using IKE. IKE may use one of the modes with pre-shared keys. For details, refer to clause 7.1.2.3.

IKE will be running asynchronous to the billing event generation and will guarantee that there is always a valid, non-expired AN-RKS Secret. This shared secret SHOULD be unique to this particular AN and RKS.

At the RKS, AN Element IDs MUST somehow be associated with the corresponding IP addresses. One possibility is to associate each pre-shared key directly with the Element ID. IKE negotiations will use an ISAKMP identity payload of type ID_KEY_ID to identify the pre-shared key. The value in that identity payload will be the Element ID used in billing event messages. For more details refer to RFC 2407.

Later, when a billing event arrives at the RKS, it will be able to query the database of IPSEC SAs and retrieve a source IP address, based on the Element ID. The RKS will make sure that it is the same as the source IP address in the IP packet header. One way to query this database is through SNMP, using an IPSEC Monitoring MIB.

## 8.4.4 Summary Security Profile Matrix

**Table 12: Radius Security Profile Matrix**

| | **RADIUS Accounting (CMS-Radius Server/RKS)** | **RADIUS Accounting (AN-Radius Server/RKS)** |
|---|---|---|
| Authentication | Yes | Yes |
| Access control | Yes | Yes |
| Integrity | Yes | Yes |
| Confidentiality | Yes | Yes |
| Non-repudiation | No | No |
| Security mechanisms. | IPSEC ESP with encryption and message integrity enabled.<br>Key management using IKE with pre-shared keys or certificates. | IPSEC ESP with encryption and message integrity enabled.<br>Key management using IKE with pre-shared keys or certificates. |

# 8.5 Call Signalling

## 8.5.1 Network Call Signalling (NCSv2)

### 8.5.1.1 Reference Architecture

Figure 9 shows the network components and the various interfaces to be discussed in this clause.

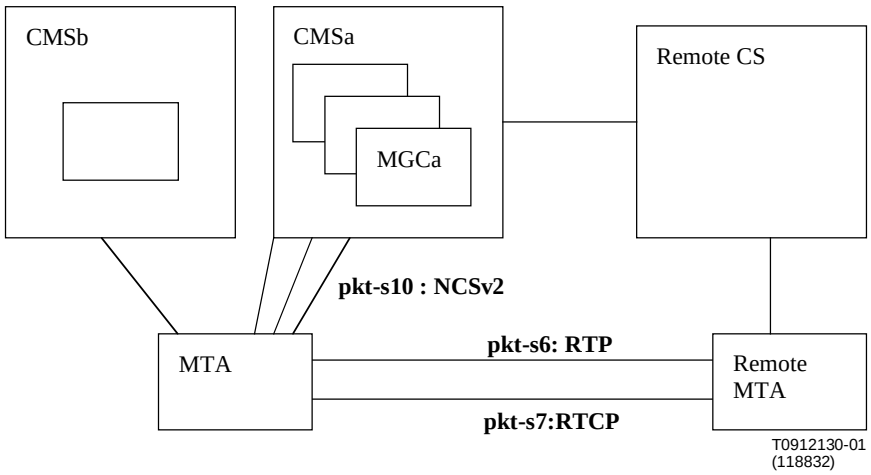



**Figure 9: NCSv2 Reference Architecture**

The figure shows the CMSs as being a cluster of several MGCs. It also shows, even though this is not a likely scenario in early deployments, that different CMSs could potentially manage different endpoints in a single MTA.

The security aspects of interfaces pkt-s6 and pkt-s7 (RTP bearer channel and RTCP) are described in clause 8.7 of the present document. The protocol interface pkt-s16 (CMS to CMS) has not been defined for IPCablecom and thus it is outside the scope of the present document.

The rest of this clause is concerned with interface pkt-s10.

### 8.5.1.2 Security Services

The same set of requirements applies to both CMS-MTA and CMS-CMS signalling interfaces. CMS-CMS signalling interface is not specified in the current version of NCSv2, so the implementation of the CMS-CMS security requirements is out of scope for IPCablecom.

*Authentication*: signalling messages must be authenticated, in order to prevent a third party masquerading as either an authorized MTA or CMS.

*Confidentiality*: NCS messages carry dialled numbers and other customer information, which must not be disclosed to a third party. Thus confidentiality of signalling messages is required.

*Message integrity*: must be assured in order to prevent tampering with signalling messages - e.g. changing the dialled phone numbers.

*Access control*: Services enabled by the NCS signalling should be made available only to authorized users - thus access control is required at the CMS.

### 8.5.1.3 Cryptographic Mechanisms

IPSEC ESP MUST be used to secure this interface. IPSEC keys MUST be derived using mechanism described in clause 7.3.3.3.

Each signalling message coming from the MTA and containing the MTA domain name (included in the NCSv2 **endpoint ID** field) must be authenticated by the CMS. This domain name is an application-level NCSv2 identifier that will be used by the Call Agent to associate the communication with a paying subscriber.

In order to perform this authentication, the CMS MUST maintain an IP address <-> domain name map for each MTA IP address that has a current SA. This map is built during key management described in the following clause and does not need to reside in permanent storage.

#### 8.5.1.4 Key Management

Kerberos with PKINIT MUST be used as the key distribution mechanism on the pkt-s10 interface. It has been described in detail in clause 7.3 of the present document. That clause also describes the mechanism to be deployed to handle timed-out IPSEC keys and Kerberos tickets. The mechanism for transparently handling key switchover from one key lifetime to another key lifetime is also defined.

The key distribution and timeout mechanism is not linked to any specific NCSv2 message. Rather, the MTA will obtain the Kerberos ticket from the TGS when started and will refresh it based on the timeout parameter. The MTA will also obtain sub-key (and thus IPSEC ESP keys) based on timeout parameters. The MTA will also obtain the IPSEC ESP keys when they are timed out and the MTA needs to transmit data to the CMS.

The IPSec Error Recovery Section also describes the technique to be deployed by the CMS in the case the IPSEC keys are timed out and the CMS needs to send data to the MTA.

For NCSv2, at the time that the CMS receives a Kerberos ticket for the purpose of establishing an IPSEC SA, it MUST extract the MTA domain name from the ticket and map it to the IP address. This map is later used to authenticate the MTA endpoint ID in the NCSv2 signalling messages.

The MTA Kerberos principal name in the ticket is a representation of the subject (X.500) name from the MTA Telephony Certificate. Within that MTA Telephony Certificate, the value of the CN (CommonName) attribute will contain the DNS name that needs to be saved.

##### 8.5.1.4.1 Call Agent Clustering

In the case a CMS is constructed as a cluster of Call Agents with different IP addresses, all Call Agents should share the same service key for decrypting a Kerberos ticket. Thus the MTA will need to execute single PKINIT Request/Reply sequence with the TGS and multiple AP Request/Reply sequence for each Call Agent in the cluster. The Kerberos messages are specified in clause 7.3.

Optimized key management is specified for the case when in the middle of a communication, a clustered Call Agent sends a message to an MTA from a new IP address, where it does not yet have an IPSEC SA with that MTA (see clause 7.3.3.1).

In this optimized approach, the CMS sends a Rekey message instead of the Wake Up. This Rekey message is authenticated with a SHA-1 HMAC, using a **Server Authentication Key**, derived from a session key used to encrypt the last AP Reply sent from the same CMS (or another CMS with the same Kerberos Principal Name).

Additionally, the Rekey message includes IPSEC parameters, to avoid the need for the AP Reply message. The MTA responds with a different version of the AP Request that includes the MTA-CMS Secret, normally sent by the CMS in the AP Reply. As a result, after the MTA responds with the AP Request, a new IPSEC SA can be established with no further messages. The total price for establishing a new IPSEC SA with this optimized approach is a single roundtrip time. This is illustrated in the following figure:

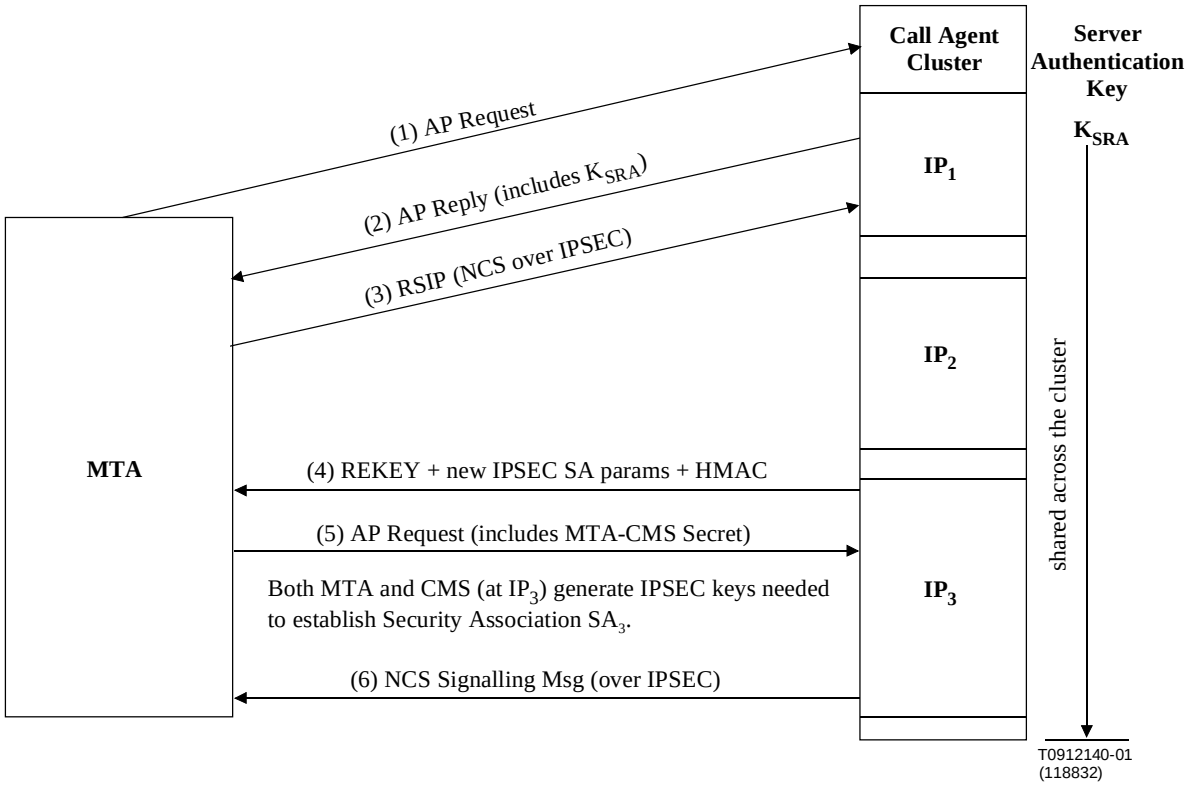



**Figure 10: Key Management for NCS Clusters**

In this figure, an NCS clustered Call Agent suddenly decides to send an NCS message from a new IP address that did not previously have any SA established with that MTA.

The first security association $SA_1$ with CMS at $IP_1$ was established with a basic AP Request/AP Reply exchange. HMAC key $K_{SRA}$ for authenticating Rekey message from the CMS was derived from the session key used to encrypt the AP Reply.

When a new $SA_3$ needs to be established between the MTA and CMS at $IP_3$, the key management is as follows:

1) The CMS at IP3 sends a REKEY message, similar in functionality to the Wake Up message, but with a significantly different content. It contains:

   - IPSEC parameters (also found in the AP Reply): SPI, selected ciphersuite, SA lifetime, grace period, re-establish flag. The purpose of adding these IPSEC parameters to REKEY is to eliminate the need for the subsequent AP Reply message.

   - SHA-1 HMAC using $K_{SRA}$.

2) AP Request that includes the MTA-CMS secret, normally sent in the AP Reply message. This is a legal Kerberos mode, where the key is contained in the AP Request and AP Reply is not used at all.

For more details, refer to clause 7.3.3.1.

#### 8.5.1.4.2 MTA Controlled by Multiple CMSs

In the case a single MTA is controlled by multiple CMSs and each CMS is associated with a different TGS, the MTA will need to execute multiple PKINIT Request/Reply, one for each CMS and then multiple AP Request/Reply in order to create the security association with the individual MGCs.

#### 8.5.1.5 Summary Security Profile Matrix

The CMS to CMS protocol is not defined in IPCablecom in the case NCSv2 signalling is used and thus is outside the scope of the present document. The corresponding column in the following matrix provides only the security requirements on that interface. Security Specifications on that interface will be added in future revisions of the present document.

**Table 13: NCS & CMS-CMS Security Profile Matrix**

| | MGCP (MTA-CMS) | CMS-CMS Interface Security Requirements (see note) |
|---|---|---|
| Authentication | Yes:<br>Two-way authentication of MTA and call agent. | Yes |
| Access control | Yes:<br>Services should only be made available to authenticated subscribers. | Yes |
| Integrity | Yes:<br>To prevent man-in-the-middle attacks. | Yes |
| Confidentiality | Yes:<br>Privacy of call signalling. | Yes |
| Non-repudiation | No. | No |
| Security mechanisms | IPSec ESP in transport mode, encryption and message integrity both enabled.<br>Kerberos with PKINIT Key management. | |
| NOTE: Although (CMS-CMS) is a proprietary interface in 1.0, the following are security requirements for the CMS-CMS interface. | | |

## 8.6 PSTN Gateway Interface

### 8.6.1 Reference Architecture

An IPCablecom PSTN Gateway consists of three functional components:

- a Media Gateway Controller (MGC) which may or may not be part of the CMS;
- a Media Gateway (MG); and
- a Signalling Gateway (SG).

#### 8.6.1.1 Media Gateway Controller

The Media Gateway Controller (MGC) is the PSTN gateway's overall controller. The MGC receives and mediates call-signalling information between the IPCablecom and the PSTN domains (from the SG), and it maintains and controls the overall state for all communications.

#### 8.6.1.2 Media Gateway

Media Gateways (MG) provide the bearer connectivity between the PSTN and the IPCablecom IP network. The MG also interfaces with MTA directly using pkt-s6, pkt-s7, and pkt-s8 IPCablecom security interfaces. These MG to MTA interfaces are discussed in clauses 4.6 and 4.8 of the present document and will not be described here.

#### 8.6.1.3 Signalling Gateway

IPCablecom provides support for C7 signalling gateways. The SG contains the SG to MGC interface. Refer to J.tgcp for more detail on signalling gateways.

The following security-related functions are performed by the C7 Signalling Gateway:

- Isolates the C7 network from the IP network. Guards the C7 network from threats such as Information Leakage, integrity violation, denial of service, and illegitimate use.

- Provides mechanism for certain trusted entities (Transaction Capabilities Application Protocol [TCAP] Users) within the IPCablecom network, such as Call Agents, to query external PSTN databases via TCAP messages sent over the C7 network.

## 8.6.2 Security Services

### 8.6.2.1 MGC-MG Interface

*Authentication*: Both the MG and the MGC must be authenticated, in order to prevent a third party masquerading as either an authorized MGC or MG.

*Access Control*: MG resources should be made available only to authorized users - thus access control is required at the MG.

*Integrity*: must be assured in order to prevent tampering with the TGCP signalling messages - e.g. changing the dialled phone numbers.

*Confidentiality*: TGCP signalling messages carry dialled numbers and other customer information, which must not be disclosed to a third party. Thus confidentiality of the TGCP signalling messages is required.

### 8.6.2.2 MGC-SG Interface

*Authentication*: signalling messages must be authenticated, in order to prevent a third party masquerading as either an authorized MGC or SG.

*Access Control*: Services enable by the NCS signalling should be made available only to authorized users - thus access control is required at the MGC.

*Integrity*: must be assured in order to prevent tampering with the signalling messages - e.g. changing the dialled phone numbers.

*Confidentiality*: NCS messages carry dialled numbers and other customer information, which must not be disclosed to a third party. Thus confidentiality of signalling messages is required.

### 8.6.2.3 CMS-SG Interface

This interface is used for TCAP queries for LNP (Local Number Portability) and other voice communications services.

*Authentication*: TCAP queries must be authenticated, in order to prevent release of information to an unauthorized party.

*Access Control*: required along with the authentication, in order to prevent release of information to an unauthorized party.

*Integrity*: must be assured in order to prevent tampering with the TCAP queries, to prevent a class of denial of service attacks.

*Confidentiality*: TCAP queries contain phone numbers and other subscriber information that MUST be kept private. Thus, confidentiality is required.

## 8.6.3 Cryptographic Mechanisms

### 8.6.3.1 MGC-MG Interface

IPSEC ESP MUST be used to both authenticate and encrypt the messages from MGC to MG and vice versa. Refer to clause 7.1 for details of how IPSEC ESP is used within IPCablecom and for the list of available ciphersuites.

The ISTP protocol allows multiple redundant connections between the SG and the MGC. Multiple connections mean multiple security associations. The assumption is that the number of multiple connections is manageably small, where ahead of time we would set up a security association for each one, using IKE with pre-shared keys.

#### 8.6.3.2 MGC-SG Interface

IPSEC ESP MUST be used to both authenticate and encrypt the messages from MGC to SG and vice versa. Refer to clause 7.1 for details of how IPSEC ESP is used within IPCablecom and for the list of available ciphersuites.

#### 8.6.3.3 CMS-SG Interface

This interface is used for TCAP queries for LNP (Local Number Portability) and other voice communications services. IPSEC ESP MUST be used to both authenticate and encrypt the messages from CMS to SG and vice versa. Refer to clause 7.1 for details of how IPSEC ESP is used within IPCablecom and for the list of available ciphersuites.

### 8.6.4 Key Management

#### 8.6.4.1 MGC-MG Interface

Key management for MGC-MG interface MUST be implemented via IKE. IKE MUST use pre-shared key mode for IPCablecom. For later versions, it may use either public key certificates mode or the pre-shared keys mode. Refer to clause 7.2 of the present document for details on the IPCablecom use of IKE.

IKE will guarantee that there is always a valid, non-expired MGC-MG Secret. This shared secret MUST be unique to this particular interface.

#### 8.6.4.2 MGC-SG Interface

Key management for MGC-SG interface MUST be implemented via IKE. IKE MUST use pre-shared key mode for IPCablecom 1.0. For later versions, it may use either public key certificates mode or the pre-shared keys mode. Refer to clause 7.2 of the present document for details on the IPCablecom use of IKE.

IKE will guarantee that there is always a valid, non-expired MGC-SG Secret. This shared secret MUST be unique to this particular interface.

#### 8.6.4.3 CMS-SG Interface

Key management for CMS-SG interface MUST be implemented via IKE. IKE MUST use pre-shared key mode for IPCablecom. For later versions, it may use either public key certificates mode or the pre-shared keys mode. Refer to clause 7.2 of the present document for details on the IPCablecom use of IKE.

IKE will guarantee that there is always a valid, non-expired CMS-SG Secret. This shared secret MUST be unique to this particular interface.

### 8.6.5 Summary Security Profile Matrix

**Table 14: TCAP, ISUP, TGCP Security Profile Matrix**

| | **TCAP-IP, ISUP-IP (MGC-SG)** | **TGCP (MG-MGC)** | **TCAP-IP (CMS-SG)** |
|---|---|---|---|
| Authentication | Yes | Yes | Yes |
| Access control | Yes | Yes | Yes |
| Integrity | Yes | Yes | Yes |
| Confidentiality | Yes | Yes | Yes |
| Non-repudiation | No | No | No |
| Security mechanisms | • IPSec.<br>• IKE with pre-shared key for IPCablecom.<br>• IKE with pre-shared keys or certificates for later versions. | • IPSec.<br>• IKE with pre-shared key for IPCablecom.<br>• IKE with pre-shared keys or certificates for later versions. | • IPSec.<br>• IKE with pre-shared key for IPCablecom.<br>• IKE with pre-shared keys or certificates for later versions. |

# 8.7 Media Stream

This security Specification allows for end-to-end ciphersuite negotiation, so that the communicating parties can choose their preferred encryption and authentication algorithms for the particular communication.

## 8.7.1 Security Services

### 8.7.1.1 RTP

*Authentication*: end-to-end authentication cannot be required, because the initiating party may want to keep their identity private. Optional end-to-end exchanges for both authentication and additional key negotiation are possible but are outside of the scope for IPCablecom.

*Encryption*: the media stream between MTAs must be encrypted for privacy. Without encryption, the stream is vulnerable to eavesdropping at any point in the network.

Key Distribution via the CMS, a trusted third party, assures the MTA (or MG) that the communication was established through valid signalling procedures, and with a valid subscriber. All this guarantees confidentiality (but not authentication).

*Message Integrity*: it is desirable to provide each packet of the media stream with a message authentication code (MAC). A MAC ensures the receiver that the packet came from the legitimate sender and that it has not been tampered with en route. A MAC defends against a variety of potential known attacks, such as replay, clogging, etc. It also may defend against as-yet-undiscovered attacks. Typically, a MAC consists of eight or more octets appended to the message being protected. In some situations, where data bandwidth is limited, a MAC of this size is inappropriate. As a tradeoff between security and bandwidth utilization, a short MAC consisting or two or four octets is specified and selectable as an option to protect media stream packets. Use of the MAC during an end-to-end connection is optional; whether it is used or not is decided during the end-to-end ciphersuite negotiation (see clause 4.7.1.3).

*Low complexity*: media stream security must be easy to implement. Of particular concern is a PSTN gateway, which may have to apply security to thousands of media streams simultaneously. The encryption and MAC algorithms used with the PSTN gateway must be of low complexity so that it is practical to implement them on such a scale.

### 8.7.1.2 RTCP

*Authentication*: see the above clause.

*Encryption*: some RTCP messages (e.g. CNAME) contain the identity of the endpoint. The requirement is to keep all RTCP messages private, so that for simplicity encryption can be done below the application layer (with IPSEC).

*Message Integrity*: RTCP signalling messages (e.g. BYE) can be manipulated to cause denial of service attacks. To prevent these attacks, message integrity is required for RTCP.

## 8.7.2 Cryptographic Mechanisms for RPT

Each media RTP packet MUST be encrypted for privacy. The MTAs have an ability to negotiate a particular encryption algorithm. Encryption MUST be applied to the packet's payload. Encryption MUST NOT be applied to its header.

Each RTP packet MAY include an optional message authentication code (MAC). The MAC algorithm can also be negotiated. The MAC computation MUST span the packet's unencrypted header and encrypted payload. The receiver MUST perform the same computation as the sender and it MUST discard the received packet if the value in the MAC field does not match the computed value.

Keys for the encryption and MAC calculation MUST be derived from the End-End secret, which is exchanged between sending and receiving MTA as described in clause 8.7.3.

### 8.7.2.1 RTP Packet Format

Figure 12 shows the format of an encoded RTP packet. IPCablecom MUST adhere to the RTP packet format as defined by RFC 1889 and RFC 1890.

The packet's header consists of 12 or more octets, as described in RFC 1889. The only field of the header that is relevant to the encoding process is the timestamp field.

The RTP header has the following format (RFC 1889):

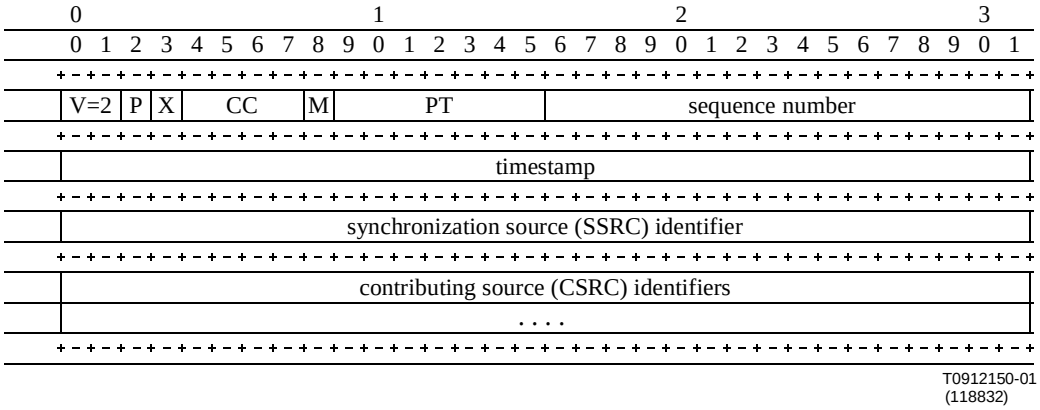



**Figure 11: RTP Header Format**

The first twelve octets are present in every RTP packet, while the list of CSRC identifiers is present only when inserted by a mixer.

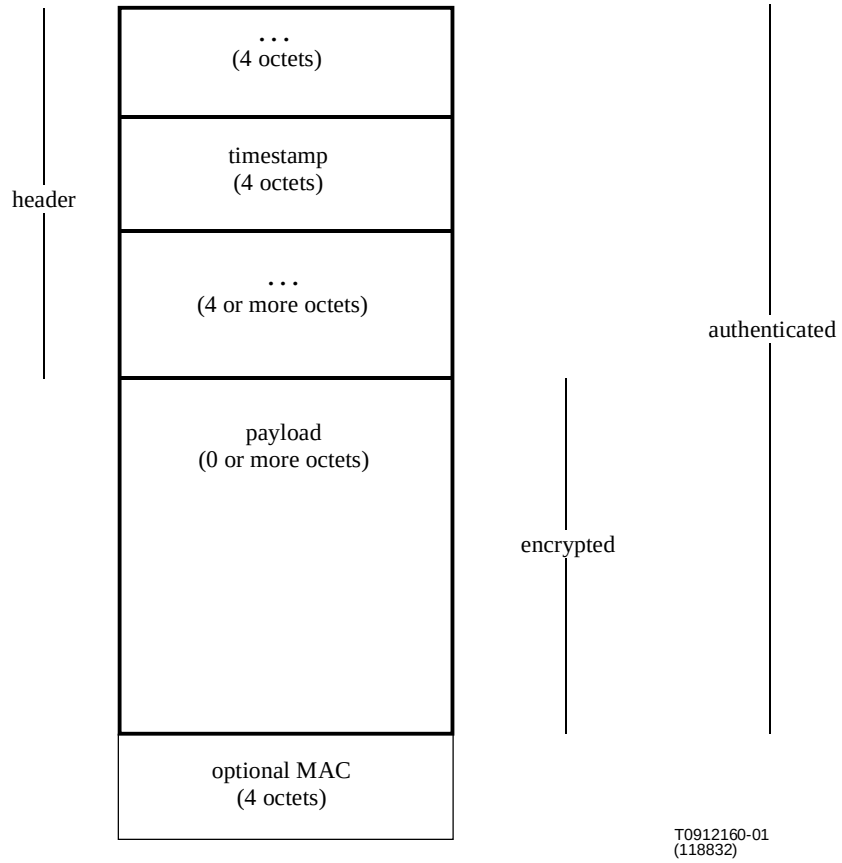



**Figure 12: Format of Encoded RTP Packet**

In IPCablecom, an RTP packet will carry compressed audio from the sender's voice codec, or it will carry a message describing one or more events such as a DTMF tone, trunk or line signalling, etc. For simplicity, the former is referred to as a "voice packet" and the latter as an "event packet".

A voice packet's payload consists of compressed audio from the sender's voice codec. The length of the payload is variable and depends on the voice codec as well as the number of codec frames carried by the packet.

An event packet's payload consists of a message describing the relevant event or events. The format of the message is outside the scope of the present document. The length of the payload is variable, but it will not exceed a known, maximum value.

For either type of packet, the payload MUST be encrypted.

If the optional MAC is selected, the MAC field is appended to the end of the packet after the payload.

Parameters representing RTP packet characteristics are defined as follows:

- $N_c$, the number of octets in one frame of compressed audio. Each codec has a well-defined value of $N_c$. In the case of a codec that encodes silence using short frames, $N_c$ refers to the number of octets in a nonsilent frame.
- $N_u$, the number of speech samples in one frame of uncompressed audio. The number of speech samples represented by a voice packet is an integral multiple of $N_u$.
- $N_f$, the frame number. The first frame of the sender's codec has a value of zero for $N_f$. Subsequent frames increment $N_f$ by one. $N_f$ increments regardless of whether a frame is actually transmitted or discarded as silent.
- $M_f$, the maximum number of frames per packet. $M_f$ is determined by the codec's frame rate and by the sender's packetization rate. The packetization rate is specified during communications setup. For NCS signalling, it is a parameter in the LocalConnectionOptions - see ITU-T Recommendation J.162.

For example, suppose the speech sample rate is 8 000 samples/sec, the frame rate is 10 ms, the packetization rate is 30 ms, and the compressed audio rate is 16 000 bits/s. Then $N_c = 20$, $N_u = 80$, $M_f = 3$, and $N_f$ counts the sequence 0, 1, 2.

- $N_e$, the maximum number of bytes that might be sent in an event packet within the duration of one codec frame. The maximum size of the payload of an event packet is $M_f * N_e$.

NOTE: IPCablecom will use $N_e = N_c$, i.e. an event packet can have a payload as large as that of a voice packet, but no longer. Future versions of IPCablecom may provide a means for $N_e$ to be determined in other ways.

- $N_m$, the number of MAC octets. This value is 0, if the optional MAC is not selected; or 2 or 4, representing the MAC size if the optional MAC is selected.

It is possible for the RTP time stamps to wrap around on a long communication. This causes the RC4 state machine also to wrap around. As a result, the same key stream bytes may be used more than once within a single communication.

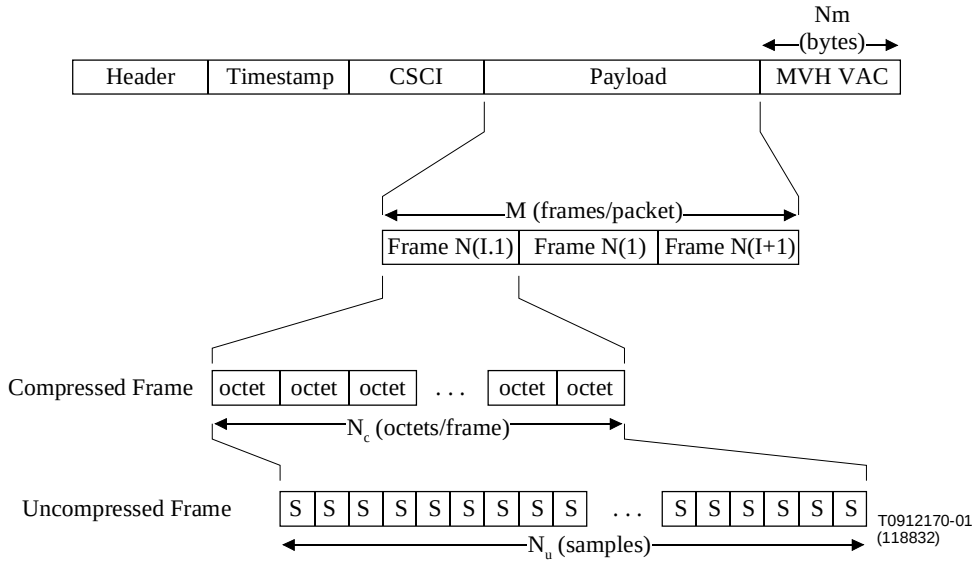



**Figure 13: RTP Packet Profile Characteristics**

#### 8.7.2.2 RTP Timestamp

According to RFC 1889, the timestamp field is a 32-bit value initially chosen at random. According to IPCablecom, the timestamp MUST increment according to the codec sampling frequency. The timestamp in the RTP header MUST reflect the sampling instant of the first octet in each RTP packet presented as offset from the initial random timestamp value. The timestamp field MAY be used by the receiver to synchronize its decryption process to the encryption process of the sender.

Based on the definition of the timestamp and the packet parameters described in the previous clause, the timestamp MUST equate to the value:

$((N_f*N_u)$ + (RTP Initial Timestamp)) modulo 2^32, where $N_f$ is the frame number of the first frame included in the packet.

#### 8.7.2.3 Packet Encoding Requirements

Prior to encoding the packets of an RTP stream, the sending MTA MUST derive the keys and parameters from the End-End Secret it shares with the receiving MTA, as specified in the Derivation of the End-to-End Keys clause.

An MTA MUST derive two distinct sets of these quantities, one set for processing outgoing packets and another set for processing incoming packets.

##### 8.7.2.3.1 Deriving an MMH MAC Key

Key size is $(N_h + N_e + N_m - 1)$, where: $N_h$ is the maximum number of octets in the RTP header; $N_e$ is maximum number of octets in an event packet's payload, as defined in clause 4.7.2.1; and $N_m$ is the number of octets in the MAC. The number of octets in the RTP header ranges from 12 to 72,

inclusive, depending on the number of CSRC identifiers that are included. An implementation MUST choose $N_h$ at least as large as required to accommodate the maximum number of CSRC identifiers that may occur during a session. An implementation MUST set $N_h$ to 72 if the maximum number of CSRC identifiers is otherwise unknown.

##### 8.7.2.3.2 Initializing the RC4 Encryption Process

The following additional parameter is defined for use with RC4:

- $N_k$, the state of the RC4 encryption process. "State" means the number of keystream octets that have been previously generated by the process, whether used or discarded. $N_k$ has value 0 immediately after the RC4 process is initialized with a new key and increments with each generated octet of keystream.

Prior to encoding the first packet, the following procedure MUST be used:

- The new RTP Privacy Key is used to initialize the RC4 encryption process.
- $N_k$ is initialized to 0.
- $N_f$ is initialized to 0.

##### 8.7.2.3.3 Packet Encoding

Each packet MUST be encoded using the following procedure:

- The timestamp is written into the timestamp field of the header. The timestamp MUST equate to the value: $((N_f*N_u)$ + (RTP Initial Timestamp)) modulo 2^32, where $N_f$ is the frame number of the first frame included in the packet.
- All other fields of the header are set to values prescribed in RFC 1889.
- The RC4 encryption state $N_k$ is set to the value $N_f*(N_e + N_m)$.

- The octets of the packet's payload are encrypted using the RC4 encryption process and inserted into the payload field. If there are B octets to be encrypted, then they are encrypted using octets $N_k + N_m$ to $N_k + N_m + B - 1$, inclusive and in order in the RC4 keystream.
- If the MAC option is enabled, the MAC digest is computed using the MMH algorithm with the RTP MAC Key. The digest calculation begins with the first octet of the unencrypted header and ends with the last octet of the encrypted payload. The computed digest is inserted into the MAC field. The digest calculation requires $N_m$ octets of keystream from the RC4 process. These $N_m$ octets are taken from the octets $N_k$ to $N_k + N_m - 1$, inclusive and in order in the RC4 keystream.

Not all of the keystream octets generated by the RC4 process are necessarily used. If a packet contains m frames, then the RC4 state is advanced by $m*(N_e + N_m)$ prior to encoding the next packet. However, only $m*N_c + N_m$ keystream octets are actually used to encode the current packet; the remaining $(m - 1)*N_m + m*(N_e - N_c)$ keystream octets are unused. The RC4 encryption process is advanced for silent codec frames that are not actually transmitted, since the value of $N_f$ increments even for silent frames. For each dropped silent frame, $(N_e + N_m)$ keystream octets are unused. Instead of dropping a silent frame, a codec might encode it using a short frame containing s octets, where $s < N_c$. For such a short frame, $(N_e - s)$ keystream octets are unused.

#### 8.7.2.3.4 Codec change

If the codec is changed without changing the keying material, the sender must continue its RC4 encryption process from its state prior to the codec change. It cannot reset the state to 0, since a stream cipher such as RC4 may not reuse keystream without exposing plaintext. The sender also needs to preserve continuity of the timestamp across the codec change, since the timestamp reflects real time.

Changing the codec is likely to change the frame parameters. This changes the proportionality constant that relates the RC4 state to the timestamp. To preserve continuity of the RC4 state and the timestamp across the codec change, the sender MUST compute a new frame number. The new frame number is applied to the first frame generated by the new codec.

The new frame number MUST be calculated as follows:

$$N_{f,new} = \text{roof}\ ((N_{f,old} + 1)*(N_{e,old} + N_m)/(N_{e,new} + N_m))$$

where: $N_{f,new}$ is the new frame number; $N_{f,old}$ is the frame number of the last frame generated by the old codec, whether transmitted or dropped as silent; $N_{e,old}$ and $N_{e,new}$ are the values of $N_e$ under the old and new codecs, respectively; and roof(x) is the function that returns the smallest integer no less than x. The new frame number is applied to the first frame generated by the new codec; thereafter, the frame number is incremented by 1.

To encode the first packet containing frames of the new codec, the sender will, in general, need to advance its RC4 state slightly from its state after processing the last packet under the old codec. This is because of rounding error introduced by requiring the frame number to be an integer value. The amount by which the RC4 state needs to be advanced is less than $(N_{e,new} + N_m)$.

#### 8.7.2.3.5 Change in the MAC Size

The MTA that initiated a call has to be prepared to receive RTP packets before ciphersuite negotiation has been completed. Once the negotiation is complete, it may turn out that the negotiated ciphersuite uses a MAC of a different size than what the initiating MTA assumed previously. At that point, the MTA will have to re-adjust the frame number based on the new MAC size.

This procedure of re-adjusting the frame number is almost identical to the one described in clause above for codec changes. The only difference, is that it is the MAC size $N_m$ and not the codec that changes:

$$N_{f,new} = \text{roof}\ ((N_{f,old} + 1)*(N_e + N_{m,old})/(N_e + N_{m,new}))$$

Here, $N_{m,old}$ and $N_{m,new}$ are the old and new MAC sizes respectively.

#### 8.7.2.3.6 Block Cipher Encryption of RTP Packets

If an implementation supports block ciphers, the residual block termination (RBT) MUST be used to terminate streams that end with less than a full block of data to encrypt (see clause on Block Termination).

#### 8.7.2.3.7 Block Cipher Initialization Vector

An 8-byte initialization value (IV) is required when using 8-byte block ciphers in CBC mode to encrypt RTP packet payloads. The IV MUST be calculated new for each RTP packet using the ECB mode of the same block encryption algorithm defined for encrypting the payload. The first 64 bits

of each RTP packet header (including sequence-number and timestamp) are first X-OR'ed with a 64-bit secret value. The secret value is calculated from the End-to-End secret. The result of the X-OR is encrypted by the block cipher using the same key(s) that will be used to encrypt the RTP payload. The resulting ciphertext will be the 64-bit IV.

#### 8.7.2.3.8 MMH-MAC Pad Derivation When Using a Block Cipher

A method for deriving the MMH-MAC pad when using RC4 was explained in an earlier clause. When using a block cipher, the pad is calculated using the block cipher. When the block cipher requires an IV, the IV value is calculated according to the clause on the initialization vector. This value will serve as the basis of the MMH-MAC pad when using a block cipher. If the MMH-MAC is used with a block cipher that does not require an IV, a corresponding value MUST be calculated according to the clause on the initialization vector and used as the basis of the MMH-MAC pad according to this clause.

The pad is subsequently calculated by performing the MMH digest function on the resulting IV and then using the appropriate number of most-significant-bytes for the MMH-MAC pad.

Additional keying material needed to calculate the digest for the pad is derived with the MMH-MAC key from the shared secret as specified in clause on the derivation of end-to-end keys.

### 8.7.2.4 Packet Decoding Requirements

Prior to decoding the packets of an RTP stream, the receiving MTA MUST derive the keys and parameters from the End-End Secret it shares with the sending MTA, as specified in clause on the derivation of end-to-end keys.

The derived quantities MUST match the corresponding quantities at the sending MTA.

#### 8.7.2.4.1 Timestamp Tolerance Check

Before processing a received packet, the receiver SHOULD perform a sanity check on the timestamp value in the RTP header, consisting of the items 1) through 4) below:

1) Beginning with the RTP timestamp in the first packet received from a sender, the receiver calculates an expected value for the timestamp of the sender's next RTP packet based on timestamps received in the sender's previous packets for the session.

2) The next packet is rejected without being processed if its timestamp value is outside a reasonable tolerance of the expected value. (Timestamps from rejected packets are not to be used to predict future packets). The tolerance value is defined to be:

   a) sufficiently tight to ensure that an invalid timestamp value cannot derail the receiver's state so much that it cannot quickly recover to decrypting valid packets; and

   b) able to account for known differences in the expected and received timestamp values, such as might occur at call startup, codec switch over and due to sender/receiver clock drift.

3) If the timestamp value in the RTP headers from a sender never comes back within the acceptable range, the receiver discontinues the session.

4) At the receipt of each packet, the receiver adjusts its time relationship with the sender within the acceptable tolerance range of estimated values.

#### 8.7.2.4.2 Packet Authentication

If authentication is used on an RTP packet stream, verification of the MAC MUST be the first step in the packet decoding process. When the timestamp tolerance check is performed, the MAC MAY be verified on packets with valid RTP timestamps immediately after the check is completed.

If the MAC does not verify, the packet MUST be rejected.

#### 8.7.2.4.3 RC4 Decryption and MMH MAC

Prior to decoding the first packet, the RTP Privacy Key is used to initialize the RC4 decryption process state $N_k$ to zero.

Each packet MUST be decoded using the following procedure:

- The frame number for the first frame in the packet, $N_f$, is computed from the value of the timestamp field in the header as follows:
  - if the value of the timestamp is greater than or equal to the value of RTP Initial Timestamp, then $N_f$ = (timestamp – (RTP Initial Timestamp))/$N_u$;
  - otherwise, $N_f$ = (timestamp + $2^{32}$ – (RTP Initial Timestamp)/$N_u$.
- If the computed value of $N_f$ is not an integer value, the packet is discarded; this indicates an invalid timestamp.
- The RC4 decryption state $N_k$ is set to the value $N_f$*($N_e$ + $N_m$).
- If the MAC option is enabled, a MAC digest is computed using the MMH algorithm with the RTP MAC key. The digest calculation begins with the first octet of the unencrypted header and ends with the last octet of the encrypted payload. The digest calculation requires $N_m$ octets of keystream from the RC4 process. These $N_m$ octets are taken from the octets $N_k$ to $N_k$ + $N_m$ – 1, inclusive and in order in the RC4 keystream. The computed digest is compared to the value in the MAC field. If the computed digest does not match the value in the MAC field, the packet is discarded.
- The octets of the packet's payload are decrypted using the RC4 decryption process. If there are B octets to be decrypted, then they are decrypted using octets $N_k$ + $N_m$ to $N_k$ + $N_m$ + B – 1, inclusive and in order in the RC4 keystream.
- All other fields of the header are processed as prescribed in RFC 1889.

Note that the state of the RC4 decryption process is adjusted to match the state of the sender's RC4 encryption process prior to decrypting the packet's payload or verifying its MAC digest. If packets arrive out of order, the receiver must, in principle, push the RC4 process backwards as well as forwards in order to match the state of the sender's RC4 process. In practice, this can be accomplished by having the receiver run its RC4 process in the forward direction only and synchronized to real time, thus making keystream available to decode packets in whatever order they arrive.

## 8.7.3 Key Management

The key management specified here for end-to-end communication is identical in the cases of the MTA–to-PSTN and MTA-to-MTA communications. In the case of the MTA-to-PSTN communications, simply one of the MTAs is replaced by a MG (Media Gateway).

***The descriptions below refer to MTA-to-MTA communications only for simplicity. In this context, an MTA actually means a communication end point, which can be an MTA or a MG. In the case that the end point is a MG, it is controlled by an MGC instead of a CMS.***

At the start of each voice communication, the CMS in the source MTA's domain shall securely distribute a secret to both MTAs. This secret may be used by both MTAs to derive the keying material for both the bearer channel and the MTA-MTA signalling messages (for both DCS and RTCP).

The MTAs may choose to ignore this secret distributed by the CMS and negotiate their own keys. However, direct MTA-MTA key negotiation is outside of the scope of IPCablecom

The distribution of End-End Secret is specific to the call signalling described in the following clauses.

### 8.7.3.1 Key Management over NCS

Figure 14 shows the actual NCS messages that are used to carry out the distribution of end-to-end keys. Each NCS message that is involved in the end-to-end key management is labelled with a number of the corresponding key management interface.

The name of each NCS message is in bold. Below the NCS message name is the information needed in the NCS message, in order to perform end-to-end key distribution. Messages between the CMSs are labelled as SIP+ messages. However, NCS has not yet defined the CMS-CMS protocol and SIP+ is only one possible choice.

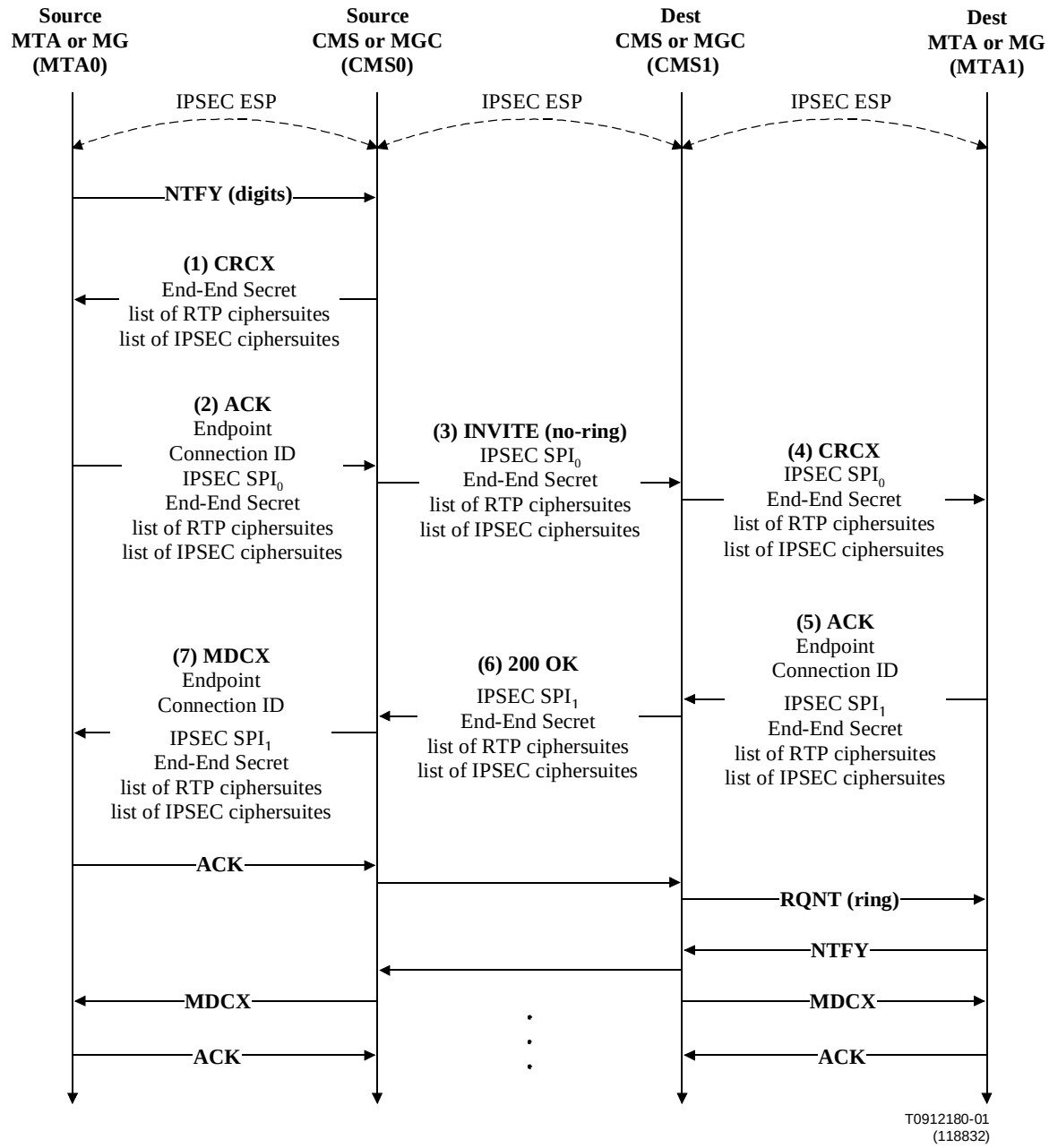



**Figure 14: End-End Secret Distribution over NCS**

This figure shows that before the start of this scenario, both the source and destination MTAs had already established an IPSEC ESP session with their local CMS. Similarly, there is a pre-existing IPSEC ESP session between the pair of CMSs. This allows the End-End Secret to be distributed securely, with privacy, integrity and anti-replay mechanisms already in place. Each of the numbered flows in figure 14 is described below:

1) CMS0 -> MTA0

CMS0 sends to the MTA0 the allowable lists of ciphersuites for the new communication - one list for RTP security and one for RTCP (over IPSEC ESP). It SHOULD also generate an End-End Secret to be used for this communication and send it to the MTA0 along with the ciphersuites. If CMS0 chooses not to generate the secret, it will be up to MTA0 to do it in the subsequent step. This information is sent with a **CreateConnection** (CRCX) command, inside the **LocalConnectionOptions** parameter.

2) MTA0 -> CMS0

MTA0 generates Connection ID for this particular connection. Thus, the End-End Secret is now associated with a Connection ID. The pair (Connection ID, Endpoint) uniquely identifies this connection, where the Endpoint is an NCSv2 identifier for MTA0.

From the list of ciphersuites in the **LocalConnectionOptions**, MTA0 MUST select a (non-empty) subset for RTP and a (non-empty) subset for RTCP and returns them in the subsequent ACK message, in the **LocalConnectionDescriptor**, in the form of Session Description Protocol(SDP) attributes. The resulting lists of ciphersuites are in a prioritized order - the preferred ciphersuites MUST be listed first.

MTA0 SHOULD take the End-End Secret from the **LocalConnectionOptions** and return it along with the ciphersuites in the **LocalConnectionDescriptor**. MTA0 MAY also generate a new End-End Secret and include it in the **LocalConnectionDescriptor** instead. In the case that CMS0 did not send the End-End Secret, MTA0 itself MUST generate the secret.

The ACK also includes the Connection ID and the Endpoint for MTA0.

In order to receive RTCP packets over IPSEC, MTA0 generates a locally unique 4-byte identifier, called SPI (Security Parameters Index). This SPI ($SPI_0$) MUST also be included in the **LocalConnectionDescriptor**, used to identify the IPSEC Security Association for the incoming RTCP packets.

Right after sending this message, MTA0 SHOULD establish its inbound RTP and RTCP (IPSEC ESP) SAs, based on the first (preferred) ciphersuite in the RTP and RTCP lists. MTA0 SHOULD be ready to receive RTP and RTCP messages, which may arrive any time after this message is received by the CMS.

If later MTA1 decides to use alternate ciphersuites listed by MTA0, MTA0 will later have to update its RTP and RTCP Security Associations. If MTA1 also decides to send MTA0 packets before ciphersuite negotiation had completed, processing on those packets at MTA0 will fail (since it assumed a different ciphersuite).

3) CMS0 -> CMS1

The CMS0 MUST send the End-End Secret and a list of ciphersuites selected by MTA0 to MTA1's local CMS (CMS1). CMS1 will later forward this information to MTA1. Although this message is shown in the figure as a SIP+ **INVITE**, the CMS-CMS protocol has not yet been specified by NCS.

4) CMS1 -> MTA1

CMS1 MUST relay the same End-End Secret (generated by CMS0 or MTA0) for the incoming communication to MTA1 along with the lists of ciphersuites selected by MTA0 - one list for RTP security and one for RTCP (over IPSEC ESP). This information MUST be sent with a **CreateConnection** (CRCX) command, inside the **LocalConnectionOptions** parameter. (It MAY also be present in the **RemoteConnectionDescriptor** parameter in the form of SDP attributes).

5) MTA1 -> CMS1

MTA1 MUST generate Connection ID for this particular connection. Thus, the End-End Secret is now associated with a pair (Endpoint, Connection ID).

From the list of ciphersuites in the **LocalConnectionOption**, MTA1 SHOULD select the first listed ciphersuite for RTP and the first one for RTCP. This allows MTA1 to immediately start sending RTP and RTCP packets to MTA0. MTA1 MAY also select alternate ciphersuites specified by MTA0. In that case, MTA1 SHOULD not try to send any packets to MTA0 until it is certain that MTA0 had been informed of the selected ciphersuites - after receiving a message labelled RQNT (ring) in figure 14.

MTA1 MUST send back an ACK message, which includes lists of the selected ciphersuites inside the **LocalConnectionDescriptor**, in the form of SDP attributes. The 1st ciphersuite in each list (one for RTP and one for RTCP) MUST be the one that was already selected by MTA1. Additional ciphersuites in each list are alternatives. If at any time, MTA0 wants to switch to one of the alternatives that were selected by MTA1, it would have to go through a new key negotiation. The ACK MUST include the Connection ID.

In order to receive RTCP packets over IPSEC, MTA1 generates a locally unique 4-byte identifier, $SPI_1$, used to identify the IPSEC Security Association for the incoming RTCP packets, and MUST include it in the **LocalConnectionDescriptor**.

The SDP attributes MUST also include the End-End Secret, even though the MTA1 does not have an option of changing it. The End-End Secret is sent for consistency and to verify that MTA1 is using the right key.

At this time, MTA1 creates both RTP and RTCP (IPSEC ESP) Security Associations, for both inbound and outbound messages, and is ready to send and receive RTP and RTCP messages for this communication.

6) CMS1 -> CMS0

CMS1 MUST forward the acknowledgement and the selected ciphersuites from MTA1 to CMS0.

7) CMS0 -> MTA0

CMS0 MUST now send a **ModifyConnection** command to MTA0. If MTA1 chose a ciphersuite that is different from MTA0's preferred choice (sent in step 2), the ciphersuites (and alternatives) MUST be included in the **LocalConnectionOptions**.

MTA0 MUST check if the first RTP and RTCP ciphersuites in the **LocalConnectionOptions** differ from the ones that it selected in step 2). If either ciphersuite had been changed, MTA0 MUST update the corresponding inbound Security Association. For updating RTP key stream when the MAC size had changed see clause labelled change in the MAC size.

**RemoteConnectionDescriptor** MAY also include the ciphersuites (and alternatives) selected by MTA1, as well as the End-End Secret, which MAY be used to verify that MTA1 is using the right key.

At this time, MTA0 MUST also establish its outbound RTP and RTCP (IPSEC ESP) SAs. MTA0 is now ready to send (in addition to receiving) RTP and RTCP messages to MTA1.

For full syntax of the NCS messages, including the End-End Secret, list of ciphersuites, and other related information, please refer to the NCSv2 signalling TS 101 909-4.

#### 8.7.3.1.1 Ciphersuite Format

Each ciphersuite for both bearer channel and signalling security (via IPSEC) MUST be represented as follows:

| **Authentication Algorithm** (1 byte) represented by 2 ASCII hex characters (using characters 0-9, A-F). | **Encryption Transform ID** (1 byte) represented by 2 ASCII hex characters (using characters 0-9, A-F). |
|---|---|

For the list of available transforms and their values, refer to clause 7.1 for IPSEC, and to clause 7.3.4 for the RTP security. For the exact syntax of how the Authentication Algorithm and the Encryption Transform ID are included in the signalling messages, refer to J.162.

#### 8.7.3.1.2 Initial End-End Key Derivation

After the receipt of the End-End Secret, each MTA MUST independently derive the keys it needs to secure all MTA-MTA communication. The size of the End-End Secret is 46 bytes (the same as with the SSL or TLS pre-master secret).

All keys are derived in pairs, since each individual authentication or encryption key is applied to a unidirectional flow. The keys MUST be derived as follows, in the specified order:

1) RTP (bearer channel security). Derive two sets of the following keys, one for sending packets and one for receiving. The derivation function is **F(S, "End-End RTP Security Association")**. Here, S is the End-End Secret and the string "End-End RTP Security Association" is taken without quotes and without a terminating null character. F is defined in clause 10.6.

   a) RTP privacy key. The only RTP encryption algorithm currently defined is RC4 with 128-bit keys.

   b) RTP Initial Timestamp (integer value, 4 octets, Big Endian byte order).

   c) RTP Initialization Key (required when using a block cipher to encrypt the RTP payload) a 64-bit value used to derive the 64-bit IV according to initialization vector, clause 8.7. The resulting IV is used for the block cipher in CBC mode (if applicable) and for the random pad used to calculate the MMH-MAC.

d) RTP packet MAC key (if MAC option is selected). The requirements for the MMH MAC key can be found in clause 10.7 labelled MMH-MAC Pad Derivation Using a Block Cipher.

2) IPSEC ESP (for RTCP SAs). Derive two sets of the following keys, one for sending packets and one for receiving. The derivation function is F(S, "End-End RTP Control Protocol Security Association").

a) Message authentication key.

b) Encryption key.

For possible message authentication and encryption algorithms that can be used with IPSEC, and for the corresponding key sizes, refer to clause 7.1.

For each set of keys and security parameters specified above, the MTA (or MG) that initiated the communication MUST derive send parameters first and receive parameters second. The target MTA (or MG) that receives the communication MUST derive receive parameters first and send parameters second.

It is also possible that a MTA is communicating to a regular phone, via a POTS Signalling Gateway. In that case, the scenario is almost identical, except that the destination MTA is replaced with the POTS Signalling Gateway.

Similarly, it is possible for a regular phone to call up a MTA, via a POTS Signalling Gateway. Again, the scenario is almost identical, except that the source MTA is replaced with the POTS Signalling Gateway.

#### 8.7.3.1.3 End-to-End Rekey Derivation

When key parameters need to be changed for a current RTP session, they MUST be generated independently by the two endpoints. The derivation function is F(S, "End-End RTP Key Change <N>"). Here, S is the same End-End Secret used to derive the original set of keys for this RTP connection. "End-End RTP Key Change <N>" is taken without quotes and without a terminating NULL character.

The value <N> stands for a counter which will have a value 1 for the 1st key change, then 2, 3, etc. The function F is defined in clause 6.6.

The RTCP keys are not derived during this key change.

The key change derivation MUST occur before the timestamp value rolls-over to a value equal-to or past its initial value from the previous key derivation. The new key parameters MUST be ready ahead of the time they are needed, and not used until the timestamp value rolls-over to a value equal-to or past its initial value. Both the sender and the received MUST maintain both old and new sets of parameters during the roll-over transition of the timestamp. Once a sender begins using the new key set, it MUST NOT use the old keys again for outgoing packets. Once the timestamp value requiring the old key parameter set is outside the acceptable tolerance for incoming timestamps, the receiver MUST delete its old key parameters.

### 8.7.4 Summary Security Profile Matrix

**Table 15: RTP/RTCP Security Profile Matrix**

| | RTP (MTA-MTA, MTA-MG) | RTCP (MTA-MTA, MTA-MG, MG-MG) |
|---|---|---|
| Authentication | Yes (indirect) (see note) | Yes (indirect) |
| Access control | Optional | Optional |
| Integrity | Optional | Yes |
| Confidentiality | Yes | Yes |
| Non-repudiation | No | No |
| Security mechanisms | Application Layer Security via RTP IPCablecom Security Profile.<br>End-to-End Secret distributed over secured MTA-CMS links. Final keys derived from this secret.<br>RC4-128 encryption algorithm.<br>Optional 2-byte or 4-byte MAC based on MMH algorithm.<br>IPCablecom requires support for ciphersuite negotiation. Currently, RC4-128 is required for encryption and MMH-based MACs of various sizes for message authentication are optional. | IPSec ESP in transport mode with both encryption and message integrity enabled.<br>The UDP check sum may need to be turned off (set to 0), depending on how NAT is implemented.<br>Keys derived from the same End-to-End Secret as the one used to derive RTP keys. |
| NOTE: MTAs do not authenticate directly. Authentication refers to the authentication of identity. | | |

## 8.8 Announcement Services

### 8.8.1 Reference Architecture

Figure 15 shows the network components and the various interfaces to be discussed in this clause.

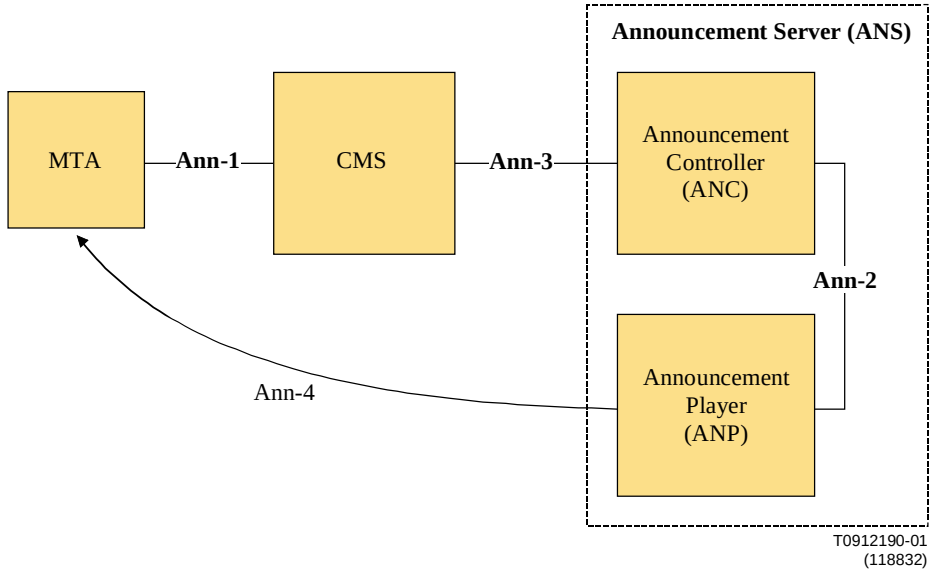



**Figure 15: Announcement Services Reference Architecture**

The figure shows a network-based Announcement Player (ANP). It has an optional TGCP interface (Ann-2) to the Announcement Controller (ANC), in the case that ANC and ANP are not integrated into a single physical entity. Security on this interface is specified in this clause.

There is also an NCS signalling interface (Ann-1) between the MTA and CMS. Refer to clause 8.5.1 for NCS signalling security. There is also a signalling interface (Ann-3) between the CMS and the ANC. This interface is proprietary for IPCablecom, and thus the corresponding security interface is not specified (although this clause lists recommended security services for Ann-3).

Finally, there is a media stream (RTP and RTCP) interface between the MTA and the ANP. This is a standard media stream interface, for which security is defined in clause 8.7 of the present document.

The Announcement Services Specification will also allows local playout of announcements at the MTA. In those cases, an announcement is initiated with NCS signalling between the MTA and the CMS (interface Ann-1). No other interfaces are needed for MTA-based announcement services.

## 8.8.2 Security Services

### 8.8.2.1 MTA-CMS NCS Signalling (Ann-1)

Refer to the security services in the NCS signalling clause 8.5.1.2.

### 8.8.2.2 ANC-ANP Signalling (Ann-2)

*Authentication*: all signalling messages must be authenticated, in order to prevent a third party masquerading as either an authorized ANC or ANP. A rogue ANC could configure the ANP to play obscene or inappropriate messages. A rogue ANP could likewise play obscene or inappropriate messages that the ANC did not intend it to play. If ANP is unable to authenticate to the ANC, the ANC should not pass it the key for media packets, preventing unauthorized announcement playout.

*Confidentiality*: if a snooper is able to monitor TGCP signalling messages on this interface, he or she might determine which services are used by a particular subscriber or which destinations a subscriber is communicating to. This information could then be sold for marketing purposes or simply used to spy on other subscribers. Thus, confidentiality is required on this interface.

*Message integrity*: must be assured in order to prevent tampering with signalling messages. This could lead to playout of obscene or inappropriate messages - see authentication above.

*Access control*: an ANC should keep a list of valid Announcement Players and which announcements are supported by each. Along with authentication, this insures that wrong announcements are not played out.

### 8.8.2.3 MTA-ANP (Ann-4)

Security services on this media packet interface are listed in clause 8.7.1.

## 8.8.3 Cryptographic Mechanisms

### 8.8.3.1 MTA-CMS NCS Signalling (Ann-1)

Refer to the cryptographic mechanisms in the NCS signalling clause 8.5.1.3.

### 8.8.3.2 ANC-ANP Signalling (Ann-2)

IPSEC ESP MUST be used to both authenticate and encrypt the messages from ANC to ANP and vice versa. Refer to clause 7.1 for details of how IPSEC ESP is used within IPCablecom and for the list of available ciphersuites.

The ANC MUST verify for each signalling message received from the ANP, that the ANP domain name (included in the **Endpoint ID**) correctly corresponds to its IP address. This check is done via a lookup into a map of IP addresses to ANP domain names. Also, in reverse, when the ANC performs a lookup of the ANP IP address based on its domain name (in order to send a message to the ANP), it consults this same map instead of performing a DNS query. Refer to clause 8.4.3 on how this map is maintained.

#### 8.8.3.3 MTA-ANP (Ann-4)

Cryptographic mechanisms on this media packet interface are specified in clause 8.7.2.

### 8.8.4 Key Management

#### 8.8.4.1 MTA-CMS NCS Signalling (Ann-1)

Refer to the key management in the NCS signalling clause 8.5.1.4.

#### 8.8.4.2 ANC-ANP Signalling (Ann-2)

ANC and ANP will negotiate a shared secret (ANC-ANP Secret) using IKE. IKE may use one of the modes with pre-shared keys. Certificates may be used in future versions of IPCablecom. For details, refer to clause 7.1.2.3.

IKE will be running asynchronous to the signalling messages and will guarantee that there is always a valid, non-expired ANC-ANP Secret. This shared secret MUST be unique to this particular ANC and ANP.

At the ANC, ANP domain names MUST somehow be associated with the corresponding IP addresses. One possibility is to associate each pre-shared key directly with the domain name. IKE

negotiations will use an ISAKMP identity payload of type ID_KEY_ID to identify the pre-shared key. The value in that identity payload will be the ANP domain name. For more details refer to RFC 2409.

Later, when a TGCP signalling message arrives at the ANC, it will be able to query the database of IPSEC SAs and retrieve a source IP address, based on the ANP domain name. The ANC will make sure that it is the same as the source IP address in the IP packet header. One way to query this database is through SNMP, using an IPSEC Monitoring MIB.

Similarly, rather than doing a DNS query to find an IP address of the correct ANP, ANC will look it up in the database of IPSEC SAs. Since we are not requiring the use of DNSSEC on this interface, responses to DNS queries can be spoofed.

#### 8.8.4.3 MTA-ANP (Ann-4)

Key Management on the media packet interface is specified in clause 8.7.3. This case is very similar to the key management for the MTA-MG media interface. The flow of signalling messages and the syntax of carrying keys and ciphersuites is the same, except that here MG is replaced with the ANP and MGC (which delivers the key to MG) is replaced with ANC (and delivers the key to ANP).

### 8.8.5 Summary Security Profile Matrix

The CMS to ANC protocol is not defined in IPCablecom and thus is outside the scope of the present document. The corresponding column in the following matrix provides only the security requirements on that interface. Security Specifications on that interface will be added in future revisions of the present document.

**Table 16: Announcement Services Security Profile Matrix**

| | Ann-1: NCS (MTA-CMS) | Ann-2: TGCP (ANC-ANP) | Ann-3: unspecified (CMS-ANC) Interface Security Requirements (see note) | Ann-4: RTP (MTA-ANP) | Ann-4: RTCP (MTA-ANP) |
|---|---|---|---|---|---|
| Authentication | Yes | Yes | Yes | Yes (indirect) | Yes (indirect) |
| Access control | Yes | Yes | Yes | Optional | Optional |
| Integrity | Yes | Yes | Yes | Optional | Yes |
| Confidentiality | Yes | Yes | Yes | Yes | Yes |
| Non-repudiation | No | No | No | No | No |
| Security mechanisms | IPSec ESP in transport mode, encryption and message integrity both enabled. Kerberos with PKINIT Key management. | IPSec. IKE with pre-shared keys for IPCablecom. IKE with pre-shared keys or certificates for later versions. | | Application Layer Security via RTP IPCablecom Security Profile. keys distributed over secured MTA-CMS and ANP-ANC links. RC4-128 encryption algorithm. Optional 2-byte or 4-byte MAC based on MMH algorithm. | IPSec ESP in transport mode with both. encryption and message integrity enabled. Keys derived from bearer channel RTP SA. |
| NOTE Although (CMS-ANC) is a proprietary interface in 1.0, the following are security requirements for the CMS-ANC interface. | | | | | |

# 8.9 Electronic Surveillance Interfaces

## 8.9.1 Reference Architecture

The IPCablecom system for Electronic Surveillance consists of the following elements and interfaces:

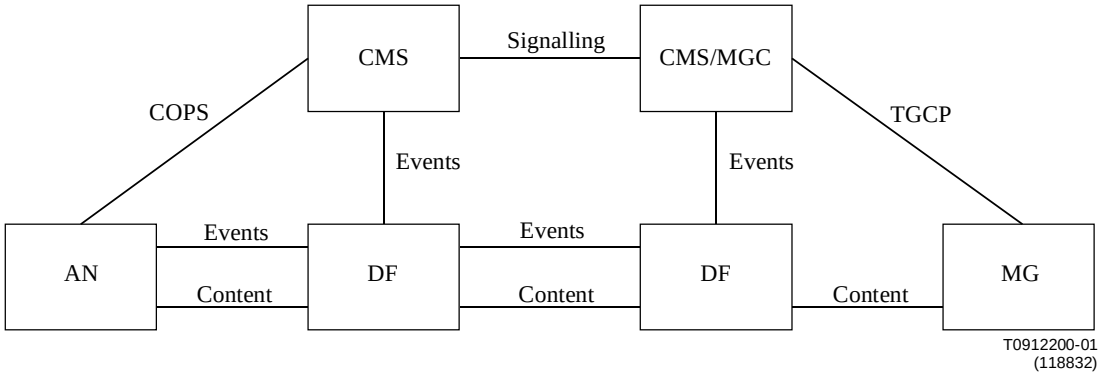



**Figure 16: Electronic Surveillance Security Interfaces**

THE DF (DELIVERY FUNCTION) IN THIS FIGURE IS RESPONSIBLE FOR REDIRECTING DUPLICATED MEDIA PACKETS TO LAW ENFORCEMENT, FOR THE PURPOSE OF WIRETAPPING.

The event interface between the CMS and the DF provides descriptions of calls, necessary to perform wiretapping. That information includes the media stream encryption key and the corresponding encryption algorithm. This event interface uses Radius and is similar to the CMS-RKS interface.

The COPS interface between the CMS and the AN is used to signal the AN to start/stop duplicating media packets to the DF for a particular phone call. This is the same COPS interface that is used for (DQoS) Gate Authorization messages. For the corresponding security services, refer to clauses 4.2.1.2.3, 4.2.1.3.3 and 4.2.1.4.2.

The TGCP signalling interface between the CMS/MGC and MG is used to signal the MG to start/stop duplicating media packets to the DF for a particular phone call. This is the same TGCP signalling interface that is used during call set-up on the PSTN Gateway side. For the corresponding security services, refer to clauses 4.6.2.1, 4.6.3.1 and 4.6.4.1.

The event interface between the AN and DF is needed to tell the DF when the actual call begins and when it ends. In IPCablecom, the start and end of the actual call is signalled with Radius event messages generated by the AN.

The interface between the AN and DF for call content is where the AN encapsulates copies of the RTP media packets - including the original IP header - inside UDP and forwards them to the DF. Since the original media packets are already encrypted (and optionally authenticated), no additional security is defined on this interface.

The Specification of the signalling interface between the two CMSs is out of scope for IPCablecom. The Specification of the corresponding security interface is also out of scope.

The event interface between the two DFs is used to forward call information in the case where a wiretapped call is forwarded to another location that is wiretapped using a different DF. This interface utilizes the Radius protocol - the same as all other event message interfaces.

The interface between the two DFs for call content is used to forward media packets (including the original IP header) in the case where a wiretapped call is forwarded to another location that is wiretapped using a different DF. Since the original media packets are already encrypted (and optionally authenticated), no additional security is defined on this interface.

## 8.9.2 Security Services

### 8.9.2.1 Event Interfaces CMS-DF, AN-DF and DF-DF

*Authentication, Access Control and Message Integrity:* required to prevent service theft and denial of service attacks. Want to insure that the DF (law enforcement) has the right parameters for wiretapping (prevent denial of service). Also, want to authenticate the DF, to make sure that the copy of the media stream is directed to the right place (protect privacy).

*Confidentiality*: required to protect subscriber information and communication patterns.

### 8.9.2.2 Call Content Interfaces AN-DF and DF-DF

*Authentication and Access Control*: already performed during the phase of key management for protection of event messages - see the above clause. In order to protect privacy, a party that is not properly authorized should not receive the call content decryption key.

*Message Integrity*: optional for voice packets, since it is generally hard to make undetected changes to voice packets. No additional security is required here - an optional integrity check would be placed into the media packets by the source (MTA or MG).

*Confidentiality*: required to protect call content from unauthorized snooping.

However, no additional security is required in this case - the packets had been previously encrypted by the source (MTA or MG).

## 8.9.3 Cryptographic Mechanisms

### 8.9.3.1 Interface between CMS and DF

This interface MUST be protected with IPSec ESP in transport mode, where each packet is both encrypted and authenticated - identical to the security for the CMS-RKS interface specified in clause 4.4.2.

Also the same as with the CMS-RKS interface, the MAC value normally used to authenticate Radius messages is not used (message integrity is provided with IPSec).

The key for this Radius MAC MUST always be hardcoded to 160-bytes.

#### 8.9.3.2 Interface between AN and DF for Event Messages

This interface MUST be protected with IPSec ESP in transport mode, where each packet is both encrypted and authenticated - identical to the security for the AN-RKS interface specified in clause 4.4.2.

Also the same as with the AN-RKS interface, the MAC value normally used to authenticate Radius messages is not used (message integrity is provided with IPSec).

The key for this Radius MAC MUST always be hardcoded to 160-bytes.

#### 8.9.3.3 Interface between DF and DF for Event Messages

This interface MUST be protected with IPSec ESP in transport mode, where each packet is both encrypted and authenticated - identical to the security for the CMS-RKS interface specified in clause 4.4.2.

Also the same as with the CMS-RKS interface, the MAC value normally used to authenticate Radius messages is not used (message integrity is provided with IPSec).

The key for this Radius MAC MUST always be hardcoded to 160-bytes.

### 8.9.4 Key Management

#### 8.9.4.1 Interface between CMS and DF

CMS and DF MUST negotiate a pair of IPSec Security Associations (inbound and outbound) using IKE with pre-shared keys.

IKE will be running asynchronous to the event message generation and will guarantee that there is always a valid, non-expired pair of SAs. The corresponding IPSec keys MUST be unique to this particular CMS and DF.

At the DF, CMS Element IDs MUST somehow be associated with the corresponding IP addresses. One possibility is to associate each pre-shared key directly with the Element ID. IKE negotiations will use an ISAKMP identity payload of type ID_KEY_ID to identify the pre-shared key. The value in that identity payload will be the Element ID used in event messages.

Later, when an event message arrives at the DF, it will be able to query the database of IPSEC SAs and retrieve a source IP address, based on the Element ID. The DF will make sure that it is the same as the source IP address in the IP packet header.

#### 8.9.4.2 Interface between AN and DF

AN and DF MUST negotiate a pair of IPSec Security Associations (inbound and outbound) using IKE with pre-shared keys. Certificates may be used in future versions of IPCablecom.

IKE will be running asynchronous to the event message generation and will guarantee that there is always a valid, non-expired pair of SAs. The corresponding IPSec keys MUST be unique to this particular AN and DF.

At the DF, AN Element IDs MUST somehow be associated with the corresponding IP addresses. One possibility is to associate each pre-shared key directly with the Element ID. IKE negotiations will use an ISAKMP identity payload of type ID_KEY_ID to identify the pre-shared key. The value in that identity payload will be the Element ID used in event messages.

Later, when an event message arrives at the DF, it will be able to query the database of IPSEC SAs and retrieve a source IP address, based on the Element ID. The DF will make sure that it is the same as the source IP address in the IP packet header.

#### 8.9.4.3 Interface between DF and DF

AN and DF MUST negotiate a shared secret (AN-DF Secret) using IKE with certificates. The IPCablecom profile for IKE with certificates is specified in clause 7.2. IKE will be running asynchronous to the event message generation. In

the case where an event message needs to be sent to a DF with which there is not a valid IPSec Security Association, the IPSec layer MUST automatically signal IKE to proceed with the key management exchanges and build a pair of IPSec SAs (inbound and outbound).

Not all interfaces between the same pair of DFs will require IPSec. For example, the call content interface does not run over IPSec. In order for the IPSec Security Associations to be established only for the DF-DF event message interface, each DF MUST allocate a set of UDP ports on which it will both send and receive DF-DF event messages. IPSec policy database for each DF MUST specify either an enumeration or a range of local UDP ports for which IPSec is enabled and which will be used exclusively for DF-DF event messages. If there are multiple calls that are simultaneously wiretapped and forwarded between the same pair of DFs (on different UDP ports) - they MUST all be protected with a single pair of IPSec Security Associations (inbound+outbound). Whenever a DF attempts to send on one of those UDP ports, it will either use an existing IPSec SA for a particular destination DF, or it will trigger IKE to establish a pair of SAs (inbound+outbound) for the specific target DF.

When the CMS tells a DF to forward event messages to another DF, it specifies the destination DF with an IP address. This means that the DF identity that needs authentication during an IKE exchange is the IP address. An IKE certificate for a DF contains the IP address of that DF. This IP address in the certificate MUST be used by IKE to validate the DF's IP address - to prevent IP address spoofing attacks.

After a pair of DF-DF Security Associations has been idle for some period of time, a DF MAY decide to remove it. In this case, the DF MUST send an ISAKMP Delete message to the other DF - to notify the other side of the SA deletion. Upon receiving a Delete message, the other DF MUST also remove that pair of Security Associations.

It will still be possible (with very small probability) that a DF uses an IPSec SA to send an event message to another DF; but when the event message arrives the target DF has already deleted the corresponding SA and has to drop the message. If there is still a problem after several timeouts and retries (e.g. ISAKMP Delete message was lost in transit), the sending DF MUST remove all of the corresponding IPSec SAs and re-run IKE to set up new Security Associations.

# 9 IPCablecom X.509 Certificate profile and management

## 9.1 Certificate Trust Hierarchy

There are two distinct certificate hierarchies used in IPCablecom. The first is the device or manufacturer's hierarchy. The second is the operator's hierarchy.

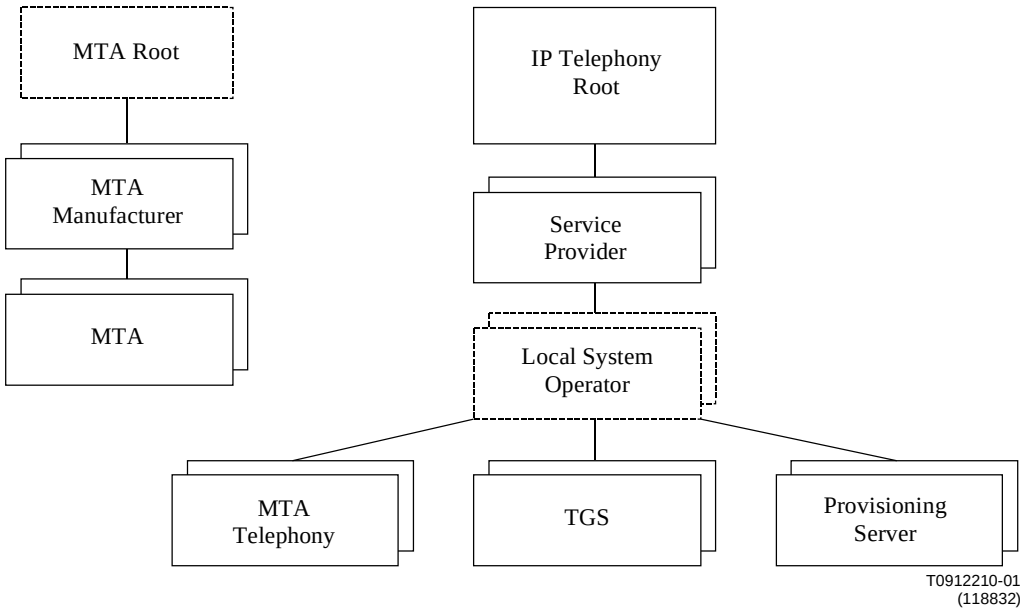



**Figure 17: IPCablecom Certificate Trust Hierarchy**

## 9.2 Device Certificate Hierarchy

The device certificate hierarchy is rooted at a (to be defined) issued IPCablecom root certificate. That certificate is used as the issuing certificate of a set of manufacturer's certificate. The manufacturer's certificates are used to sign the individual device certificates.

### 9.2.1 MTA Root Certificate

The root certificate has identical Subject and Issuer name forms of:

C=UN??,
O=ITU??,
OU=IPCablecom,
CN=IPCablecom Root Device Certificate Authority

The certificate is self-signed. Its lifetime shall exceed the lifetime of the MTA Manufacturer certificates that it issues.

This certificate MUST contain the following critical X.509 v.3 certificate extensions:

keyUsage with the keyCertSign bit (only) set

It MAY also contain the following critical X.509 v.3 certificate extensions:

basicConstraints with the cA=TRUE, and pathLenConstraint=0

Any other certificate extensions MAY also be included as non-critical. If the following certificate extensions are present, they are used as follows:

subjectKeyIdentifier as the 20 octet SHA-1 hash of the subjectPublicKey

The RSA modulus of the root public key MUST be 2 048 bits long.

### 9.2.2 MTA Manufacturer Certificate

A manufacturer's certificate is signed by the IPCablecom MTA Root Certificate. Its Subject name form is:

C=<country>,
O=<CompanyName>,
[S=<state/province>],
[L=<city>],
OU=IPCablecom,
[OU=<Manufacturer's Facility>],
CN=<CompanyName> IPCablecom CA

In the above, state/province, city and manufacturer's facility are optional attributes. A manufacturer may have more than one manufacturer's certificate - in general there will be one or more certificates per manufacturer.

MTA Manufacturer certificate's validity period SHOULD exceed the validity period of the MTA device certificates that it issues.

This certificate MUST contain the following critical X.509 v.3 certificate extensions:

keyUsage with the keyCertSign bit (only) set

It MAY also contain the following critical X.509 v.3 certificate extensions:

basicConstraints with the cA=TRUE and the pathLenConstraint=0

Any other certificate extensions MAY also be included as non-critical. If the following certificate extensions are present, they are used as follows:

subjectKeyIdentifier as the 20 octet SHA-1 hash of the SubjectPublicKey
authorityKeyIdentifier with only the keyIdentifier field set (from the subjectKeyIdentifier of the issuer)

The RSA modulus of the manufacturer public key MUST be 1 024, 1 536 or 2 048 bits long.

### 9.2.3 MTA Manufacturer Code Verification Certificate

Code Verification Certificate (CVC) Specification for embedded MTAs is identical to the J.112.

### 9.2.4 MTA Device Certificate

An MTA device certificate is always signed by a manufacturer's certificate. Its Subject name form is:

C=<country>,
O=<Company Name>,
[S=<state/province>],
[L=<city>],
OU=IPCablecom,
OU=<Product Name>,
[OU=<Manufacturer's Facility>],
CN=<MAC Address>,
CN=<MTA Hardware Version>

In the above, state/province, city and manufacturer's facility are optional attributes.

The MAC address is expressed as six pairs of hexadecimal digits separated by colons, e.g."00:60:21:A5:0A:23". The Alpha HEX characters (A-F) MUST be expressed as uppercase letters.

The characters employed in the representation of MTA Hardware Version MUST be restricted to:

- A-Z (0x41-0x5A)
- a-z (0x61-0x7A)
- 0-9 (0x30-0x39)
- (space) (0x20), "-" (0x2D), "." (0x2E)

This MUST be the same MTA Hardware Version as the one reported via SNMP, inside the MIB variable **pktcMtaDevHardwareVersion**.

An MTA device certificate will not be renewable and thus must have a validity period greater than the operational lifetime of the MTA. This validity period SHOULD extend at least 20 years after the manufacture date.

Any X.509 v.3 certificate extensions MAY be included as non-critical. If the following certificate extensions are present, they are used as follows:

keyUsage with the digitalSignature, keyEncipherment, and keyAgreement bits setauthorityKeyIdentifier with only the keyIdentifier field set (from the subjectKeyIdentifier of the issuer)

The RSA modulus of the MTA public key MUST be 1 024 bits long.

## 9.3 Operator Certificate Hierarchy

The general theory of operation is that, as part of the customer enrollment process, either the Local System CA or a central Service Provider CA issues a system-specific certificate to the MTA. The public key bound into this certificate is the same key that is bound into the device certificate. This certificate is rooted at a global System Operator Root, which issues a certificate for each System Operator.

Each MTA MUST be manufactured with a copy of the System Operator Root public key.

### 9.3.1 IP Telephony Root Certificate

The root certificate has identical Subject and Issuer name forms of:

C=UN??,
O=ITU??,
OU=IPCablecom,
CN=IPCablecom Root IP Telephony Certificate Authority

The certificate is self-signed. Its lifetime shall exceed the lifetime of the Telephony Service Provider certificates that it issues.

This certificate MUST contain the following critical X.509 v.3 certificate extensions:

keyUsage with the keyCertSign bit (only) set

It MAY also contain the following critical X.509 v.3 certificate extensions:

basicConstraints with the cA=TRUE, and pathLenConstraint=0

Any other certificate extensions MAY also be included as non-critical. If the following certificate extensions are present, they are used as follows:

subjectKeyIdentifier as the 20 octet SHA-1 hash of the subjectPublicKey

The RSA modulus of the root public key MUST be 2 048 bits long.

### 9.3.2 Telephony Service Provider Certificate

This is the certificate held by the overall system operator, signed by the IP Telephony Root CA. This certificate has a name form:

C=<country>,
O=<Company>,
OU=IPCablecom,
CN=<Company> IPCablecom System Operator CA

The lifetime of this certificate shall exceed the lifetime of all of the certificates that it issues. Generally, this lifetime is at least ten years.

This certificate MUST contain the following critical X.509 v.3 certificate extensions:

keyUsage with the keyCertSign and keyCrlSign bits (only) set

It MAY also contain the following critical X.509 v.3 certificate extensions:

basicConstraints with the cA=TRUE and the pathLenConstraint=0

Any other certificate extensions MAY also be included as non-critical. If the following certificate extensions are present, they are used as follows:

subjectKeyIdentifier as the 20 octet SHA-1 hash of the SubjectPublicKey
authorityKeyIdentifier with only the keyIdentifier field set (from the subjectKeyIdentifier of the issuer)

The RSA modulus of the System Operator public key MUST be 2 048 bits long.

### 9.3.3 Local System Certificate

This is the certificate held by the local system. The private key for this certificate is held by the local provisioning server and is used to issue network-specific MTA certificates. The existence of this certificate is optional, as the System Operator CA may be used to directly sign all MTA and network server certificates.

It is signed by the operator certificate and has the name form of:

C=<Country>,
O=<Company>,
OU=IPCablecom,
OU=<Local System Name>,
CN=<Company> IPCablecom Local System CA

Its lifetime shall exceed the lifetime of all of the certificates that it issues. It is generally issued with a validity period of not less than a year and not more than five years.

This certificate MUST contain the following critical X.509 v.3 certificate extensions:

keyUsage with the keyCertSign and keyCrlSign bits (only) set

It MAY also contain the following critical X.509 v.3 certificate extensions:

basicConstraints with the cA=TRUE and the pathLenConstraint=0

Any other certificate extensions MAY also be included as non-critical. If the following certificate extensions are present, they are used as follows:

subjectKeyIdentifier as the 20 octet SHA-1 hash of the SubjectPublicKey
authorityKeyIdentifier with only the keyIdentifier field set (from the subjectKeyIdentifier of the issuer)

The RSA modulus of the Local System public key MUST be 1 024, 1 536 or 2 048 bits long.

### 9.3.4 MTA Telephony Certificate

This is the certificate issued to each MTA for service in a specific network. It authenticates the call signalling identity of that MTA - its FQDN (see note 1). The public key for this certificate is the same as that embedded in the device certificate. This certificate is signed by the either the Local System CA or the System Operator CA. It has a name form of:

C=<Country>,
O=<Company>,
OU=IPCablecom,
[OU=<Local System Name>],
CN=<Subscriber ID>, where the Subscriber ID is a DNS name (see note 2).

NOTE 1: Call Signalling identity also includes the port number, which is not in the MTA Telephony certificate. It was determined that it is not worthwhile to cryptographically authenticate the port number. In order for the port number to be forged, an adversary has to hack the MTA or somehow modify the message stream. If in addition to illegally using a false port number, an adversary also has to use a certificate for a wrong port, that is not a serious obstacle to this attack.

NOTE 2: The X.500 CN (CommonName) attribute is of ASN.1 type PrintableString, which restricts it to a subset of all printable characters. This restriction can be met by using the preferred name syntax for DNS names, as it is defined in RFC 1035.

Although Local System Name is optional, it is REQUIRED when the Local System CA signs this certificate.

This certificate is generally issued with a validity period of not less than 45 days and for no more than one year. The validity period generally coincides with the billing cycle. The certificate is automatically re-issued by the provisioning - see clause 8.1.1.2.9.

Any X.509 v.3 certificate extensions MAY be included as non-critical. If the following certificate extensions are present, they are used as follows:

keyUsage with the digitalSignature, keyEncipherment, and keyAgreement bits set
authorityKeyIdentifier with only the keyIdentifier field set (from the subjectKeyIdentifier of the issuer)

The public key is identical to the MTA public key with the RSA modulus that MUST be 1 024 bits long.

## 9.3.5 Operational Ancillary Certificates

All of these are signed by the either the Local System CA or by the Service Provider CA.

### 9.3.5.1 Ticket Granting Server

This is the certificate used by the Kerberos Ticket Granting Server (TGS). This certificate has a name form of:

C=<Country>,
O=<Company>,
OU=IPCablecom,
OU=[<Local System Name>],
CN= IPCablecom Ticket Granting Server,
CN=<DNS Name>

Although Local System Name is optional, it is REQUIRED when the Local System CA signs this certificate.

This certificate is generally issued with a validity period of not less than a year and not more than five years.

Any X.509 v.3 certificate extensions MAY be included as non-critical. If the following certificate extensions are present, they are used as follows:

keyUsage with the digitalSignature (only) bit set
authorityKeyIdentifier with only the keyIdentifier field set (from the subjectKeyIdentifier of the issuer)

In addition, the PKINIT Specification requires the TGS certificate to include the **subjectAltName** v.3 certificate extension, the value of which must be the Kerberos principal name of the TGS (TGS is referred to as the KDC in the PKINIT spec). Refer to draft RFC: "The Kerberos Network Authentication Service (V5)", for the exact syntax of **subjectAltName**.

The RSA modulus of the TGS public key MUST be 1 024 bits long.

### 9.3.5.2 Provisioning Server

This certificate is used to sign the configuration file sent to the MTA. This certificate has a name form of:

C=<Country>,
O=<Company>,
OU=IPCablecom,
OU=[<Local System Name>],
CN=IPCablecom Provisioning Server
CN=<DNS Name>

Although Local System Name is optional, it is REQUIRED when the Local System CA signs this certificate.

This certificate is generally issued with a validity period of not less than a year and not more than five years.

Any X.509 v.3 certificate extensions MAY be included as non-critical. If the following certificate extensions are present, they are used as follows:

keyUsage with the digitalSignature (only) bit set
authorityKeyIdentifier with only the keyIdentifier field set (from the subjectKeyIdentifier of the issuer)

In addition, the CMS certificate MUST include the **subjectAltName** v.3 certificate extension, the value of which must be the Kerberos principal name of the CMS. Refer to draft RFC: "The Kerberos Network Authentication Service (V5)", for the exact syntax of encoding the Kerberos principal names inside **subjectAltName.**

The RSA modulus of the Provisioning Server public key MUST be 1 024 bits long.

## 9.3.6 Operator Code Verification Certificate

Code Verification Certificate (CVC) Specification for embedded MTAs is identical to the J.112.

## 9.4 Certificate Revocation

Out of scope for this current Specification of IPCablecom security.

# 10 Cryptographic Algorithms

This clause describes the cryptographic algorithms used in the IPCablecom security Specification. When a particular algorithm is used, the algorithm MUST follow the corresponding Specification.

## 10.1 DES

The Data Encryption Standard (DES) is specified in PKCS#9.

### 10.1.1 XDESX

An option for the encryption of RTP packets is DESX-XEX. XDESX, or DESX, has been proven as a viable method for overcoming the weaknesses in DES while not greatly adding to the implementation complexity. The strength of DESX against key search attacks is presented in RFC 1750. The CBC mode of DESX-XEX is shown in figure 18, where DESX-XEX is executed within the block called "block cipher." Inside the block, DESX-XEX is performed as shown in figure 20 using a 192-bit key. K1 is the first 8-bytes of the key, and K2 represents the second 8-bytes of key, and K3 the third 8-bytes of key.

### 10.1.2 DES-CBC-PAD

This variant of DES is also based on the analysis of DESX presented in RFC 1750. When using DESX in CBC mode, an optimized architecture is possible. It can be described in terms of the DES-CBC configuration plus the application of a random pad on the final DES-CBC output blocks. This configuration uses 128 bits of keying material, where 64 bits are applied to the DES block according to PKCS#9, and an additional 64 bits of keying material is applied as the random pad on the final DES-CBC output blocks.

In this case, the same IV used to initialize the CBC mode is used as keying material for the random pad. Each block of DES-CBC encrypted output is XOR-ed with the 64-bit Initialization Vector that was used to start the CBC operation. If a short block results from using Residual Block Termination (see clause 6.1.5), the left-most-bits of the IV are used in the final XOR padding operation. This mode of DES-CBC is shown in figure 19, where DES is executed in the block called "block cipher." A 64-bit key value is used.

### 10.1.3 3DES-EDE

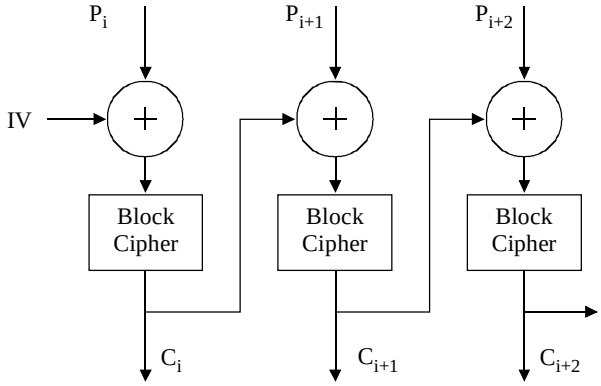
Another option for the encryption of RTP packets for IPCablecom, is 3DES-EDE-CBC. The CBC mode of 3DES-EDE is shown in figure 18, where 3DES-EDE is executed within the block called "block cipher." Inside the block, 3DES-EDE is performed as shown in figure 21 using a 128-bit key. K1 is the first 8 bytes of the key, and K2 represents the second 8 bytes of key; and K3=K1.

### 10.1.4 Block Termination

If block ciphers are supported, a short block (p-bits < 64-bits) MUST be terminated by residual block termination as shown in figure 22. Residual block termination (RBT) is executed as follows:

Given a final block having n bits, where n is less than 64, the n bits are padded up to a block of 64 bits by appending (64 – n) bites of arbitrary value to the right of the n-bits. The resulting block is DES encrypted using CFB64 mode, with the next-to-last ciphertext block serving as the initialization vector for the CFB64 operation (see FIPS-81 for a description of the CFB64 mode of DES). The leftmost n bits of the resulting ciphertext are used as the short cipher block. In the special case where the complete payload is less than 64 bits, the procedure is the same as for a short final block, with the provided initialization vector serving as the initialization vector for the DES-FB64 operation. Residual block termination is illustrated in figure 22 for both encryption and decryption operations.

CBC Encryption Architecture




CBC Decryption Architecture

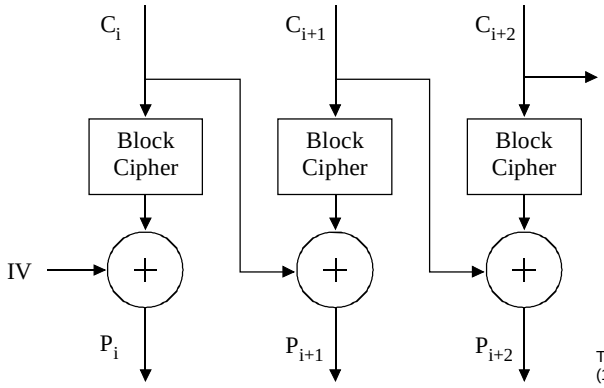
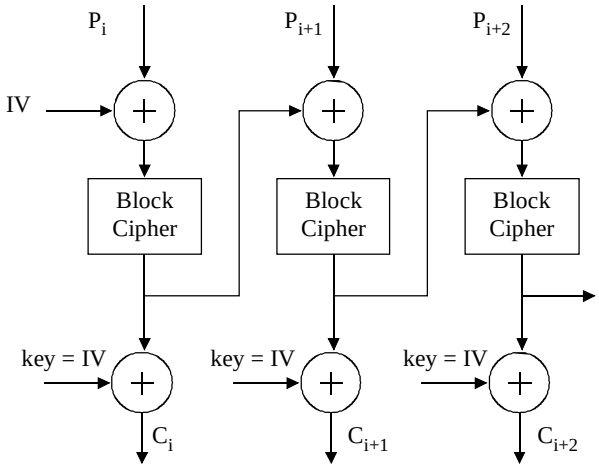



**Figure 18: CBC Mode**

CBC-PAD Encryption Architecture




CBC-PAD Decryption Architecture

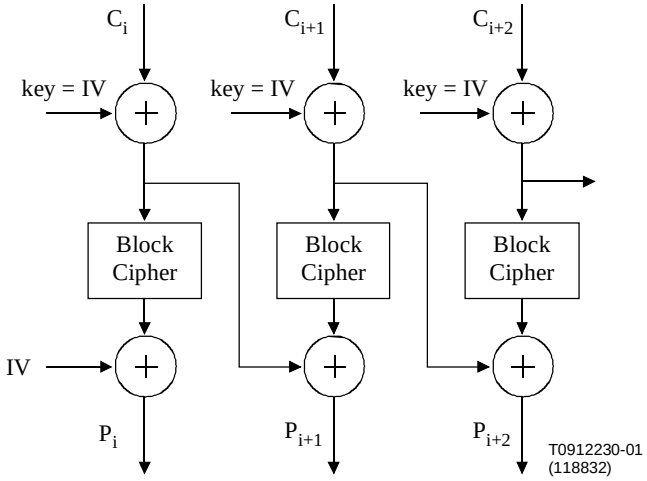



**Figure 19: CBC PAD Mode**

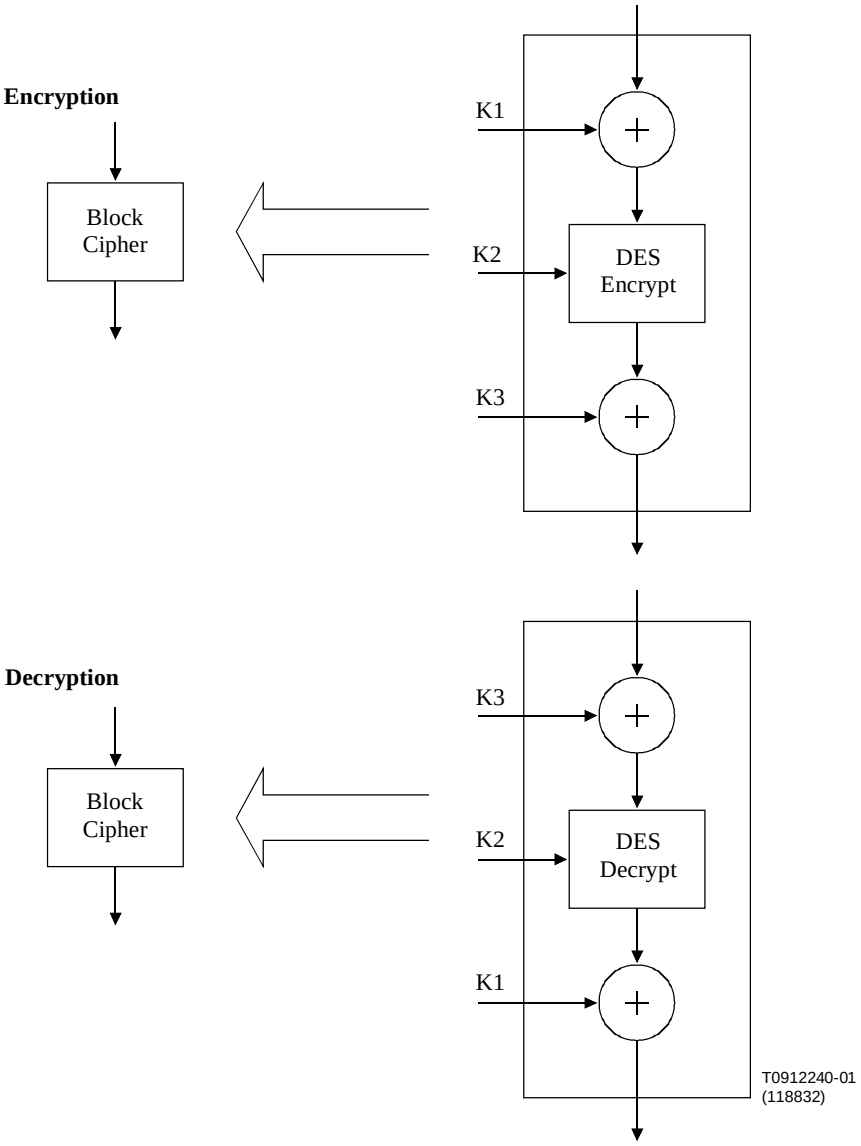



**Figure 20: DESX-XEX as Block Cipher**

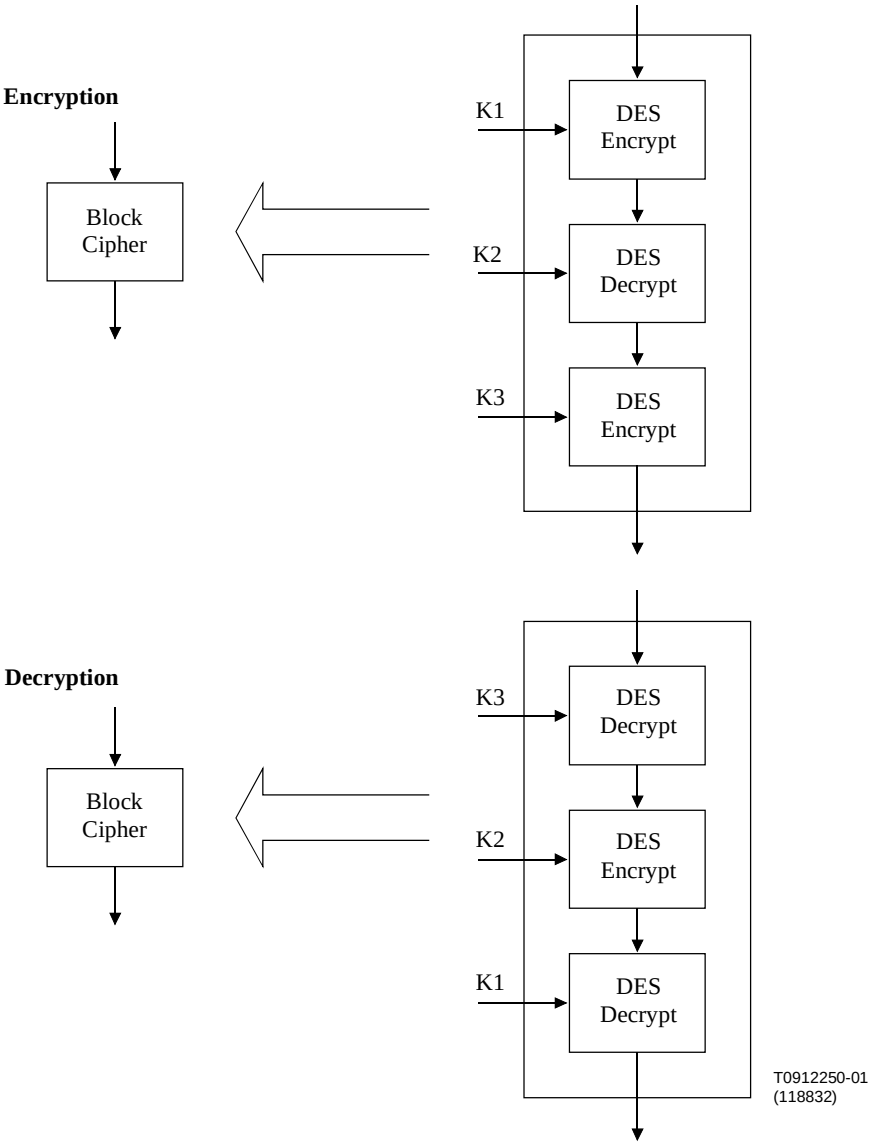



**Figure 21: 3DES-EDE as Block Cipher**

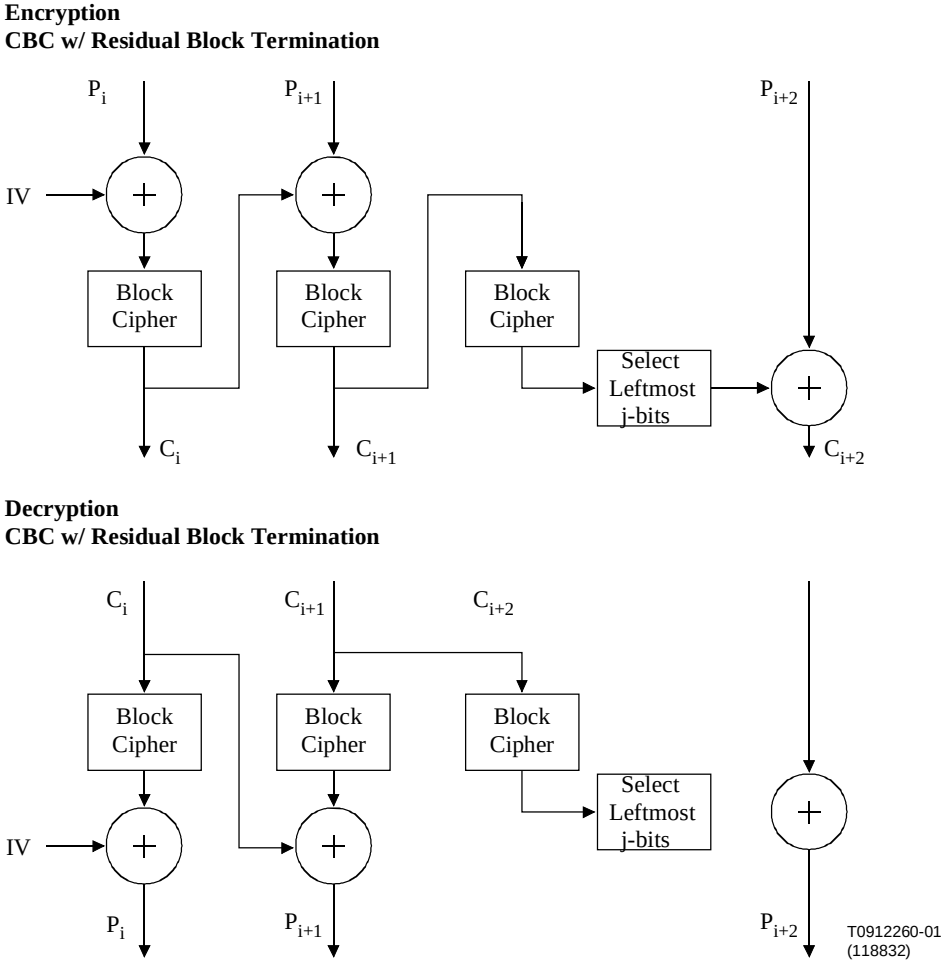



**Figure 22: CBC with Residual Block Termination**

## 10.2 RC4

RC4 is a very efficient symmetric cipher. RC4 is used in clause 8.8 to encrypt media flows. Key management is described in clause 8.8.4. The algorithm uses variable length keys. For IPCablecom the key length is set to 128 bits. The generation of this 128-bit key from the session key is described in clause 10.6.

RC4 is a pseudo-random number generator in output feedback mode. A stream is generated from the key and xored with the plaintext. There is no integrity protections on the data. RC4 uses a 256 entry substitution box (Sbox), which must be initialized. The entries in the Sbox are represented as bytes $S_0$, $S_1$, …, $S_{255}$. To initialize the Sbox, it is first filled with each entry matching its index. So,

$S_0 = 0$, $S_1 = 1$, … $S_{255} = 255$.

Another 256-byte array is filled with the key, repeating as necessary to fill the array, $K_0$, $K_1$, … , $K_{255}$. An index, j, is set to 0. Then, the Sbox is filled as follows:

for i = 0 to 255

j = (j + Si + Ki) mod 256

swap Si and Sj.

After the loop completes, the Sbox is initialized. The Sbox is dependent on the key, so a new one must be initialized for each key. The Sbox can now be used to generate a pseudo-random stream. i and j are initialized to 0, and a random byte is produced as follows:

i = (i + 1) mod 256

$j = (j + S_i) \bmod 256$

swap $S_i$ and $S_j$

$t = (S_i + S_j) \bmod 256$

random_byte = $S_t$.

To generate more bytes, the process is repeated using the values for i and j that result from the previous iteration.

## 10.3 RSA Signature

All public key signatures for IPCablecom are generated and verified using the RSA signature algorithm PKCS#1v1.5. The format for all IPCablecom signatures is the RSA Cryptographic Message Syntax (CMS).

See clause 8.2.1.2.1 for the CMS format Specification of the signed SNMP messages sent during device provisioning. See clause 8.2.1.2.2 for the CMS format Specification of the TFTP-Get performed during device provisioning.

## 10.4 RSA Encryption

The RSA encryption algorithm used in IPCablecom is specified in PKCS#1v2.

PKCS#1v2 encryption is used in the EnvelopedData format as defined by CMS. The TFTP-Get of clause 8.2.1.2 uses the CMS EnvelopedData format. In this cases the PKCS#1v2 encryption is used to encrypt a 3-key 3-DES key.

## 10.5 HMAC-SHA-1

The keyed hash employed by the HMAC-Digest Attribute MUST use the HMAC message authentication method (RFC 2104) with the SHA-1 hash algorithm FIPS 180-1.

HMAC-SHA-1 is used to authenticate AN-AN UDP messages for DQoS, as described in clause 8.2.1.2.2. The key is 160 bits long. The key management for the HMAC keys is described in clause 8.2.1.4.1.

## 10.6 Key Derivation

Key derivation clauses in the present document refer to a function F(S, seed), where S is a shared secret from which keying material is derived, and seed is a constant string of bytes. Below is the Specification of F(S, seed), borrowed from TLS (RFC 2246).

F(S, seed) = HMAC_SHA-1(S, A(1) + seed) +

HMAC_SHA-1(S, A(2) + seed) +

HMAC_SHA-1(S, A(3) + seed) + …

where + indicates concatenation.

A() is defined as: A(0) = seed

A(i) = HMAC_SHA-1(S, A(i-1))

F(S, seed) is iterated as many times as is necessary to produce required quantity of data. Unused bytes at the end of the last iteration will be discarded.

## 10.7 The MMH-MAC

In this clause the MMH Function and the MMH Message Authentication Code (MAC) are described. The MMH-MAC is the message authentication code option for the media flows. As discussed in clause 8.7.2, the MMH-MAC is computed over the RTP header and the payload is generated by the codec. The MMH Function will be described next, followed by a description of the MMH-MAC.

## 10.7.1 The MMH Function

The Multilinear Modular Hash (MMH) Function described below is a variant of the MMH Function described in the Halevi and Krawczyk MMH reference. Some of the computations described below use signed arithmetic whereas the computations in the Halevi and Krawczyk MMH reference use unsigned arithmetic. The signed arithmetic variant described here was selected for its computational efficiency when implemented on DSPs. All of the properties shown for the MMH function in the Halevi and Krawczyk MMH reference continue to hold for the signed variant.

The MMH Function has three parameters: the word size, the number of words of input, and the number of words of output. MMH[□, *s*, *t*] specifies the hash function with word size □, *s* input words and *t* output words. For IPCablecom the word size is fixed to 16 bits: □ = 16. The number of output words will be either 1 or 2: $t \in \{1, 2\}$. The MMH Hash Function will first be described for $t = 1$, i.e. one output word.

### 10.7.1.1 MMH[16, s, 1]

For the remainder of this clause 10.7, MMH[16, *s*, 1] is denoted by *H*. In addition to *s* words of input, *H* also takes as input a key of *s* words. When *H* is used in computing the MMH-MAC, the key is randomly generated and remains fixed for several inputs as described in clause 10.7.2. The key is denoted by *k* and the *i*th word of the key by $k_i$: $k = k_1, k_2, \ldots, k_s$. Likewise the input message is denoted by *m* and the *i*th word of the input message by $m_i$: $m = m_1, m_2, \ldots, m_s$.

To describe *H*, the following definitions are needed. For any even positive integer *n*, $S_n$ is defined to be the following set of *n* integers: $\{-n/2, \ldots, 0, \ldots, (n/2)-1\}$. For example, $S_{2^{16}} = \{-2^{15}, \ldots, 0, \ldots, 2^{15}-1\}$ is the set of signed 16-bit integers. For any integer *z*, *z* smod *n* is the unique element □ of $S_n$ such that $z \equiv$ □ (mod *n*). For example, if *z* is a 32-bit signed integer in 32-bit twos complement representation, then *z* smod $2^{16}$ can be computed by taking the 16 least significant bits of *z* and interpreting those bits in 16-bit twos complement representation.

For any positive integer *q*, $Z_q$ denotes the following set of *q* integers: $\{0, 1, \ldots, q-1\}$.

As described above *H* takes as input a key of *s* words. Each of the *s* words is interpreted as a 16-bit signed integer, i.e. an element of $S_{2^{16}}$. *H* also takes as input a message of *s* words. Each of the *s* words is interpreted as a 16-bit signed integer, i.e. an element of $S_{2^{16}}$. The output of *H* is an unsigned 16-bit integer, i.e. an element of $Z_{2^{16}}$. Alternatively, the range of H is $S_{2^{16}}^s \times S_{2^{16}}^s$ and the domain is $Z_{2^{16}}$.

*H* is defined by a series of steps. For $k,m \in S_{2^{16}}^s$,

1) Define $H_1$ as $H_1(k, m) = \sum_{i=1}^{s} k_i \cdot m_i \; s\bmod 2^{32}$.
2) Define $H_2$ as $H_2(k, m) = H_1(k, m) \bmod p$ where *p* is the prime number $p = 2^{16}+1$.
3) Define *H* as $H(k, m) = H_2(k, m) \bmod 2^{16}$.

Equivalently,

$$H(k,m) = \left( \left( \left( \sum_{i=1}^{s} k_i \cdot m_i \right) s\bmod 2^{32} \right) \bmod p \right) \bmod 2^{16}$$

Each step is discussed in detail below.

**Step 1**. $H_1(k, m)$ is the inner product of two vectors each of *s* 16-bit signed integers. The result of the inner product is taken smod $2^{32}$ to yield an element of $S_{2^{32}}$ (see note). That is, if the inner product is in twos complement representation of 32 or more bits, the 32 least significant bits are retained and the resulting integer is interpreted in 32-bit twos complement representation.

NOTE: The entire sum need not be computed before performing the smod $2^{32}$ operation. The smod $2^{32}$ operation can be computed on partial sums since (x + y) smod $2^{32}$ = (*x* smod $2^{32}$ + *y* smod $2^{32}$) smod $2^{32}$.

**Step 2**. This step consists of taking an element $x$ of $S_{2^{32}}$ and reducing it mod $p$ to yield an element of $Z_p$. If $x$ is represented in 32-bit twos complement notation then this reduction can be accomplished very simply as follows. Let $a$ be the unsigned integer given by the 16 most significant bits of $x$. Let $b$ be the unsigned integer given by the 16 least significant bits of $x$. There are two cases depending upon whether $x$ is negative.

**Case 1**. If $x$ is non-negative then $x = a2^{16} + b$ where $a \in \{0, \ldots, 2^{15}-1\}$ and b $\in$ $\{0, \ldots, 2^{16}\text{-}1\}$. From the modular equation:

$$a2^{16} + b \equiv a2^{16} + b - a(2^{16} + 1) \pmod{(2^{16} + 1)}$$

it follows that $x \equiv b - a \pmod{p}$. The quantity $b - a$ is in the range $\{-2^{15} + 1, \ldots, 2^{16}-1\}$. Therefore if $b - a$ is non-negative then $x \bmod p = b - a$. If $b - a$ is negative then $x \bmod p = b - a + p$.

**Case 2**. If $x$ is negative then $x = a2^{16} + b - 2^{32}$ where $a \in \{2^{15}, \ldots, 2^{16}-1\}$and $b \in \{0, \ldots, 2^{16} - 1\}$. From the modular equation:

$$a2^{16} + b - 2^{32} \equiv b + a2^{16} - a(2^{16} + 1) - 2^{32} + 2^{16}(2^{16} + 1) \pmod{(2^{16} + 1)}$$

it follows that $x \equiv b - a + 2^{16} \pmod{p}$. The quantity $b - a + 2^{16}$ is in the range $\{2^{15} + 1, \ldots, 2^{17} - 1\}$. Therefore, if $b - a < p$ then $x \bmod p = b - a$. If $b - a \geq p$ then $x \bmod p = b - a - p$.

**Step 3**. This step takes an element of $Z_p$ and reduces it mod $2^{16}$. This is equivalent to taking the 16 least significant bits.

### 10.7.1.2 MMH[16, s, 2]

This clause describes the MMH Function with an output length of two words which in this case is 32 bits. For convenience, let $H'$ = MMH[16, $s$, 2]. $H'$ takes a key of $s + 1$ words. Let $k = k_1, \ldots, k_{s+1}$. Furthermore, define $k^{(1)}$ to be the $s$ words of $k$ starting with $k_1$, i.e. $k^{(1)} = k_1, \ldots, k_s$. Define $k^{(2)}$ to be the $s$ words of $k$, starting with $k_2$, i.e. $k^{(2)} = k_2, \ldots, k_{s+1}$. For any $k \in S_{2^{16}}^{s+1}$ and any $m \in S_{2^{16}}^{s}$ $m$, $H'(k, m)$ is computed by first computing $H(k^{(1)}, m)$ and then H($k^{(2)}$, $m$) and concatenating the results. That is, $H'(k, m) = H(k^{(1)}, m) \circ H(k^{(2)}, m)$.

## 10.7.2 The MMH-MAC

This clause describes the MMH-MAC. The MMH-MAC has three parameters; the word size, the number of words of input, and the number of words of output. MMH-MAC[ω, $s$, $t$] specifies the message authentication code with word size ω, $s$ input words and $t$ output words. For IPCablecom the wordsize is fixed to 16 bits: ω = 16. The number of output words will be either 1 or 2: $t \in \{1, 2\}$.

For convenience, let $M$ = MMH-MAC[16, $s$, $t$]. When using $M$, a sender and receiver share a key $k$ of $s + t - 1$ words. In addition, they share a sequence of key streams of $t$ words each, one one-time pad for each message sent. Let $r^{(i)}$ be the key stream used for the $i$th message sent and received. For the $i$th message, $m^{(i)}$, the message authentication code is computed as:

$$M(k, r^{(i)}, m^{(i)}) = \mathrm{H}(k, m^{(i)}) + r^{(i)}.$$

Here $H$ = MMH[16, $s$, $t$], $r^{(i)}$ is in $Z_{2^{16}}$ and addition is mod $2^{16}$.

### 10.7.2.1 MMH-MAC When Using RC-4

When calculating the MMH-MAC when using RC4, the sequence of key streams is generated by an RC4 key stream as described in clause 8.7.2. The $2(s + t - 1)$-byte key for MMH-MAC[16, $s$, $t$] are randomly generated as described in clause 8.7.2 from a session key for the media flows which is generated by the key agreement protocol give in clause 8.7.3.

### 10.7.2.2 MMH-MAC When Using a Block Cipher

When calculating the MMH-MAC when encryption is performed by one of the available block ciphers, the block cipher is used to calculate the $t$ words of $r(i)$ key stream (pad) as defined in clause 8.7.2.3.8.

#### 10.7.2.3 Odd Payload Sizes

If a message *m* is not of length *s* words, but rather of length $v < s$ words, then the input to *M* is a new message *m'* given by $m' = m_1, \ldots m_v, e_{v+1}, \ldots e_s$ where $e_{v+1} = \ldots = e_s$ is the all zeroes word.

## 10.8 Random Number Generation

Good random number generation is vital to most cryptographic mechanisms. Implementations SHOULD do their best to produce true-random seeds; they should also use cryptographically strong pseudo-random number generation algorithms. RFC 1750 gives some suggestions; other possibilities include use of a per-MTA secret installed at manufacture time and used in the random number generation process.

# 11 Physical security

## 11.1 Protection for MTA Key Storage

The IPCablecom security Specification requires that an embedded MTA (MTA-E) and a standalone MTA (MTA-S) maintain persistent IPSEC encryption and authentication keys and Kerberos session keys. An MTA MUST also maintain in permanent write-once memory an RSA key pair. An MTA SHOULD deter unauthorized physical access to this keying material.

The level of physical protection of keying material required by the IPCablecom security Specification for an MTA is specified in terms of the security levels defined in the FIPS PUBS 140-1, Security Requirements for Cryptographic Modules, standard. An MTA-E or MTA-S SHOULD, at a minimum meet FIPS PUBS 140-1 Security Level 1 requirements.

The IPCablecom Security Specification's minimal physical security requirements for an MTA will not, in normal practice, jeopardize a customer's data privacy. Assuming the subscriber controls the access to the MTA with the same diligence they would protect a cellular phone, physical attacks on that MTA to extract keying data are likely to be detected by the subscriber.

An MTA's weak physical security requirements, however, could undermine the cryptographic protocol's ability to meet its main security objective: to provide a service operator with strong protection from theft of high value network.

The IPCablecom Security Specification requirements protect against unauthorized access to these network services by enforcing an end-to-end message integrity and encryption of signalling flows across the network and by employing an authenticated key management protocol. If an attacker is able to legitimately subscribe to a set of services and also gain physical access to an MTA containing keying material, then in the absence of strong physical protection of this information, the attacker can extract keying material from the MTA. And redistribute the keys to other users running modified illegitimate MTAs, effectively allowing theft of network services.

There are two distinct variations of "active attacks" involving the extraction and redistribution of cryptographic keys. These include the following:

1) An "RSA active clone" would actively participate in IPCablecom key exchanges. An attacker must have some means by which to remove the cryptographic keys that enable services, from the clone master, and install these keys into a clone MTA. An active clone would work in conjunction with an active clone master to passively obtain the clone master's keying material and then actively impersonate the clone master. A single active clone may have numerous active clone master identities from which to select to obtain access to network services. This attack allows, for example, the theft of non-local voice communications.

2) A DH active clone would also actively participate in the IPCablecom key exchanges and like the RSA active clone, would require an attacker to extract the cryptographic keys that enable the service from the clone master and install these keys into a clone MTA. However, unlike the RSA active clone, the DH active clone must obtain the clone masters random number through alternate means or perform the key exchange and risk detection. Like an RSA active clone, an DH active clone may have numerous clone master identities from which to select to obtain access to the network services.

3) An "active black box" MTA, holding another MTA's session or IPSEC keys, would use the keys to obtain access to network-based services or traffic flows similar to the RSA active clone. Since both session keys and IPSEC keys change frequently, such clones have to be periodically updated with the new keying material, using some out-of-band means.

An active RSA clone, for example, could operate on a cable access network within whatever geographic region the cloned parent MTA was authorized to operate in. Depending upon the degree to which a service operator's subscriber authorization system restricted the location from which the MTA could operate, the clone's scope of operation could extend well beyond a single J.112 MAC domain.

An active clone attack may be detectable by implementing the appropriate network controls in the system infrastructure. Depending on the access fraud detection methods that are in place, a service operator has a good probability of detecting a clone's operation should it attempt to operate within

the network. The service operator could then take defensive measures against the detected clone. For example, in the case of an active RSA clone, it could block the device's future network access by including the device certificate on the certificate hot list. Also the service operator's subscriber authorization system could limit the geographic region over which a subscriber, identified by its cryptographic credentials, could operate. Additionally the edge router functionality in the AN could limit any access based upon IP address. These methods would limit the region over which an active RSA clone could operate and reduce the financial incentive for such an attack.

The architectural guidelines for IPCablecom security are determined by balancing the revenues that could be lost due to the classes of active attacks against the cost of the methods to prevent the attack. At the extreme side of preventive methods available to thwart attacks, both physical security equivalent to FIPS PUB 140-1 Level 3 and network based fraud detection methods could be used to limit the access fraud that allows theft of network based services. The network based intrusion detection of active attacks allows operators to consider operational defenses as an alternative to increased physical security. If the revenues threatened by the active attacks increase significantly to the point where additional protective mechanisms are necessary, the long term costs of operational defenses would need to be compared with the costs of migrating to MTAs with stronger physical security. The inclusion of physical security should be an implementation and product differentiation specific decision.

Although the scope of the current IPCablecom Specifications do not specifically define requirements for MTAs to support any requirements other than voice communications, the goal of the IPCablecom effort is to provide for the eventual inclusion of integrated services. Part of these integrated services may include the "multicast" of high value content or extremely secure multicast corporate videoconference sessions.

Two additional attacks enabling a compromise of these types of services are defined:

1) An "RSA passive clone" passively monitors the parent MTA's key exchanges and, having a copy of the parent MTA's RSA private key, is able to obtain the same traffic keying material the parent MTA has access to. The clone then uses the keying material to decrypt downstream traffic flows it receives across the shared medium. This attack is limited in that it only allows snooping, but if the traffic were of high value, the attack could facilitate the theft of high value multicast traffic.

2) A "Passive black box" MTA, holding another MTA's short term (relative to the RSA key) keys, uses the keying material to gain access to encrypted traffic flows similar to the RSA passive clone.

The passive attacks, unlike the active attacks, are not detectable using network based intrusion detection techniques since these units never make themselves known to the network while performing the attack. However, this type of service theft has unlimited scale since the passive clones and black boxes, even though they operate on different cable access networks (sometimes referred to as the same J.112 MAC domain) as the parent MTA from whom the keys were extracted, gain access to the protected data the parent MTA is currently receiving since the encryption of the data most likely occurred at the source. (These are general IP multicast services, not to be confused with the specific J.112 multicast implementation, where passive clones would be restricted to a single downstream AN segment.) The snooping of the point-to-point data is limited to the CM MAC domain of the parent MTA. Passive attacks may be prevented by ensuring that the cryptographic keys that are used to enable the services cannot be tampered with in any manner.

In setting goals and guidelines for the IPCablecom security architecture, an assessment has to be made of the value of the services and content that can be stolen or monitored by key extraction and redistribution to passive MTAs. The cost of the solution should not be greater that the lost revenue due to theft of the service or subscribers terminating the service due to lack of privacy. However at this time, there is no clear cost that can be attributed to either the lost revenue from high value multicast services or the loss of subscribers due to privacy issues unique to this type of network. Therefore, it was concluded that passive key extraction and redistribution attacks would pose an indeterminate financial risk to service operators; and that the cost of protection (i.e. incorporation of stronger physical security into the MTA) should be balanced against the value of the risk. As with the active attacks, the decision to include additional functionality to implement physical security in the MTA should be left as an implementation and product differentiation issue and not be mandated as a requirement of the IPCablecom security Specification.

## 11.2 MTA Key Encapsulation

As stated in the previous clause, FIPS PUB 140-1 Security Level 1 specifies very little actual physical security and that an MTA MUST deter unauthorized "physical" access to its keying material. This restricted access also includes any ability to directly read the keying material using any of the MTA interfaces.

Two of the (many) requirements of FIPS PUB 140-1 Security Level 3 recommends "data ports for critical security parameters be physically separated from other data ports" and, entry/exit of keys in encrypted form or direct entry/exit with split knowledge procedures". As also mentioned in the previous clause, the IPCablecom security Specification is not requiring compliance with any of the FIPS PUB 140-1 Security Level 3 requirements.

However, it is strongly recommended that any persistent keying material SHOULD be encapsulated such that there is no way to extract the keying material from the MTA using any of the MTA interfaces (either required in the IPCablecom Specifications or proprietary provided by the vendor) without modifications to the MTA.

In particular, an MTA subscriber may also be connected to the Internet via a CM (which may be embedded in the same MTA). In that case, hackers may potentially exploit any weakness in the configuration of the subscriber's local network and steal MTA's secret and private keys over the network. If instead, the MTA subscriber is connected to a company Intranet, the same threat still exists, although from a smaller group of people.

# Annex A (normative): Additional requirements for J.112 annex A

## A.1 Overview

The access network employs J.112 annex A based CMs. When J.112 annex A based CMs are used in the access network , the requirements described in this annex must be adhered to.

## A.2 Requirements

All MTAs MUST use J.112 annex A compliant CMs and MUST implement the security option described in J.112 annex A. The security option provides security services to the data link layer traffic streams running across the cable access network, i.e. between CM and INA. These services are message confidentiality and access control which provide CM users with data privacy across the cable network and protect cable operators from theft of service.

The INA MUST encrypt all packet cable traffic in the downstream direction.

The CM MUST encrypt all packet cable traffic in the upstream direction.

## A.3 Security Mechanisms Provided

The protected J.112 annex A data communications services fall into three categories:

- best-effort, high-speed, IP data services;
- QoS (e.g. constant bit-rate) data services; and
- IP multicast group services.

Employing the J.112 annex A security option which meets the above requirements, the INA protects against unauthorized access to these data transport services by enforcing encryption of the associated traffic flows across the cable network. Key management is performed using an extended set of MAC messages. For unicast streams, the keys are derived using Diffie-Hellman between the INA and CM. For mulitcast streams, a client/server approach is adopted with the INA controlling the distribution of the keys to the CMs. During the key exchange for both unicast and multicast streams the CM is authenticated by the INA.

## A.4 Packet Data Encryption

J.112 annex A encryption services are defined as a set of optional services within the MAC. Encryption can be selectively applied to the various payload data streams. For each secure stream two session keys can be used for encrypting and decrypting it of which only one of the keys is used to process any particular payload unit. Each key can be used for processing both upstream and downstream payload data.

Having two keys allows negotiation of a new key to take place while payload data is processed using the old one, and then an immediate switch-over can be performed once the new key is agreed upon, without interrupting payload traffic. The INA initiates the key exchanges, and can start using a session key for downstream traffic encryption once the key exchange is complete. For upstream traffic encryption, the NIU should use whichever key was used by the INA in the most recent payload unit.

A payload stream is identified by either of:

- A 24-bit (UNI) ATM virtual circuit VPI/VCI: this is used for ATM-based IB downstream, OOB downstream, and upstream payload data. The ATM circuit can be one-to-one, or one end-point of a multicast circuit.

- A 48-bit MAC-address: this is used for Multiprotocol Encapsulation downstream payload data. The MAC-address can be the physical address of the STB or a pseudo address used for MAC-address based multicasting.

For ATM-based payload streams, the unit of encryption is a single ATM cell. The 48-byte cell payload is encrypted using the security context implied by the 24-bit VPI/VCI of the cell header. For Multiprotocol Encapsulation payload streams, the unit of encryption is a single Multiprotocol Encapsulation section. The datagram_data_bytes (between the MAC-address and the CRC/checksum) are encrypted using the security context implied by the 48-bit MAC address in the section header.

Bits in the ATM header (GFC field) and in the MPE section (scrambling field) are used to identify whether the payload is encrypted and the session key it was encrypted with.

The currently supported algorithms are 40 and 56-bit DES in CBC mode. Additional algorithms may be support in later revisions of ES 200 800.

## A.5 Key Management

This is achieved by using Diffie-Hellman, which requires no up-front shared secret, or a simpler protocol based on a long-term shared secret between INA and CM called a cookie. The cookie is also used for authenticating the CM to the INA during the key exchanges. Three mechanisms exist for establishing a shared key between the INA and the CM and all three are initiated by the INA, these are:

**Main Key Exchange**: This uses Diffie-Hellman to derive a shared secret between the INA and CM, which is independent of the cookie value. It can also be used to update the cookie value held in the CM.

**Quick Key Exchange**: This uses the existing cookie value to derive a shared secret key.

**Explicit Key Exchange**: This is used by the INA to deliver a pre-determined session key to the CM. The session key is encrypted under a temporary key derived from the cookie value.

As stated earlier, two session keys can be in place for a given stream, therefore during the key exchange the particular session key being generated is identified by the INA. The above mechanisms are also used to update the keys for an active stream. See ES 200 800 for further information about key management.

# Annex B (normative): Additional requirements for J.112 annex B

All MTAs MUST use J.112 compliant CMs and MUST implement BPI+. Baseline Privacy Plus (BPI+) provides security services to the J.112 data link layer traffic flows running across the cable access network, i.e. between CM and AN. These services are message confidentiality and access control. The BPI+ security services operating in conjunction with J.112 provide CM users with data privacy across the cable network and protect cable operators from theft of service.

The protected J.112 MAC data communications services fall into three categories:

- best-effort, high-speed, IP data services;
- QoS (e.g. constant bit-rate) data services; and
- IP multicast group services.

Employing BPI+, the AN protects against unauthorized access to these data transport services by (1) enforcing encryption of the associated traffic flows across the cable network and (2) authenticating the CM MAC management messages that CMs use to establish QoS service flows. BPI+ employs a client/server key management protocol in which the AN (the server) controls distribution of keying material to client CMs. The key management protocol ensures that only authorized CMs receive the encryption and authentication keys needed to access the protected services.

Baseline Privacy Plus has two component protocols:

- An encapsulation protocol for encrypting packet data across the cable network. This protocol defines (1) the frame format for carrying encrypted packet data within CM MAC frames, (2) a set of supported *cryptographic suites*, i.e. pairings of data encryption and authentication algorithms, and (3) the rules for applying those algorithms to a CM MAC frame's packet data.
- A key management protocol (Baseline Privacy Key Management, or "BPKM") provides the secure distribution of keying data from AN to CMs. Through this key management protocol, CM and AN synchronize keying data; in addition, the AN uses the protocol to enforce conditional access to network services.

## B.1 BPI+ Packet Data Encryption

BPI+ encryption services are defined as a set of extended services within the CM MAC sublayer. Packet Header information specific to BPI+ is placed in a Baseline Privacy Extended Header element within the MAC Extended Header. BPI+ encrypts a CM MAC Frame's packet data; the CM MAC Frame's Header is not encrypted.

Currently, BPI+ supports a single packet date encryption algorithm: the Cipher Block Chaining (CBC) mode of the US Data Encryption Standard (DES). BPI+ does not pair DES CBC with any packet data authentication algorithm. Additional data encryption algorithms may be supported in future enhancements to the BPI+ protocol Specification, and these algorithms may be paired with data authentication algorithms.

## B.2 BPI+ Key Management Protocol

CMs use the Baseline Privacy Key Management (BPKM) protocol to obtain traffic keying material from the AN. The key management protocol uses X.509 digital certificates, RSA public key encryption and two-key triple DES to secure key exchanges between CM and AN.

The Baseline Privacy Key Management protocol adheres to a client/server model, where the CM, a BPKM "client", requests keying material, and the AN, a BPKM "server", responds to those requests, ensuring individual CM clients only receive keying material they are authorized for. The BPKM protocol uses CM MAC management messaging.

BPI+ uses public-key cryptography to establish a shared secret (i.e. an Authorization Key) between CM and AN. The shared secret is then used to secure subsequent BPKM exchanges of traffic encryption keys. This two-tiered mechanism for key distribution permits refreshing of traffic encryption keys without incurring the overhead of computation-intensive public-key operations.

Each CM carries a unique X.509 digital certificate issued by the CM's manufacturer. The digital certificate contains the CM's Public Key along with other identifying information; i.e. CM MAC address, manufacturer ID and serial number. When requesting an Authorization Key, a CM presents its digital certificate to a AN. The AN verifies the digital certificate, and then uses the verified Public Key to encrypt an Authorization Key, which the AN then sends back to the requesting CM.

The AN indirectly authenticates the client CM during the subsequent requests and responses for the CM's traffic encryption keys. Access to the authorization key is required to successfully complete these exchanges and the CM requires the private key paired with the verified digital certificate's subject public key in order to gain access to that authorization key.

The AN associates a CM's authenticated identity to a paying subscriber, and hence to the data services that subscriber is authorized to access. Thus, with the Authorization Key exchange, the AN establishes an authenticated identity of a client CM, and the services (i.e. specific traffic encryption keys) the CM is authorized to access.

Since the AN authenticates CMs, it can protect against an attacker employing a *cloned* modem, masquerading as a legitimate subscriber's modem. The use of the X.509 certificates prevents cloned modems from passing fake credentials onto an AN.

Authentication is one-way; CMs do not authenticate the AN they exchange BPKM messages with. CM designers did not view AN spoofing as a significant threat due to the high cost and limited scope of such an attack.

## B.3 Secure Software upgrade

The scope of IPCablecom includes only Embedded MTAs. Therefore IPCablecom MTAs MUST be embedded with the J.112 CM which MUST implement BPI+. IPCablecom Embedded MTAs will have their software upgraded according to the J.112 requirements. The CM will verify the code file using CM parameters that include the CM root key and the Code Verification Certificate (CVC).

The future implementation of secure software upgrades for Standalone MTAs is expected to utilize a similar method for secure software upgrades. The details of these requirements will be defined in a subsequent version of the present document.

# Annex C (informative): Example of MMH Algorithm Implementation

This annex gives an example implementation of the MMH MAC algorithm. There may be other implementations that have advantages over this example in particular operating environments. This example is for informational purposes only and is meant to clarify the specification.

The example implementation uses the term "MMH16" for the case where the MAC length is 2 octets and "MMH32" for the case where the length is 4 octets.

A main program is included for exercising the example implementation. The output produced by the program is included.

```
/*
    Demo of IPCablecom  MMH16 and MMH32 MAC algorithms.

    This program has been tested using Microsoft C/C++ Version 5.0.
    It is believed to port easily to other compilers, but this has
    not been tested.  When porting, be sure to pick the definitions
    for int16, int32, uint16, and uint32 carefully.
*/

#include <stdio.h>

/*
Define signed and unsigned integers having 16 and 32 bits.
This is machine/compiler dependent, so pick carefully.
*/
typedef short          int16;
typedef unsigned short uint16;
typedef int            int32;
typedef unsigned int   uint32;

/*
Define this symbol to see intermediate values.
Comment it out for clean display.
*/
#define VERBOSE

int32 reduceModF4(int32 x) {


    /*
    Routine to reduce an int32 value modulo F4, where F4 = 0x10001.
    Result is in range [0, 0x10000].
    */


    int32 xHi, xLo;

    /* Range of x is [0x80000000, 0x7fffffff]. */

    /*
    If x is negative, add a multiple of F4 to make it non-negative.
    This loop executes no more than two times.
    */
    while (x < 0) x += 0x7fff7fff;

    /* Range of x is [0, 0x7fffffff]. */

    /* Subtract high 16 bits of x from low 16 bits. */
    xHi = x >> 16;
    xLo = x & 0xffff;
    x = xLo - xHi;

    /* Range of x is [0xffff8001, 0x0000ffff]. */

    /* If x is negative, add F4. */
    if (x < 0) x += 0x10001;

    /* Range of x is [0, 0x10000]. */
```

```
    return x;
}

uint16 mmh16(
    unsigned char *message,
    unsigned char *key,
    unsigned char *pad,
    int msgLen
) {

    /*
    Compute and return the MMH16 MAC of the message using the
    indicated key and pad.

    The length of the message is msgLen bytes; msgLen must be even.

    The length of the key must be at least msgLen bytes.

    The length of the pad is two bytes.  The pad must be freshly
    picked from a secure random source.
    */

    int16 x, y;
    uint16 u, v;
    int32 sum;
    int i;

    sum = 0;

    for (i=0; i<msgLen; i+=2) {

        /* Build a 16-bit factor from the next two message bytes. */
        x = *message++;
        x <<= 8;
        x |= *message++;

        /* Build a 16-bit factor from the next two key bytes. */
        y = *key++;
        y <<= 8;
        y |= *key++;

        /* Accumulate product of the factors into 32-bit sum */
        sum += (int32)x * (int32)y;

        #ifdef VERBOSE
        printf("  x %04x  y %04x  sum %08x\n", x & 0xffff, y & 0xffff, sum);
        #endif

    }

    /* Reduce sum modulo F4 and truncate to 16 bits. */
    u = (uint16) reduceModF4(sum);

    #ifdef VERBOSE
    printf("  sum mod F4, truncated to 16 bits:  %04x\n", u & 0xffff);
    #endif

    /* Build the pad variable from the two pad bytes */
    v = *pad++;
    v <<= 8;
    v |= *pad;

    #ifdef VERBOSE
    printf("  pad variable:  %04x\n", v & 0xffff);
    #endif

    /* Accumulate pad variable, truncate to 16 bits */
    u = (uint16)(u + v);

    #ifdef VERBOSE
    printf("  mmh16 value:  %04x\n", u & 0xffff);
    #endif

    return u;
}

uint32 mmh32(
    unsigned char *message,
```

```
    unsigned char *key,
    unsigned char *pad,
    int msgLen
) {

    /*
    Compute and return the MMH32 MAC of the message using the
    indicated key and pad.

    The length of the message is msgLen bytes; msgLen must be even.

    The length of the key must be at least (msgLen + 2) bytes.

    The length of the pad is four bytes.  The pad must be freshly
    picked from a secure random source.
    */

    uint16 x, y;
    uint32 sum;

    x = mmh16(message, key,   pad,   msgLen);
    y = mmh16(message, key+2, pad+2, msgLen);
    sum = x;
    sum <<= 16;
    sum |= y;

    return sum;
}


void show(char *name, unsigned char *src, int nbytes) {


    /*
    Routine to display a byte array, in normal or reverse order
    */

    int i;
    enum {
        BYTES_PER_LINE = 16
    };

    if (name) printf("%s", name);

    for (i=0; i<nbytes; i++) {
        if ((i % BYTES_PER_LINE) == 0) printf("\n");
        printf("%02x ", src[i]);
    }
    printf("\n");
}

int main() {

    uint16 mac16;
    uint32 mac32;

    unsigned char message[] = {
        0x4e, 0x6f, 0x77, 0x20, 0x69, 0x73, 0x20, 0x74, 0x68,
        0x65, 0x20, 0x74, 0x69, 0x6d, 0x65, 0x2e,
    };

    unsigned char key[] = {
        0x35, 0x2c, 0xcf, 0x84, 0x95, 0xef, 0xd7, 0xdf, 0xb8,
        0xf5, 0x74, 0x05, 0x95, 0xeb, 0x98, 0xd6, 0xeb, 0x98,
    };

    unsigned char pad16[] = {
        0xae, 0x07,
    };

    unsigned char pad32[] = {
        0xbd, 0xe1, 0x89, 0x7b,
    };

    unsigned char macBuf[4];


    printf("Example of MMH16 computation\n");
```

```
    show("message", message, sizeof(message));
    show("key", key, sizeof(message));
    show("pad", pad16, 2);

    mac16 = mmh16(message, key, pad16, sizeof(message));
    macBuf[1] = (unsigned char)mac16; mac16 >>= 8;
    macBuf[0] = (unsigned char)mac16;

    show("MMH16 MAC", macBuf, 2);
    printf("\n");

    printf("Example of MMH32 computation\n");
    show("message", message, sizeof(message));
    show("key", key, sizeof(message)+2);
    show("pad", pad32, 4);

    mac32 = mmh32(message, key, pad32, sizeof(message));
    macBuf[3] = (unsigned char)mac32; mac32 >>= 8;
    macBuf[2] = (unsigned char)mac32; mac32 >>= 8;
    macBuf[1] = (unsigned char)mac32; mac32 >>= 8;
    macBuf[0] = (unsigned char)mac32;

    show("MMH32 MAC", macBuf, 4);
    printf("\n");

    return 0;
}

Here is the output produced by the program:
Example of MMH16 computation
message
4e 6f 77 20 69 73 20 74 68 65 20 74 69 6d 65 2e
key
35 2c cf 84 95 ef d7 df b8 f5 74 05 95 eb 98 d6
pad
ae 07
  x 4e6f  y 352c  sum 104a7614
  x 7720  y cf84  sum f9bac294
  x 6973  y 95ef  sum ce0a23f1
  x 2074  y d7df  sum c8f3d4fd
  x 6865  y b8f5  sum abfb55a6
  x 2074  y 7405  sum bab087ea
  x 696d  y 95eb  sum 8f00bff9
  x 652e  y 98d6  sum 663aa46d
  sum mod F4, truncated to 16 bits:  3e33
  pad variable:  ae07
  mmh16 value:  ec3a
MMH16 MAC
ec 3a

Example of MMH32 computation
message
4e 6f 77 20 69 73 20 74 68 65 20 74 69 6d 65 2e
key
35 2c cf 84 95 ef d7 df b8 f5 74 05 95 eb 98 d6
eb 98
pad
bd e1 89 7b
  x 4e6f  y 352c  sum 104a7614
  x 7720  y cf84  sum f9bac294
  x 6973  y 95ef  sum ce0a23f1
  x 2074  y d7df  sum c8f3d4fd
  x 6865  y b8f5  sum abfb55a6
  x 2074  y 7405  sum bab087ea
  x 696d  y 95eb  sum 8f00bff9
  x 652e  y 98d6  sum 663aa46d
  sum mod F4, truncated to 16 bits:  3e33
  pad variable:  bde1
  mmh16 value:  fc14
  x 4e6f  y cf84  sum f125323c
  x 7720  y 95ef  sum bfca091c
  x 6973  y d7df  sum af427949
  x 2074  y b8f5  sum a640e84d
  x 6865  y 7405  sum d590b646
  x 2074  y 95eb  sum c81e04c2
  x 696d  y 98d6  sum 9da1dde0
  x 652e  y eb98  sum 95912b30
  sum mod F4, truncated to 16 bits:  959f
```

```
  pad variable:  897b
  mmh16 value:  1f1a
MMH32 MAC
fc 14 1f 1a
```

# Annex D (informative): IPCablecom Administration Guidelines and Best Practices

This annex describes various administration guidelines and best practices recommended by IPCablecom. These are included to help facilitate network administration and/or strengthen overall security in the IPCablecom network.

## D.1 Routine CMS Service Key Refresh

IPCablecom recommends that the CMS service keys be routinely changed (refreshed) at least once every 90 days in order to reduce the risk of key compromises. The refresh period should a provisioned parameter that can be use in one the following ways:

1) In the case manual key changes, an administrator is prompted or reminded to manually change a CMS service key.

2) In the case of autonomous key changes (using Kerberos Set/Change Password) it will define the refresh period.

Note that in the case of autonomous key refreshes, whereby administrative overhead and scalability are not an issue, it may be desirable to use a refresh period that is less than ninety days (but at least the maximum ticket lifetime). This may further reduce the risk of key compromise.

# Annex E (informative): Bibliography

Schneier "Applied Cryptography," John Wiley & Sons Inc, second edition, 1996.

"How to Protect DES Against Exhaustive Key Search", J. Killian, P. Rogaway (Edited version presented at Proceedings of Crypto '96), July 1997.

S. Halevi and H. Krawczyk, "MMH: Software Message Authentication in Gbit/s Rates", Proceedings of the 4th Workshop on Fast Software Encryption, (1997) vol. 1267 Springer-Verloag, pp. 172-189.

SCTE-DSS-00-09: "Baseline Privacy Interface Plus Specification".

RFC 1035 (1987): "Domain names - implementation and specification" (Status: Standard).

RFC 1750 (1994): "Randomness Recommendations for Security", Donald Eastlake, Stephen Crocker and Jeff Schiller.

RFC 2137: "Secure Domain Name System Dynamic Update".

RFC 2246 (1999): "The TLS Protocol Version 1.0".

RFC 2327 (1998): "SDP: Session Description Protocol".

RFC 2404 (1998): "The Use of HMAC-SHA-1-96 within ESP and AH".

RFC 2630 (1999): "Cryptographic Message Syntax".

IETF draft: "Public Key Cryptography for Initial Authentication in Kerberos", http://www.ietf.org/internet-drafts/draft-ietf-cat-kerberos-pk-init-14.txt.

IETF draft (June 2001): "Kerberos Set/Change Password", Version 2 (J. Trostle, M. Swift, J. Brezak, B. Gossman), http://www.ietf.org/internet-drafts/draft-ietf-cat-kerberos-set-passwd-06.txt.

IETF draft (February 2001): "IPsec Monitoring MIB", http://www.ietf.org/internet-drafts/draft-ietf-ipsec-monitor-mib-04.txt.

ETSI TS 101 909-2: "Access and Terminals (AT); Digital Broadband Cable Access to the Public Telecommunications Network; IP Multimedia Time Critical Services; Part 2: Architectural framework for the delivery of time critical services over cable Television networks using cable modems".

ETSI TS 101 909-3: "Access and Terminals (AT); Digital Broadband Cable Access to the Public Telecommunications Network; IP Multimedia Time Critical Services; Part 3: Audio Codec Requirements for the Provision of Bi-Directional Audio Service over Cable Television Networks using Cable Modems".

ETSI TS 101 909-8: "Access and Terminals (AT); Digital Broadband Cable Access to the Public Telecommunications Network; IP Multimedia Time Critical Services; Part 8: Media Terminal Adaptor (MTA) Management Information Base (MIB)".

ETSI TS 101 909-9: "Access and Terminals (AT); Digital Broadband Cable Access to the Public Telecommunications Network; IP Multimedia Time Critical Services; Part 9: Network Based Call Signalling (NCS) Management Information Base (MIB) Requirements".

ETSI TS 101 909-10: "Access and Terminals (AT); Digital Broadband Cable Access to the Public Telecommunications Network; IP Multimedia Time Critical Services; Part 10: Event Message Requirements for the Provision of Real Time Services over Cable Television Networks using Cable Modems".

ETSI TS 101 909-12: "Access and Terminals (AT); Digital Broadband Cable Access to the Public Telecommunications Network; IP Multimedia Time Critical Services; Part 12: Internet Signalling Transport Protocol".

ETSI TS 101 909-13: "Access and Terminals (AT); Digital Broadband Cable Access to the Public Telecommunications Network; IP Multimedia Time Critical Services; Part 13: Trunking Gateway Control Protocol".

ITU-T Recommendation X.680 (1997): "Information technology - Abstract Syntax Notation One (ASN.1): Specification of basic notation".

ITU-T Recommendation X.681 (1997): "Information technology - Abstract Syntax Notation One (ASN.1): Information object specification".

ITU-T Recommendation X.682 (1997): "Information technology - Abstract Syntax Notation One (ASN.1): Constraint Specification".

ITU-T Recommendation X.683 (1997): "Information technology - Abstract Syntax Notation One (ASN.1): Parameterization of ASN.1 specifications".

ITU-T Recommendation X.690 (1997): "Information technology - ASN.1 encoding rules - Specification of Basic Encoding Rules (BER), Conanical Encoding Rules (CER) and Distinguished Encoding Rules (DER)".

List of ITU-T Recommendations referring to IP Cablecom:

ITU-T Recommendation J.160: "Architectural framework for the delivery of time critical services over cable television networks using cable modems".

ITU-T Recommendation J.161: "Audio codec requirements for the provision of bidirectional audio service over cable television networks using cable modems".

ITU-T Recommendation J.162: "Network call signalling (NCS) MIB requirements".

ITU-T Recommendation J.163: "Dynamic quality of service for the provision of real time services over cable television networks using cable modems".

ITU-T Recommendation J.164: "Event Message requirements for the support of real-time services over cable television networks using cable modems".

ITU-T Recommendation J.165: "IPCablecom Internet Signalling Transport Protocol".

ITU-T Recommendation J.166: "IPCablecom Management information base (MIB) framework".

ITU-T Recommendation J.167: "Media terminal adapter (MTA) device provisioning requirements for the delivery of real-time services over cable television networks using cable modems".

ITU-T Recommendation J.168: "IPCablecom media terminal adapter (MTA) MIB requirements".

ITU-T Recommendation J.169: "IPCablecom network call signalling (NCS) MIB requirements".

ITU-T Recommendation J.170: "IPCablecom Security specification".

ITU-T Recommendation J.171: "IPCablecom Trunking Gateway Control Protocol (TGCP)".

# History

| Document history | | |
|---|---|---|
| V1.1.1 | July 2001 | Publication |
| | | |
| | | |
| | | |
| | | |