ETSI TS 103 161-9 V1.1.1 (2011-10)

**Access, Terminals, Transmission and Multiplexing (ATTM);
Integrated Broadband Cable and Television Networks;
IPCablecom 1.5;
Part 9: Security**

Reference

DTS/ATTM-003011-9

Keywords

access, broadband, cable, IP, multimedia, PSTN

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://ipr.etsi.org).

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, Terminals, Transmission and Multiplexing (ATTM).

The present document is part 9 of a multi-part IPCablecom 1.5 deliverable covering the Digital Broadband Cable Access to the Public Telecommunications Network; IP Multimedia Time Critical Services, as identified below:

Part 1: "Overview";

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: "Event Message Specification";

Part 7: "Media Terminal Adapter (MTA Management Information Base (MIB)";

Part 8: "Network Call Signalling (NCS) MIB Requirements";

Part 9: "Security";

Part 10: "Management Information Base (MIB) Framework";

Part 11: "Media terminal adapter (MTA) device provisioning";

Part 12: "Management Event Mechanism";

Part 13: "Trunking Gateway Control Protocol - MGCP option";

Part 14: "Embedded MTA Analog Interface and Powering Specification"

Part 15: "Analog Trunking for PBX Specification";

Part 16: "Signalling for Call Management Server";

Part 17: "CMS Subscriber Provisioning Specification";

Part 18: "Media Terminal Adapter Extension MIB";

Part 19: "IPCablecom Audio Server Protocol Specification - MGCP option";

Part 20: "Management Event MIB Specification";

Part 21: "Signalling Extension MIB Specification".

NOTE 1: Additional parts may be proposed and will be added to the list in future versions.

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

1 Scope and Introduction

1.1 Scope

The scope of the present document is to define 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, signalling and media content integrity, confidentiality, and non-repudiation security services are provided as defined herein for each of the network element interfaces.

1.2 Goals

The present document describes the security relationships between the elements on the IPCablecom network. 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 defines 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 need to be safe from eavesdropping. Unauthorized message modification, insertion, deletion and replays anywhere in the network need to be easily detectable and not affect proper network operation.
- **Reasonable cost** - The IPCablecom network security defines 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 are to 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 provide a framework into which new security methods and algorithms may be incorporated as necessary.

1.2.1 Assumptions

The following assumptions are made relative to the current scope of the present document:

- Embedded Multimedia Terminal Adapters (E-MTAs) and Standalone Multimedia Terminal Adapters (S-MTAs) are within the scope of the present document.
- NCS is the only call signalling method, on the access network, addressed in the present document.
- This version of the IPCablecom Security Specification specifies security for a single administrative domain and the communications between domains.
- Security for chained RADIUS servers is not currently in the scope.
- The IPCablecom Security Specification does not have a requirement for exportability outside the United States; 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 handling. Record Keeping Servers (RKS), Network Management Systems, File Transfer Protocol (FTP) servers and Dynamic Host Configuration Protocol (DHCP) servers are all considered to be unique to any service provider's implementation and are beyond the scope of the present document.

- The present document assumes that MTAs implement the specified IPCablecom 1.5 modules, and optionally, can implement the IETF IPCDN MIB modules. As such, any reference to a specific MIB Object is assumed to be a reference in either MIB module unless explicitly specified or the MIB Object exists in only one set of MIB modules.

1.2.2 Requirements

The following requirement is made relative to the current scope of the IPCablecom Security Specification:

- All E-MTAs are to use DOCSIS® (1.1 or later) compliant cable modems and implement BPI+ [9]. Any references to DOCSIS®, including specific references to DOCSIS® versions 1.1 or 2.0, are understood to refer to DOCSIS® version 1.1 or later.

2 References

References are either specific (identified by date of publication and/or edition number or version number) or non-specific. For specific references, only the cited version applies. For non-specific references, the latest version of the reference document (including any amendments) applies.

Referenced documents which are not found to be publicly available in the expected location might be found at http://docbox.etsi.org/Reference.

NOTE: While any hyperlinks included in this clause were valid at the time of publication, ETSI cannot guarantee their long term validity.

2.1 Normative references

The following referenced documents are necessary for the application of the present document.

[1] ETSI TS 103 161-2: "Access, Terminals, Transmission and Multiplexing (ATTM); Integrated Broadband Cable and Television Networks; IPCablecom 1.5; Part 2: Architectural framework for the delivery of time critical services over Cable Television Networks using Cable Modems".

[2] ETSI TS 103 161-4: "Access, Terminals, Transmission and Multiplexing (ATTM); Integrated Broadband Cable and Television Networks; IPCablecom 1.5; Part 4: Network Call Signalling Protocol".

[3] ETSI TS 103 161-5: "Access, Terminals, Transmission and Multiplexing (ATTM); Integrated Broadband Cable and Television Networks; IPCablecom 1.5; Part 5: Dynamic Quality of Service for the Provision of Real Time Services over Cable Television Networks using Cable Modems".

[4] ETSI TS 103 161-11: "Access, Terminals, Transmission and Multiplexing (ATTM); Integrated Broadband Cable and Television Networks; IPCablecom 1.5; Part 11: Media Terminal Adapter (MTA) device provisioning".

[5] ETSI TS 103 161-13: "Access, Terminals, Transmission and Multiplexing (ATTM); Integrated Broadband Cable and Television Networks; IPCablecom 1.5; Part 13: Trunking Gateway Control Protocol - MGCP option".

[6] ETSI TS 103 161-6: "Access, Terminals, Transmission and Multiplexing (ATTM); Integrated Broadband Cable and Television Networks; IPCablecom 1.5; Part 6: Event Message Specification".

[7] ETSI TS 103 161-3: "Access, Terminals, Transmission and Multiplexing (ATTM); Integrated Broadband Cable and Television Networks; IPCablecom 1.5; Part 3: Audio Codec Requirements for the Provision of Bi-Directional Audio Service over Cable Television Networks using Cable Modems".

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

[9] ETSI ES 202 488-3: "Access and Terminals (AT); Second Generation Transmission Systems for Interactive Cable Television Services - IP Cable Modems; Part 3: Baseline privacy plus interface specification".

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

[11] IETF RFC 2104 (1997): "HMAC: Keyed-Hashing for Message Authentication".

[12] IETF RFC 2630 (1999): "Cryptographic Message Syntax".

[13] IETF RFC 2866 (2000): "RADIUS Accounting".

[14] IETF RFC 4120 (2005): "The Kerberos Network Authentication Service (V5)".

[15] NIST, FIPS PUB 180-1 (1995): "Secure Hash Standard".

[16] IETF RFC 2437 (1998): "PKCS#1: RSA Cryptography Specifications Version 2.0".

[17] IETF RFC 2246 (1999): "The TLS Protocol Version 1.0".

[18] MMH: "Software Message Authentication in Gbit/second Rates, Proceedings of the 4th Workshop on Fast Software Encryption", (1997), S. Halevi and H. Krawczyk.

[19] IETF RFC 2401 (1998): "Security Architecture for the Internet Protocol".

[20] IETF RFC 2406 (1998): "IP Encapsulating Security Payload (ESP)".

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

[22] IETF RFC 2451 (1998): "The ESP CBC-Mode Cipher Algorithms".

[23] IETF RFC 2404 (1998): "The Use of HMAC-SHA-1-96 within ESP and AH".

[24] IETF RFC 2409 (1998): "The Internet Key Exchange (IKE)".

[25] ETSI TS 103 161-7: "Access, Terminals, Transmission and Multiplexing (ATTM); Integrated Broadband Cable and Television Networks; IPCablecom 1.5; Part 7: Media Terminal Adapter (MTA) Management Information Base (MIB)".

[26] ETSI TS 103 161-8: "Access, Terminals, Transmission and Multiplexing (ATTM); Integrated Broadband Cable and Television Networks; IPCablecom 1.5; Part 8: Network Call Signalling (NCS) MIB Requirements".

[27] ETSI TS 103 161-19: "Access, Terminals, Transmission and Multiplexing (ATTM); Integrated Broadband Cable and Television Networks; IPCablecom 1.5; Part 19: IPCablecom Audio Server Protocol Specification - MGCP option".

[28] IETF RFC 3414 (2002): "User-based Security Model (USM) for version 3 of the Simple Network Management Protocol (SNMPv3)".

[29] IETF RFC 2367 (1998): "PF-KEY Key Management API, Version 2".

[30] ETSI TS 102 836-1 (V1.1.1): "Access, Terminals, Transmission and Multiplexing (ATTM); Lawful Interception (LI); Part 1: Interception of IP Telephony Service on Cable Operator's Broadband IP Network: Internal Network Interfaces".

[31] NIST, FIPS PUB 81 (1980): "DES Modes of Operation".

[32] ETSI TS 103 161-16: "Access, Terminals, Transmission and Multiplexing (ATTM); Integrated Broadband Cable and Television Networks; IPCablecom 1.5; Part 16: Signalling for Call Management Server".

[33] ITU-T Recommendation X.509 (1997 E): "Information Technology - Open Systems Interconnection - The Directory: Authentication Framework", June 1997.

[34] IETF RFC 2459 (1999): "Internet X.509 Public Key Infrastructure Certificate and CRL Profile".

[35] NIST, FIPS PUB 197 (2001): "Advanced Encryption Standard (AES)".

[36] ITU-T Recommendation X.690 (2008): "Information technology - ASN.1 encoding rules: Specification of Basic Encoding Rules (BER), Canonical Encoding Rules (CER) and Distinguished Encoding Rules (DER)".

[37] IETF RFC 2403 (1998): "The Use of HMAC-MD5-96 within ESP and AH".

[38] IETF RFC 3412 (2002): "Message Processing and Dispatching for the Simple Network Management Protocol (SNMP)".

[39] IETF RFC 2782 (2000): "A DNS RR for specifying the location of services (DNS SRV)".

[40] IETF RFC 3261 (2002): "SIP: Session Initiation Protocol".

[41] IETF RFC 3268 (2002): "Advanced Encryption Standard (AES) Ciphersuites for Transport Layer Security (TLS)".

[42] IETF RFC 1890 (1996): "RTP Profile for Audio and Video Conferences with Minimal Control".

[43] "Applied Cryptography", B. Schneier, John Wiley and Sons Inc, second edition, 1996.

[44] IETF RFC 1750 (1994): "Randomness Recommendations for Security".

[45] "How to Protect DES Against Exhaustive Key Search", J. Kilian, P. Rogaway, February 2, 2000.

[46] NIST, FIPS PUB 140-2 (2001): "Security Requirements for Cryptographic Modules".

[47] IETF RFC 4682 (2006): "Multimedia Terminal Adapter (MTA) Management Information Base for PacketCable- and IPCablecom-Compliant Devices".

[48] IETF RFC 5098 (2008): "Signaling MIB for PacketCable and IPCablecom Multimedia Terminal Adapters (MTAs)".

[49] IETF RFC 4556 (2006): "Public Key Cryptography for Initial Authentication in Kerberos (PKINIT)".

[50] IETF RFC 1510 (1993): "The Kerberos Network Authentication Service (V5)".

2.2 Informative references

The following referenced documents are not necessary for the application of the present document but they assist the user with regard to a particular subject area.

Not applicable.

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

audio server: plays informational announcements in IPCablecom network

NOTE: Media announcements are needed for communications that do not complete and to provide enhanced information services to the user. The component parts of Audio Server services are Media Players and Media Player Controllers.

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

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

cipher: algorithm that transforms data between plaintext and ciphertext

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

NOTE: 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 anyone other than the intended parties. Information is encrypted to provide confidentiality

NOTE: Also known as privacy.

cryptanalysis: process of recovering the plaintext of a message or the encryption key without access to the key

cryptographic algorithm: algorithm used to transfer text between plaintext and ciphertext

Diffie-Hellman: specific method of exchanging keys allowing two parties that have no prior knowledge of each other to jointly establish a shared secret key over an insecure communications channel

downstream: direction from the headend toward the subscriber location

encryption: method used to translate plaintext into ciphertext

encryption key: key used in a cryptographic algorithm to translate the plaintext to ciphertext

event message: message capturing a single portion of a connection

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

NOTE: Examples are the Media Gateway, which provides the bearer circuit interfaces to the PSTN and transcodes the media stream, and the Signaling 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

network management: functions related to the management of data across the network

nonce: random value used only once that is sent in a communications protocol exchange to prevent replay attacks

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

Oakley: protocol by which two authenticated parties can agree on secure and secret keying material using the Diffie-Hellman key exchange algorithm

plaintext: original (unencrypted) state of a message or data

NOTE: Also called cleartext.

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

NOTE: 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 is to be kept secret

Proxy: facility that indirectly provides some service or acts as a representative in delivering information, thereby eliminating the need for a host to support the service

public key: key used in public key cryptography that belongs to an individual entity and is distributed publicly. Other entities use this key to encrypt data to be sent to the owner of the key

root private key: private signing key of the highest-level Certification Authority

NOTE: 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 generated with the corresponding root private key

symmetric key: cryptographic key used in a symmetric key algorithm, which results in the secrecy of the encrypted data depending solely upon keeping the key a secret, also known as a secret key

upstream: direction from the subscriber location toward the headend

X.509 certificate: public key certificate recommendation

3.2 Abbreviations

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

AES	Advanced Encryption Standard
AH	Authentication header
AKE	Authenticated Key Agreement
AP	Access Point
AS	Authentication Server
ASCII	American Standard Code for Information Interchange
ASD	Application-Specific Data
ASN	Access Service Network
ASP	Audio Server Protocol
BPI	Baseline Privacy Interface
BPI+	Baseline Privacy Plus Interface Specification
BPKM	Baseline Privacy Key Management
CA	Certification Authority
CBC	Cipher Block Chaining mode
CM	DOCSIS® Cable Modem
CMS	Call Management Server
CMS	Cryptographic Message Syntax
CMSS	Call Management Server Signaling
CMTS	Cable Modem Termination System
Codec	COder-DECoder
COPS	Common Open Policy Service
CRCX	Create Connection
CSR	Customer Service Representative
DER	Distinguished Encoding Rules
DES	Data Encryption Standard
DF	Delivery Function
DHCP	Dynamic Host Configuration Protocol
DNS	Domain Name Service
DOCSIS®	Data-Over-Cable Service Interface Specifications
DOI	Domain of Interpretation
DQoS	Dynamic Quality-of-Service
DTMF	Dual-tone Multi Frequency (tones)
EBP	Exterior Border Proxies
E-MTA	Embedded Multimedia Terminal Adapters
ESP	IPsec Encapsulating Security Payload
FQDN	Fully Qualified Domain Name
GC	Gate Controller

HFC	Hybrid Fibre/Coaxial
HMAC	Hashed Message Authentication Code
HTTP	Hypertext Transfer Protocol
ID	Identification
IETF	Internet Engineering Task Force
IKE	Internet Key Exchange
IKE+	The use of IKE with X.509 certificates for authentication
IP	Internet Protocol
IPsec	Internet Protocol Security
IV	Initialization Vector
IVR	Interactive Voice Response system
KDC	Key Distribution Centre
KM	Key Management service
LNP	Local Number Portability
MAC	Media Access Control
MAC	Message Authentication Code
MD5	Message Digest 5
MDCX	Modify Connection
MG	Media Gateway
MGC	Media Gateway Controller
MGCP	Media Gateway Control Protocol
MIB	Management Information Base
MMH	Multilinear Modular Hash
MPC	Media Player Controller
MSB	Most Significant Bit
MSO	Multi-System Operator
MTA	Multimedia Terminal Adapter
NCS	Network Call Signaling
OID	Object Identification
OSS	Operations Systems Support
PKI	Public-Key Infrastructure
PKINIT	Public-Key Cryptography for Initial Authentication
PS/OSS	Packet Switch Operation Support System
PSTN	Public Switched Telephone Network
QoS	Quality of Service
RADIUS	Remote Authentication Dial-In User Service
RFC	Request for Comments
RFI	DOCSIS® Radio Frequency Interface specification
RKS	Record Keeping Server
RSA Key Pair	Public/private key created for use with RSA cryptographic algorithm
RSA	A public-key, or asymmetric, cryptographic algorithm
RTCP	Real-Time Control Protocol
RTP	Real-time Transport Protocol
SA	Security Association
SDP	Session Description Protocol
SG	Signaling Gateway
SHA-1	Secure Hash Algorithm 1
SIP	Session Initiation Protocol
SIP+	Session Initiation Protocol Plus
S-MTA	Standalone Multimedia Terminal Adapters
SNMP	Simple Network Management Protocol
SPI	Security Parameter Index
SRV	Service Record
SS7	Signaling System number 7
SSL	Secure Socket Layer
TCAP	Transaction Capabilities Application Protocol
TCP	Transmission Control Protocol
TD	Timeout for Disconnect
TEXP	Expiration Time
TFTP	Trivial File Transfer Protocol
TGCP	Trunking Gateway Control Protocol
TGS	Ticket Granting Server

TGT	Ticket Granting Ticket
TLS	Transport Layer Security
TLS	Transport Layer Security
TTL	Time To Live
UDP	User Datagram Protocol
UTC	Coordinated Universal Time
VoIP	Voice-over-IP

4 Void

5 Architectural Overview of IPCablecom Security

5.1 IPCablecom Reference Architecture

Security requirements have been defined for every signalling and media link within the IPCablecom IP network. 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 overview, refer to the IPCablecom 1.5 Architecture Framework Technical Report [1].

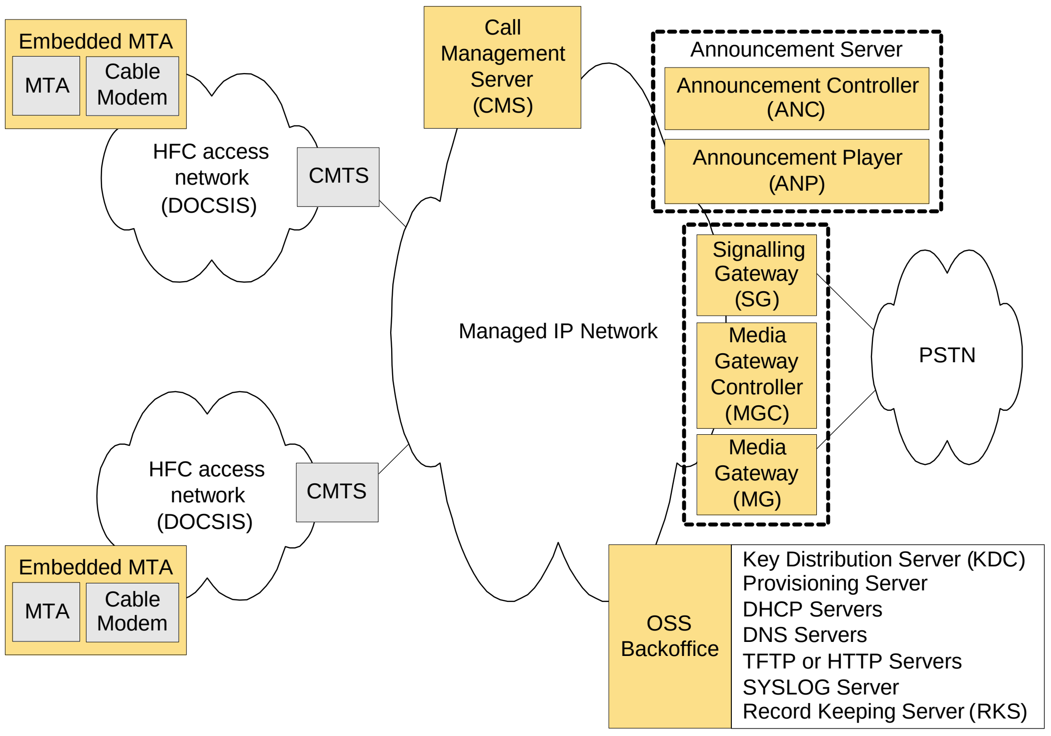


Figure 1: IPCablecom Single Zone Architecture

5.1.1 HFC Network

In figure 1, the Access Network between the MTAs and the CMTS is an HFC network, which employs DOCSIS® 1.1 physical layer and MAC layer protocols [8]. DOCSIS® BPI+ [9] and QoS protocols are enabled over this link.

5.1.2 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 establishment of Dynamic Quality of Service (DQoS). The CMS also performs queries at the PSTN Gateway for LNP (Local Number Portability) and other services necessary for voice communications, including interfacing with the PSTN.

As described in the IPCablecom Architecture Framework [1], 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.
- Media Player Controller (MPC) - The MPC initiates and manages all announcement services provided by the Media Player. The MPC accepts requests from the CMS and arranges for the MP 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.

5.1.3 Functional Categories

The IPCablecom Architecture Framework identifies the following functional categories within the architecture:

- MTA device provisioning
- Quality of Service (HFC access network and managed IP backbone)
- Billing interface security
- Security (specified herein)
- Network call signalling (NCS)
- PSTN interconnectivity
- CODEC functionality and media stream mapping
- Audio Server services
- Lawful Interception (DF interfaces)

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

5.1.3.1 Device and Service Provisioning

During MTA provisioning, the MTA gets its configuration with the help of the DHCP and 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 MTA provisioning, but provides detailed specifications only for the security parameters. Refer to [4] for a full specification on MTA provisioning and customer enrolment.

5.1.3.2 Dynamic Quality of Service

IPCablecom provides guaranteed Quality of Service (QoS) for each voice communication within a single zone with Dynamic QoS (DQoS) [3].

DQoS is controlled by the Gate Controller function within the CMS and can guarantee Quality of Service within a single administrative domain. The Gate Controller utilizes the COPS protocol to download QoS policy into the CMTS. After that, the QoS reservation is established via DOCSIS® 1.1 QoS messaging between the MTA and the CMTS on both sides of the connection.

5.1.3.3 Billing System Interfaces

The CMS, CMTS and the PSTN Gateway are all required to send out billing event messages to the Record Keeping Server (RKS). 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.

5.1.3.4 Call Signalling

The call signalling architecture defined within IPCablecom is Network Based Call Signalling (NCS). The CMS is used to control call setup, termination and most other call signalling functions. In the NCS architecture [2], the Call Agent function within the CMS is used in call signalling and utilizes the MGCP protocol.

5.1.3.5 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. For further detail on PSTN Gateways, refer to [5].

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 the security requirements and specifications for the PSTN Gateway links.

When communications from an MTA to a PSTN phone are made, bearer channel traffic is passed directly between an MTA and an MG. The protocols used in this case are RTP and RTCP, 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 the scope of the present document.

5.1.3.6 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 BPI+ 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.

NOTE 1: In general, it is possible for an MTA-to-MTA or MTA-to-PSTN connection to cross the 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 a 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 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 that can be used to terminate an RTP session.

NOTE 2: The RTCP BYE message should not be confused with the SIP+ BYE message that is also used to indicate the end of a voice communication within the network.

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 (SIP+) signalling and are addressed in the same manner.

In addition to MTAs and PSTN Gateways, Media Servers may also participate in the media stream flows. Media Servers are 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, and so on. Both media stream and signalling interfaces to a Media Server are the same as the interfaces to an MTA. For more information on Codec functionality, see [7].

5.1.3.7 Audio Server Services

Audio Server interfaces provide a suite of signalling protocols for providing announcement and audio services in an IPCablecom network.

5.1.3.7.1 Media Player Controller (MPC)

The Media Player Controller (MPC) initiates and manages all announcement services provided by the Media Player. The MPC accepts requests from the CMS and arranges for the MP to provide the announcement in the appropriate stream so that the user hears the announcement. The MPC also serves as the termination for certain calls routed to it for IVR services. When the MP collects information from the end-user, the MPC is responsible for interpreting this information and managing the IVR session accordingly. The MPC manages call state.

5.1.3.7.2 Media Player (MP)

The Media Player (MP) is a media resource server. It is responsible for receiving and interpreting commands from the MPC and for delivering the appropriate announcement(s) to the MTA. The MP provides the media stream with the announcement contents. The MP also is responsible for accepting and reporting user inputs (e.g. DTMF tones). The MP functions under the control of the MPC.

5.1.3.8 Lawful Interception

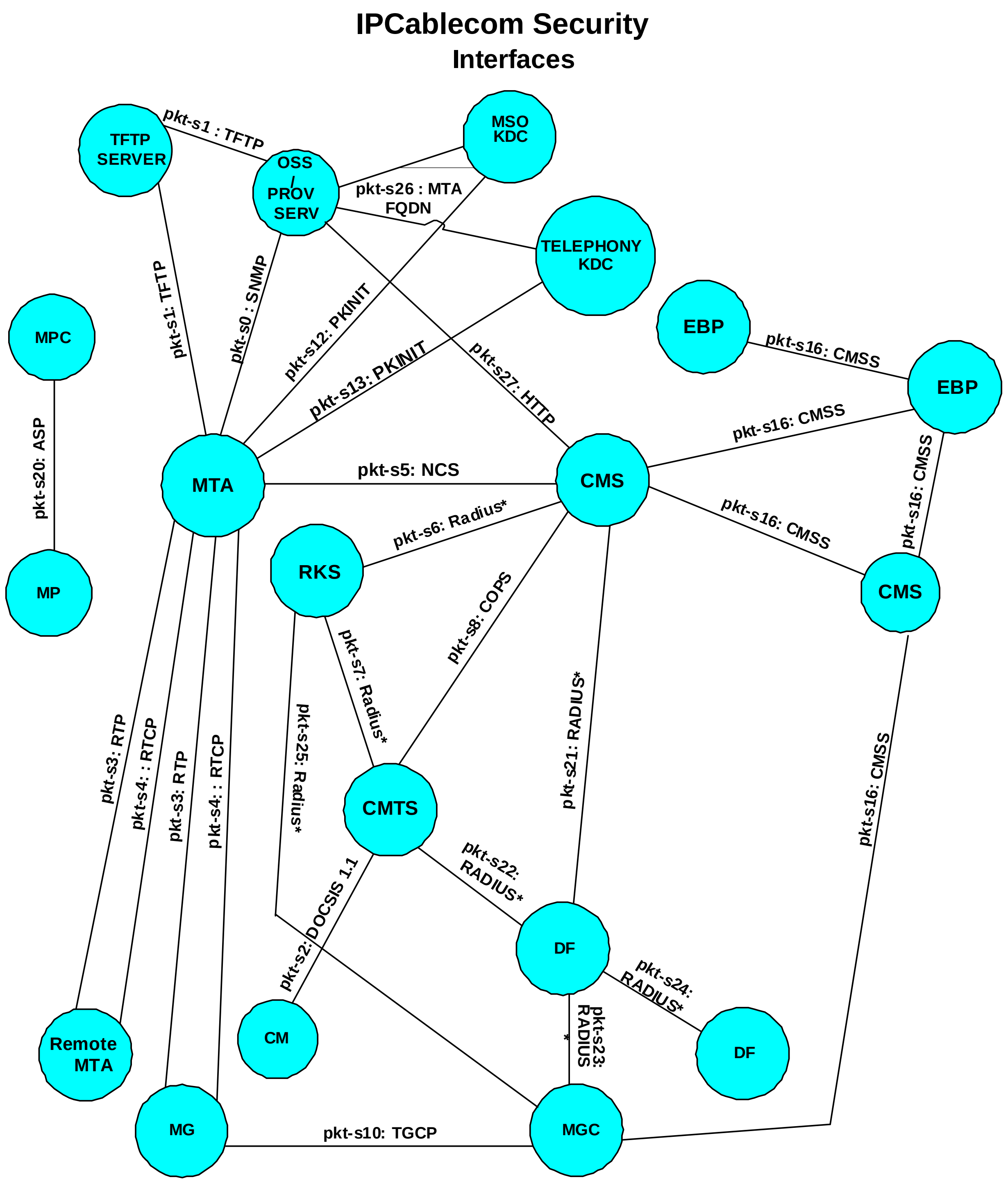
The event interface between the CMS and the DF provides descriptions of calls, necessary to perform wiretapping. This information includes the media stream encryption key and the corresponding encryption algorithm. This event interface uses RADIUS and is similar to the CMS-RKS interface.

The COPS interface between the CMS and the CMTS is used to signal the CMTS to start/stop duplicating media packets to the DF for a particular call. This is the same COPS interface that is used for (DQoS) Gate Authorization messages.

5.2 Threats

Figure 2 contains the interfaces that were analysed for security.

There are additional interfaces 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 the figure 2. The interfaces for which security is not required in IPCablecom are not listed as well.



NOTE: The interfaces marked "RADIUS*" carry event messages, which use the RADIUS format as defined by [13].

Figure 2: IPCablecom Secured Interfaces

Following 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 5.2.6.

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 vendors and cable operators implement fraud management and physical security will differ and in this case a standard is not required for interoperability.

5.2.1 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
- Video conferencing
- Network-based three-way calling
- Quality of Service

5.2.1.1 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 cryptanalysis.

When an MTA is broken into 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 cable operator providing the voice communications service had been setup. The customer name, address, Social Security 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 particularly easy to steal or discover with cryptanalysis.

5.2.1.2 Other Clones

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

5.2.1.3 Subscription Fraud

A customer sets up an account under false information.

5.2.1.4 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 method to revoke the customer's access to the network.

5.2.1.5 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.

5.2.1.6 Protocol Attacks against Other Network Elements

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

5.2.1.7 Theft of Services Provided by the MTA

Services such as the support for multiple MTA ports, 3-way calling and call waiting may be implemented entirely in the MTA, without any required interaction with the network.

5.2.1.7.1 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 clause 5.2.1.7.

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

5.2.1.8 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.

5.2.2 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.

5.2.2.1 Attacks

Clones of MTAs and other Network Elements, as well as protocol manipulation attacks, also apply in the case of Bearer Channel Information threats. These attacks are 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 inconceivable. An owner of an MTA may distribute clones to unsuspecting victims, so that he or she can later spy on them.

5.2.2.1.1 Off-line Cryptanalysis

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

5.2.3 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.

5.2.3.1 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 is 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.

5.2.3.1.1 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.

5.2.3.1.2 Information with Marketing Value

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

5.2.4 Service Disruption Threats

This class of threats is aimed at disrupting the normal operation of 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.

5.2.4.1 Attacks

5.2.4.1.1 Remote Interference

A perpetrator is able to manipulate the protocol to close down on-going voice communications. This might be achieved by masquerading as an MTA involved in such an on-going 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 setup 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 are described under Service Theft threats.

MTA cloning attacks mounted by the actual owner of the MTA can theoretically be 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.

5.2.5 Repudiation

In a network where masquerading (using the above-mentioned cloning and protocol manipulation techniques) is common or easily achievable, a customer may repudiate a particular communication (and, thus 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.

5.2.6 Threat Summary

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

5.2.6.1 Primary Threats

Theft of Service. Attacks are:

- **Subscription Fraud.** This attack is prevalent in 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 the 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, an insider may still compromise it. 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 cryptanalysis.** 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 those for bearer channel privacy and are not described here:

- Simple snooping

- MTA clones

- Protocol manipulation

- Off-line cryptanalysis

- Service disruption

5.2.6.2 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.

5.3 Security Architecture

5.3.1 Overview of Security Interfaces

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

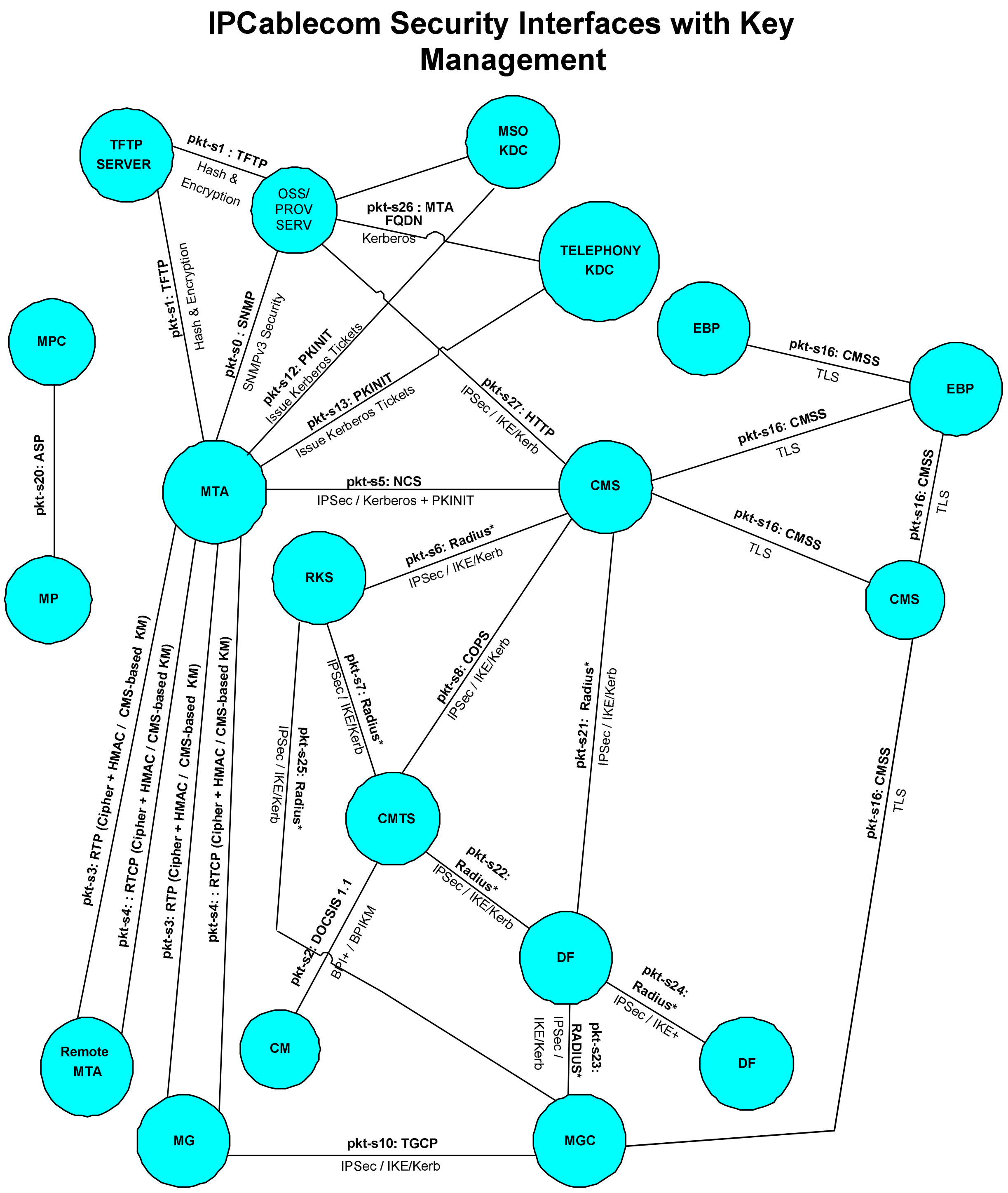

IPCablecom Security Interfaces with Key Management
TFTP SERVER
MSO KDC
OSS/ PROV SERV
TELEPHONY KDC
EBP
EBP
MPC
MP
MTA
CMS
CMS
RKS
CMTS
DF
DF
CM
Remote MTA
MG
MGC
pkt-s1 : TFTP Hash & Encryption
pkt-s1: TFTP Hash & Encryption
pkt-s26 : MTA FQDN Kerberos
pkt-s0 : SNMP SNMPv3 Security
pkt-s12: PKINIT Issue Kerberos Tickets
pkt-s13: PKINIT Issue Kerberos Tickets
pkt-s27: HTTP IPSec / IKE/Kerb
pkt-s20: ASP
pkt-s16: CMSS TLS
pkt-s16: CMSS TLS
pkt-s16: CMSS TLS
pkt-s16: CMSS TLS
pkt-s16: CMSS TLS
pkt-s5: NCS IPSec / Kerberos + PKINIT
pkt-s6: Radius* IPSec / IKE/Kerb
pkt-s7: Radius* IPSec/ IKE/Kerb
pkt-s8: COPS IPSec / IKE/Kerb
pkt-s21: Radius* IPSec / IKE/Kerb
pkt-s25: Radius* IPSec / IKE/Kerb
pkt-s3: RTP (Cipher + HMAC / CMS-based KM)
pkt-s4: : RTCP (Cipher + HMAC / CMS-based KM)
pkt-s3: RTP (Cipher + HMAC / CMS-based KM)
pkt-s4: : RTCP (Cipher + HMAC / CMS-based KM)
pkt-s2: DOCSIS 1.1 BPI+ / BPIKM
pkt-s22: Radius* IPSec / IKE/Kerb
pkt-s24: Radius* IPSec / IKE+
pkt-s23: RADIUS* IPSec / IKE/Kerb
pkt-s10: TGCP IPSec / IKE/Kerb

Figure 3: IPCablecom Security Interfaces with Key-Management

In figure 3, each interface label is of the form:

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

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

The following abbreviations are used in figure 3:

IKE/Kerb	IKE (with pre-shared keys or X.509 certificates) or Kerberos
IKE+	IKE with X.509 certificates
CMS-based KM	Keys randomly generated and exchanged inside signalling messages
RADIUS*	Event messages, which use the RADIUS format as defined by [13]

Table 1 briefly describes each of the interfaces shown in figure 3:

Table 1: IPCablecom Security Interfaces Table

Interface	Components	Description
pkt-s0	MTA - PS/OSS	Immediately after the DHCP sequence in the Secure Provisioning Flow, the MTA performs Kerberos-based key management with the Provisioning Server to establish SNMPv3 keys. The MTA bypasses Kerberized SNMPv3 and uses SNMPv2c in the Basic and Hybrid Flows.
pkt-s1	MTA - TFTP	MTA Configuration file download. When the Provisioning Server in the Secure Provisioning Flow sends an SNMP Set command to the MTA, it includes both the configuration name and the hash of the file. Later, when the MTA downloads the file, it authenticates the configuration file using the hash value. The configuration file may be optionally encrypted.
pkt-s2	CM - CMTS	DOCSIS® 1.1: This interface should be secured with BPI+ using BPI Key Management. BPI+ privacy is provided on the HFC link.
pkt-s3	MTA - MTA MTA - MG	RTP: End-to-end media packets between two MTAs, or between MTA and MG. RTP packets are encrypted directly with the chosen cipher. Message integrity is optionally provided by an MMH MAC. Keys are randomly generated, and exchanged by the two endpoints inside the signalling messages via the CMS or other application server.
pkt-s4	MTA - MTA MTA - MG	RTCP: RTCP control protocol for RTP. Message integrity and encryption by selected cipher. The RTCP keys are derived using the same secret negotiated during the RTP key management. No additional key management messages are needed or utilized.
pkt-s5	MTA - CMS	NCS: Message integrity and privacy via IPsec. Key management is with Kerberos with PKINIT (public key initial authentication) extension.
pkt-s6	RKS - CMS	RADIUS:IPsec is used for both message integrity and privacy. Key management is IKE or Kerberos.
pkt-s7	RKS - CMTS	RADIUS: IPsec is used for both message integrity and privacy. Key management is IKE or Kerberos.
pkt-s8	CMS - CMTS	COPS: COPS protocol between the GC and the CMTS, used to download QoS authorization to the CMTS. IPsec is used for message integrity, as well as privacy. Key management is IKE or Kerberos.
pkt-s9		This interface has been removed from the IPCablecom architecture.
pkt-s10	MGC - MG	TGCP: IPCablecom interface to the PSTN Media Gateway. IPsec is used for both message integrity and privacy. Key management is IKE or Kerberos.
pkt-s11		This interface has been removed from the IPCablecom architecture.
pkt-s12	MTA - MSO KDC	PKINIT: An AS-REQ message is sent to the KDC with public-key cryptography used for authentication. The KDC verifies the certificate and issues either a service ticket or a ticket granting ticket (TGT), depending on the contents of the AS Request. The AS Reply returned by the KDC contains a certificate chain and a digital signature that are used by the MTA to authenticate this message. In the case that the KDC returns a TGT, the MTA then sends a TGS Request to the KDC to which the KDC replies with a TGS Reply containing a service ticket. The TGS Request/Reply messages are authenticated using a symmetric session key inside the TGT.
pkt-s13	MTA - Telephony KDC	PKINIT: See pkt-s12 above.
pkt-s14		This interface has been removed from the IPCablecom architecture.
pkt-s15		This interface has been removed from the IPCablecom architecture.
pkt-s16	CMS - CMS CMS - MGC CMS - EBP EBP - EBP	SIP: TLS is used for both message integrity and privacy. Certificates are used for mutual authentication during the TLS handshake.
pkt-s17		This interface has been removed from the IPCablecom architecture.
pkt-s18		This interface has been removed from the IPCablecom architecture.
pkt-s19		This interface has been removed from the IPCablecom architecture.

Interface	Components	Description
pkt-s20	MPC - MP	ASP: IPsec is used for both message integrity and privacy. Key management is IKE or Kerberos.
pkt-s21	DF - CMS	RADIUS: IPsec is used for both message integrity and privacy. Key management is IKE or Kerberos.
pkt-s22	DF - CMTS	RADIUS: IPsec is used for both message integrity and privacy. Key management is IKE or Kerberos.
pkt-s23	DF - MGC	RADIUS: IPsec is used for both message integrity and privacy. Key management is IKE or Kerberos.
pkt-s24	DF - DF	RADIUS: IPsec is used for both message integrity and privacy. Key management is IKE+.
pkt-s25	RKS - MGC	RADIUS: IPsec is used for both message integrity and privacy. Key management is IKE or Kerberos.
pkt-s26	OSS/Prov Serv - MSO KDC OSS/Prov Serv - Telephony KDC	The KDC uses Kerberos to map the MTA's MAC address to its FQDN for the purpose of authenticating the MTA before issuing it a ticket.
pkt-s27	CMS-PS/OSS	HTTP: IPsec is used for both message integrity and privacy. Key management is IKE or Kerberos.
pkt-s28		This interface has been removed from the IPCablecom architecture.

5.3.2 Security Assumptions

5.3.2.1 BPI+ CMTS Downstream Messages Are Trusted

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

Most messages secured in the present document either move over the shared IP network in addition to the DOCSIS® path, or do not go over DOCSIS® at all.

In one case - the case of DOCSIS® QoS messages exchanged between the CMTS and the CM - this assumption does not apply. Although DOCSIS® QoS messages (both upstream and downstream) include an integrity check, the corresponding (BPI+) key management does not authenticate the identity of the CMTS. The CM is unable to cryptographically know that the network element it has connected to is the true CMTS for that network. However, even if a CMTS could be impersonated, it would allow only limited denial-of-service attacks. This vulnerability is not considered to be worth the effort and the expense of impersonating a CMTS.

5.3.2.2 Non-Repudiation Not Supported

Non-repudiation, in the present document, means that an originator of a message cannot deny that he or she sent that message. In this voice communications architecture, non-repudiation is not supported for most messages, with the exception of the top key management layer. This decision was based on the performance penalty incurred with each public key operation. The most important use for non-repudiation would have been during communications setup - to prove that a particular party had initiated that particular communication. However, due to very strict requirements on the setup time, it is not possible to perform public key operations for each communication.

5.3.2.3 Root Private Key Compromise Protection

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. However, with proper safeguards, the probability of this happening is very low, to the point that the risk of a root private key compromise occurring is outweighed by the benefits of this architecture.

The corresponding Root Public Key is stored as a read-only parameter in many components of this architecture. Once the Root Private Key has been compromised, each manufacturer's certificate would have to 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.

5.3.2.4 Limited Prevention of Denial-of-Service Attacks

The present document does not attempt to address all or even most 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 on-going 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 setup 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 the 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.

5.3.3 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 the 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 are described in clause 5.2.

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 BPI+ security that is assumed on the HFC DOCSIS® links. The goal of this security specification is to address threats that are relevant to this voice communications system.

5.3.3.1 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.

5.3.3.2 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. This would be done in the privacy of the customer's home.

The MTA communicates with the CMTS over the shared DOCSIS® 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:
  - Accessing basic service and any enhanced features in the name of another user's account.
  - Violating privacy of the owner of the compromised MTA that does not know that the keys were stolen.

- Identity fraud.
- An MTA running a bad code image that disrupts communications made by other MTAs or degrades network performance.

5.3.3.3 CMTS

The CMTS communicates both over the DOCSIS® path and over the shared IP network. When the CMTS sends downstream messages over the DOCSIS® path, it is assumed that a perpetrator cannot modify them or impersonate the CMTS. BPI+ over that path provides privacy.

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

While the CMTS, 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 CMTS should be addressed by cryptographic authentication and authorization of the CMTS operators, as well as by physical and procedural security, which are all out of the scope of the IPCablecom specifications.

A CMTS is responsible for:

- Reporting billing-related statistics to the RKS
- QoS allocation for MTAs over the DOCSIS® path
- Implementation of BPI+ (MAC layer security) and corresponding key management

A compromise of a CMTS may result in:

- Service theft by reporting invalid information to the RKS
- Unauthorized levels of QoS
- Loss of privacy, since the CMTS holds BPI+ 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

5.3.3.4 Voice Communications Network Servers are Untrusted Network Elements

Application 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.

5.3.3.4.1 CMS

The Call Management Server is responsible for:

- Authorizing individual voice communications by subscribers
- QoS allocation

- Initializing the billing information in the CMTS
- Distributing per communication keys for MTA-MTA signalling, bearer channel, and DQoS messages on the MTA-CMTS and CMTS-CMTS 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
    - Uploading invalid or wrong billing information to the CMTS
    - Combination of both of the above
    - Loss of privacy, since the CMS distributes bearer channel keys
    - Unauthorized allocation of QoS
    - Unauthorized disclosure of customer identity, location (e.g. IP address), communication patterns, and a list of services to which the customer subscribes

5.3.3.4.2 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 identity, personal information, service usage patterns, and a list of services to which the customer subscribes

5.3.3.4.3 OSS, DHCP and 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 the MTA completely inoperable
    - Degrade voice communications performance on the IPCablecom or HFC network
    - Configure 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

5.3.3.5 PSTN Gateways

5.3.3.5.1 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

5.3.3.5.2 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 identity, location (e.g. IP address), usage patterns and a list of services to which the customer subscribes

6 Security Mechanisms

Unless explicitly stated otherwise, the following requirements apply to messages described by the present document:

1) ASN.1 encoded messages and objects must conform to the Distinguished Encoding Rules [36].
2) FQDNs used as components of principal names and principal identifiers must be rendered in lower case.
3) FQDNs must not include the root domain (i.e. they must not include a trailing dot).
4) All Kerberos messages in IPCablecom must utilize only UDP/IP.

6.1 IPsec

6.1.1 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, IPsec ESP and IPsec AH, as specified in [19].

IPsec ESP provides 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 [20], since authentication of the IP header does not significantly improve security within the IPCablecom architecture.

Each protocol supports two modes of use: transport mode and tunnel mode. IPCablecom only utilizes IPsec ESP transport mode. For more detail on IPsec and these two modes, refer to [19]. Note that in [19], all implementations of ESP are required to support the concept of Security Associations (SAs). [19] also provides a general model for processing IP traffic relative to SAs. Although particular IPsec implementations need not follow the details of this general model, the external behaviour of any IPsec implementation must match the external behaviour of the general model. This ensures that components do not accept traffic from unknown addresses and do not send or accept traffic without security (when security is required). IPCablecom components that implement IPsec are expected to provide behaviour that matches the general model described in [19].

6.1.2 IPCablecom Profile for IPsec ESP (Transport Mode)

6.1.2.1 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 [21]. Within IPCablecom, the same Transform Identifiers are used by all IPsec key management protocols: IKE, Kerberos and application layer (embedded in IP signalling messages).

Table 2 describes the IPsec Transform Identifiers (all of which use the CBC mode specified in [22]) supported by IPCablecom.

Table 2: IPsec ESP Transform Identifiers

Transform ID	Value (Hex)	Key Size (in bits)	Must Support	Description
ESP_3DES	0x03	192	yes	3-DES in CBC mode.
ESP_RC5	0x04	128	no	RC5 in CBC mode
ESP_IDEA	0x05	128	no	IDEA in CBC mode
ESP_CAST	0x06	128	no	CAST in CBC mode
ESP_BLOWFISH	0x07	128	no	BLOWFISH in CBC mode
ESP_NULL	0x0B	0	yes	Encryption turned off
ESP_AES	0x0C	128	no	AES-128 in CBC mode with 128-bit block size

The ESP_3DES and ESP_NULL Transform IDs must be supported. ESP_AES is included as an optional encryption algorithm. For all of the above transforms, the CBC Initialization Vector (IV) is carried in the clear inside each ESP packet payload [22]. AES-128 [35] must be used in CBC mode with a 128-bit block size and a randomly generated Initialization Vector (IV). AES-128 requires 10 rounds of cryptographic operations [35].

IKE allows 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.

6.1.2.2 IPsec ESP Authentication Algorithms

The 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 [21]. Within IPCablecom, the same Authentication Algorithms are used by all IPsec key management protocols: IKE, Kerberos and application layer (embedded in IP signalling messages).

IPCablecom supports the following IPsec Authentication Algorithms:

Table 3: IPsec Authentication Algorithms

Authentication Algorithm	Value (Hex)	Key Size (in bits)	Must Support	Description
HMAC-MD5-96	0x01	128	yes (also required by [21])	First 12 bytes of the HMAC-MD5 as described in [37]
HMAC-SHA-1-96	0x02	160	yes	First 12 bytes of the HMAC-SHA1 as described in [23]

The HMAC-MD5-96 and HMAC-SHA-1-96 authentication algorithms must be supported.

6.1.2.3 Replay Protection

In general, IPsec provides an optional replay-protection service (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 the anti-replay service is turned on, an IPsec sequence number cannot overflow and roll over to 0. Before that happens, a new Security Association must be created as specified in [20].

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.

6.1.2.4 Key Management Requirements

Within IPCablecom, IPsec is used on a number of different interfaces with different security and performance requirements. Because of this, several different key management protocols have been chosen for different IPCablecom interfaces. On some interfaces it is IKE (see clause 6.2), on other interfaces it is Kerberos/PKINIT (see clause 6.4).

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 (SAs). IPsec Security Associations must be automatically established or re-established as required. This implies that 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 SA is about to expire or there is no SA on a particular interface).

In addition, some network elements are required to run multiple key management protocols. In particular, the Application Server (such as a CMS) and the MTA must support multiple key management protocols. The MTA must support Kerberos/PKINIT on the MTA-CMS signalling interface. IKE must be supported on the CMS-CMTS and CMS-RKS interfaces.

The PF_KEY interface (see [29]) 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 signals the event to all registered key-management applications. Based on the Identity Extension associated with that Security Association, each key-management application decides if it should handle the event.

6.2 Internet Key Exchange (IKE)

6.2.1 Overview

IPCablecom utilizes IKE as one of the key management protocols for IPsec [24]. It is utilized on interfaces where:

- There is not a very large number of connections
- The endpoints on each connection know about each other's identity in advance

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

IKE is a peer-to-peer key management protocol. It consists of 2 phases. In the first phase, a shared secret is negotiated via a Diffie-Hellman key exchange. It is then used to authenticate the second IKE phase. The second phase negotiates another secret, used to derive keys for the IPsec ESP protocol.

6.2.2 IPCablecom Profile for IKE

6.2.2.1 First IKE Phase

There are several modes defined for authentication during the first IKE phase.

6.2.2.1.1 IKE Authentication with Signatures

In this mode, both peers must be 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 [24]).

6.2.2.1.2 IKE Authentication with Public-Key Encryption

IPCablecom must not utilize this IKE authentication with public key encryption. In order to perform this mode of IKE authentication, the initiator must already have the responder's public key, which is not supported by IPCablecom.

6.2.2.1.3 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.

When using pre-shared keys, the strength of the system is dependent upon the strength of the shared secret. The goal is to keep the shared secret from being the weak link in the chain of security. This implies that the shared secret needs to contain as much entropy (randomness) as the cipher being used. In other words, the shared secret should have at least 128 bits to 160 bits of entropy. This means if the shared secret is just a string of random 8-bit bytes, then of the key can be 16 bytes to 20 bytes. If the shared secret is derived from a passphrase that is a string of random alpha-numerics (a-zA-Z0-9/+), then it should be at least 22 to 27 characters. This is because there are only 64 characters (6 bits) instead of 256 characters (8 bits) per 8-bit byte, which implies an expansion of 4/3 the length for the same amount of entropy. Both random 8-bit bytes and random 6-bit bytes assume truly random numbers. If there is any structure in the password/passphrase, like deriving from English, then even longer passphrases are necessary. A passphrase composed of English would need on the order of 60 to 100 characters, depending on mixing of case. Using English passphrases (or any language, for that matter) creates the problem that, if an attacker knows the language of the passphrase then they have less space to search. It is less random. This implies fewer bits of entropy per character, so a longer passphrase is required to maintain the same level of entropy.

6.2.2.2 Second IKE Phase

In the second IKE phase, an IPsec ESP SA is established, including the IPsec ESP keys and ciphersuites. It is possible to establish multiple Security Associations with a single second-phase IKE exchange.

First, a shared second phase secret is established, and then all the IPsec keying material is derived from it using the one-way function specified in [24].

The second-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 second IKE phase in order to avoid the associated performance penalties.

The second IKE phase is authenticated using a shared secret that was established in the first phase. Supported authentication algorithms are the same as those specified for IPsec in clause 6.1.2.2.

6.2.2.3 Encryption Algorithms for IKE Exchanges

Both phase 1 and phase 2 IKE exchanges include some symmetrically-encrypted messages. The encryption algorithms supported as part of the IPCablecom Profile for IKE must be the same algorithms identified in the IPCablecom profile for IPsec ESP in table 2 of clause 6.1.2.1.

6.2.2.4 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 [24]. The use of Diffie-Hellman groups within IPCablecom IKE is identical to that specified in [24]: the first group must be supported and the remaining groups should be supported. Note that this is different from the requirements pertaining to the IPCablecom use of groups in PKINIT described in clause 6.4.2.1.1. Annex E provides details of the first and second Oakley groups.

6.2.2.5 Security Association Renegotiation

Renegotiation or rekeying of an IKE Security Association (SA) is triggered by the end of its lifetime as measured in elapsed time or number of kilobytes of data protected by the SA. Each peer should transition from the old SA to the new SA the same way to avoid interoperability problems. After successful IKE phase 1 ISAKMP SA renegotiation both peers must use the new SA when sending traffic and be able to receive traffic on the new SA. After successful IKE phase 2 IPsec SA renegotiation both peers must use the new outbound SA when sending traffic and be able to receive traffic on the new inbound SA.

6.3 SNMPv3

Any mention of SNMP in the present document without a specific reference to the SNMP protocol version must be interpreted as SNMPv3.

IPCablecom supports use of SNMPv2c coexistence for network management operations for devices provisioned under the Basic Flow or the Hybrid Flow. It also supports the SNMPv3/v2c coexistence for network management operations when the device is provisioned under the Secure Flow. Refer to the provisioning specification [4] for the use of SNMP coexistence in IPCablecom.

For any interface within the IPCablecom architecture utilizing SNMPv3, SNMPv3 authentication must be turned on at all times and SNMPv3 privacy may also be utilized.

In order to establish SNMPv3 keys, all IPCablecom SNMP interfaces should utilize Kerberized SNMPv3 key management (as specified in clause 6.5.4). In addition, SNMPv3 key management techniques specified in [28] may also be used.

6.3.1 SNMPv3 Transform Identifiers

The SNMPv3 Transform Identifier (1 byte) is used by Kerberized key management to negotiate an encryption algorithm for use by SNMPv3.

For IPCablecom, the following SNMPv3 Transform Identifiers are supported.

Table 4: SNMPv3 Transform Identifiers

Transform ID	Value (Hex)	Key Size (in bits)	Must be Supported	Description
SNMPv3_DES	0x21	128	yes	DES in CBC mode. The first 64 bits are used as the DES Key and the remaining 64 are used as the pre-IV as described in [28]
SNMPv3_NULL	0x20	0	yes	Encryption turned off

The SNMPv3_DES and the SNMPv3_NULL Transform IDs must be supported. The DES encryption transform for SNMPv3 is specified in [28]. Note that DES encryption does not provide strong privacy but is currently the only encryption algorithm specified by the SNMPv3 standard.

6.3.2 SNMPv3 Authentication Algorithms

SNMPv3 Authentication Algorithm (1 byte) is used by Kerberized key management to negotiate an SNMPv3 message authentication algorithm.

For IPCablecom, the following SNMPv3 Authentication Algorithms are supported (both of which are specified in [28]).

Table 5: SNMPv3 Authentication Algorithms

Authentication Algorithm	Value (Hex)	Key Size (in bits)	Must be Supported	Description
SNMPv3_HMAC-MD5	0x21	128	yes (also required by [28])	MD5 HMAC
SNMPv3_HMAC-SHA-1	0x22	160	no (should be supported)	SHA-1 HMAC

The SNMPv3_HMAC-MD5 Authentication Algorithm must be supported. The SNMPv3_HMAC-SHA-1 Authentication Algorithm should be supported.

6.4 Kerberos / PKINIT

6.4.1 Overview

IPCablecom utilizes the concept of Kerberized IPsec for signalling between an Application Server, such as the CMS, and the MTA. This refers to the ability to create IPsec Security Associations using keys derived from the subkeys exchanged using the Kerberos AP Request/AP Reply messages. On this interface, Kerberos (annex B) is utilized with the PKINIT public key extension (also see annex C).

Kerberized IPsec consists of three distinct phases:

1) A client should obtain a TGT (Ticket Granting Ticket) from the KDC (Key Distribution Centre). Once the client obtains the TGT, it must use the TGT in the subsequent phase to authenticate to the KDC and obtain a ticket for the specific Application Server, e.g. a CMS.

   In Kerberos, tickets are symmetric authentication tokens encrypted with a particular server's key. (For a TGT, the server is the KDC.) Tickets are used to authenticate a client to a server. A PKI equivalent of a ticket would be an X.509 certificate. In addition to authentication, a ticket is used to establish a session key between a client and a server, where the session key is contained in the ticket.

   The logical function within the KDC that is responsible for issuing TGTs is referred to as an Authentication Server or AS.

2) A client obtains a ticket from the KDC for a specific Application Server. In this phase, a client can authenticate with a TGT obtained in the previous phase. A client can also authenticate to the KDC directly using a digital certificate or a password-derived key, bypassing phase 1.

   The logical function within the KDC that is responsible for issuing Application Server tickets based on a TGT is referred to as the Ticket Granting Server - TGS. When the TGT is bypassed, it is the Authentication Server that issues the Application Server tickets.

3) A client utilizes the ticket obtained in the previous phase to establish a pair of Security Parameters (one to send and one to receive) with the server. This is the only key management phase that is not already specified in an IETF standard. The previous two phases are part of standard Kerberos, while this phase defines new messages that tie together Kerberos key management and IPsec.

Figure 4 illustrates the three phases of Kerberos-based key management for IPsec.

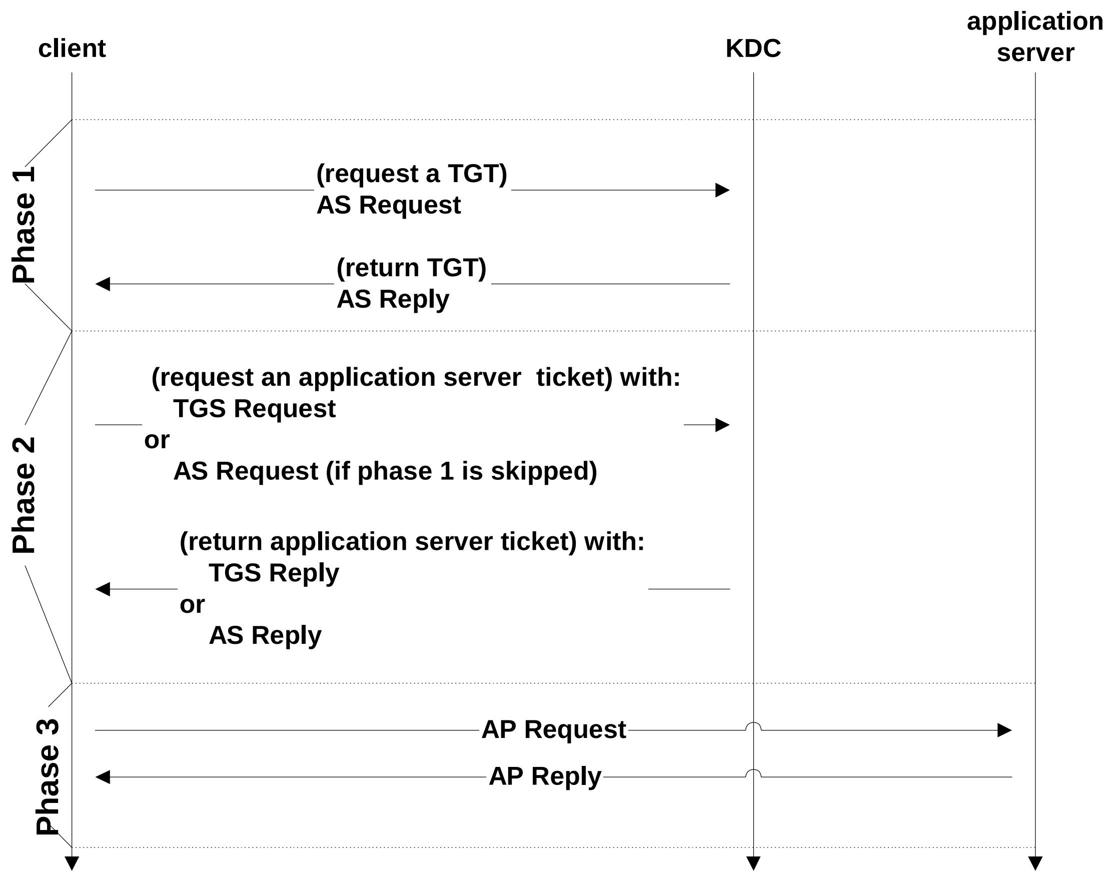

client
KDC
application server
Phase 1
(request a TGT) AS Request
(return TGT) AS Reply
Phase 2
(request an application server ticket) with: TGS Request or AS Request (if phase 1 is skipped)
(return application server ticket) with: TGS Reply or AS Reply
Phase 3
AP Request
AP Reply

Figure 4: Kerberos-Based Key Management for IPsec

During the AS Request / AS Reply exchange (that can occur in either phase 1 or phase 2), the client and the KDC perform mutual authentication. In standard Kerberos, a client key that is shared with the KDC is used for this authentication (see clause 6.4.4). The same AS Request / AS Reply exchange may also be authenticated with digital signatures and certificates when the PKINIT public key extension is used (see clause 6.4.2). Both the TGT and the Application Server tickets used within IPCablecom have a relatively long lifetime (days or weeks). This is acceptable as 3-DES, a reasonably strong symmetric algorithm, is required by IPCablecom.

IPCablecom utilizes the concept of a TGT (Ticket Granting Ticket), used to authenticate subsequent requests for Application Server tickets. The use of a TGT has two main advantages:

- It limits the exposure of the relatively long-term client key (that is in some cases reused as the service key). This consideration does not apply to clients that use PKINIT.
- It reduces the number of public key operations that are required for PKINIT clients.

The Application Server ticket contains a symmetric session key, which must be used in phase 3 to establish a set of keys for the IPsec ESP protocol. The keys used by IPsec must expire after a configurable time-out period (e.g. 10 minutes). Normally, the same Application Server ticket should be used to automatically establish a new IPsec SA. However, there are instances where it is desirable to drop IPsec sessions after a Security Association time out and establish them on-demand later. This allows for improved system scalability, since an application server (e.g. CMS) does not need to maintain a SA for every client (e.g. MTA) that it controls. It is also possible that a group of application servers (e.g. CMS cluster) may control the same subset of clients (e.g. MTAs) for load balancing. In this case, the MTA is not required 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 an expiration of the old one and how to establish IPsec SAs on-demand, when a signalling message needs to be sent.

IPCablecom also utilizes the Kerberos protocol to establish SNMPv3 keys between the MTAs and the Provisioning Server. Kerberized SNMPv3 key management is very similar to the Kerberized IPsec key management and consists of the same phases that were explained above for Kerberized IPsec. Each MTA again utilizes the PKINIT extension to Kerberos to authenticate itself to the KDC with X.509 certificates.

Once an MTA obtains its service ticket for the Provisioning Server, it utilizes the same protocol that is used for Kerberized IPsec to authenticate itself to the Provisioning Server and to generate SNMPv3 keys. The key management protocol is specified to allow application-specific data that has different profiles for SNMPv3 and IPsec. The only exception is the Rekey exchange that is specified for IPsec in order to optimize the MTA hand-off between the members of a CMS cluster. The Rekey exchange is not utilized for SNMPv3 key management.

A recipient of any Kerberos message that does not fully comply with the IPCablecom requirements must reject the message.

6.4.1.1 Kerberos Ticket Storage

Kerberos clients that store tickets in persistent storage will be able to re-use the same Kerberos ticket after a reboot. In the event that PKINIT is used, this avoids the need to perform public key operations.

A Kerberos client must not obtain a new TGT upon reboot if it possesses a valid service ticket.

An MTA must store the Provisioning Server service ticket in persistent storage. An MTA must be capable of storing a minimum of pktcMtaDevEndPntCount+1 CMS service tickets in persistent storage, where pktcMtaDevEndPntCount is the MIB object specifying the number of physical endpoints on the MTA. An MTA must store all CMS service tickets that correspond to active endpoints. This means that an MTA that reaches the maximum number of CMS service tickets that can be stored in persistent storage will not over-write CMS service tickets that correspond to active endpoints.

Kerberos clients other than MTAs should retain service tickets in persistent storage.

Note that Kerberos clients will need to store additional information in order to use and validate the ticket, such as the session key information, the client IP address, and the ticket validity period. Refer to clause 7.1 for additional information on reusing stored tickets.

6.4.2 PKINIT Exchange

Figure 5 illustrates how a client may use PKINIT to either obtain a TGT (phase 1) or a Kerberos ticket for an Application Server (phase 2).

The PKINIT Request is 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 plug in is specified in annex B.

In this clause, the PKINIT client is referred to as an MTA, as it is currently the only IPCablecom element that authenticates itself to the KDC with the PKINIT protocol. If in the future other IPCablecom elements will also utilize the PKINIT protocol, the same specifications will apply. IPCablecom use of the AS Request / AS Reply exchange without PKINIT is covered in clause 6.4.3.

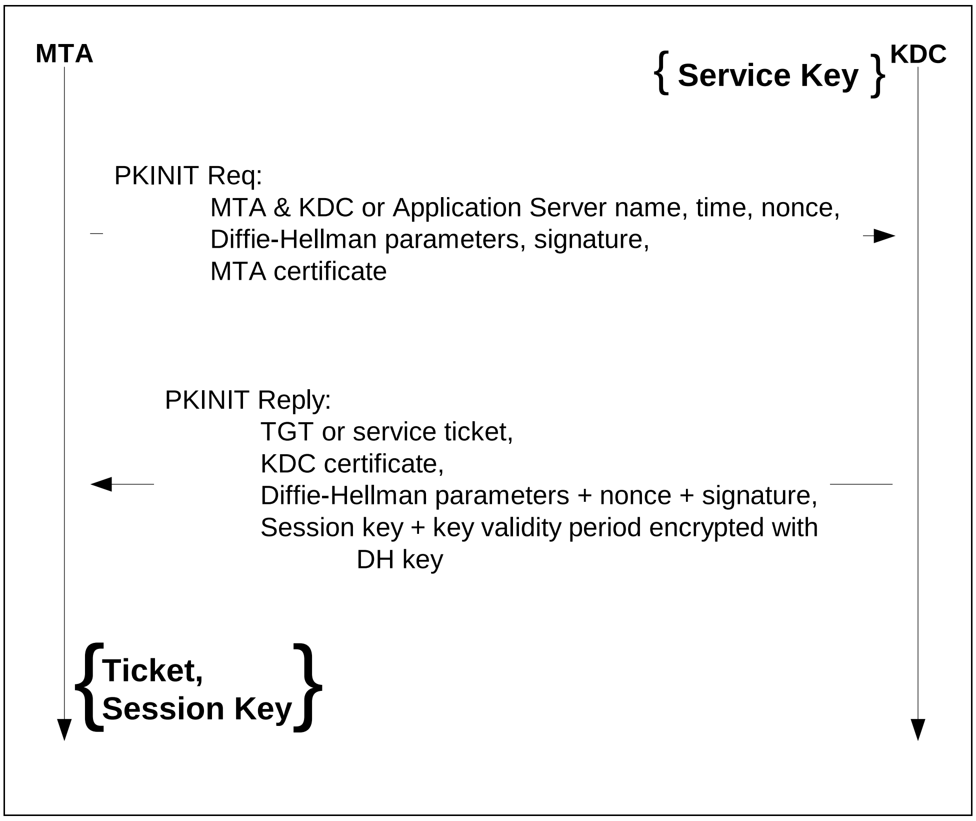

MTA
KDC
{ Service Key }
PKINIT Req:
MTA & KDC or Application Server name, time, nonce,
Diffie-Hellman parameters, signature,
MTA certificate
PKINIT Reply:
TGT or service ticket,
KDC certificate,
Diffie-Hellman parameters + nonce + signature,
Session key + key validity period encrypted with
DH key
{Ticket,
Session Key}

Figure 5: PKINIT Exchange

Figure 5 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 (see annex B). For the format used in IPCablecom, see clause 6.4.7.
- KDC or Application Server (Kerberos principal) name - found in the KDC-REQ-BODY Kerberos structure (see annex B). For the format used in IPCablecom, see clause 6.4.6.
- Time - found in the PKAuthenticator structure, specified by PKINIT (annex C).
- Nonce - found in the PKAuthenticator structure, specified by PKINIT (annex C). There is also a second nonce in the KDC-REQ-BODY Kerberos structure.
- Diffie-Hellman parameters, signature and MTA certificate - these are all specified by PKINIT (annex C) and their use in IPCablecom is specified in clause 6.4.2.1.1. Annex E provides details of the first and second Oakley groups.

PKINIT Reply

- TGT or Application Server Ticket - found in the KDC-REP Kerberos structure (see annex B).
- KDC Certificate, Diffie-Hellman parameters, signature - these are all specified by PKINIT (see annex C) and their use in IPCablecom is specified in clause 6.4.2.1.2. Annex E provides details of the first and second Oakley groups.
- Nonce - found in the KdcDHKeyInfo structure, specified by PKINIT (annex C). 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 (see annex B). 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 (see annex B).

In figure 5, 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 server via the server's ticket, where the application server may be the KDC (in which case the ticket is the TGT).

6.4.2.1 PKINIT Profile for IPCablecom

A particular MTA implementation must utilize the PKINIT exchange to either obtain Application Server tickets directly, or obtain a TGT first and then use the TGT to obtain Application Server tickets. An MTA implementation may also support both uses of PKINIT, where the decision to get a TGT first or not is local to the MTA and is dependent on a particular MTA implementation. On the other hand, the KDC must be capable of processing PKINIT requests for both a TGT and for Application Server tickets.

The PKINIT exchange occurs independent of the signalling protocol, based on the current Ticket Expiration Time ($\text{Ticket}_{\text{EXP}}$) and on the PKINIT Grace Period ($\text{PKINIT}_{\text{GP}}$). If the PKINIT client is an MTA and the ticket it currently possesses corresponds to the Provisioning Server in the MIB, a KDC for a REALM that currently exists in the REALM table, or a CMS that currently exists in the CMS table, the MTA must initiate the PKINIT exchange at the time: $\text{Ticket}_{\text{EXP}}$ - $\text{PKINIT}_{\text{GP.}}$ If the PKINIT client is an MTA and the ticket it currently possesses does not correspond to the Provisioning Server in the MIB, a KDC for a REALM that currently exists in the REALM table, or a CMS that currently exists in the CMS table, the MTA must not initiate a PKINIT exchange. On the interfaces where $\text{PKINIT}_{\text{GP}}$ is not defined, the MTA should perform PKINIT exchanges on-demand.

In the case where PKINIT is used to obtain an Application Server ticket directly, the use of the grace period accounts for a possible clock skew between the MTA and the CMS or other application server. If the MTA is late with the PKINIT exchange, it still has until $\text{Ticket}_{\text{EXP}}$ before the Application Server starts rejecting the ticket. Similarly, if PKINIT is used to obtain a TGT the grace period accounts for a possible clock skew between the MTA and the KDC.

The PKINIT exchange stops after the MTA obtains a new ticket, and therefore does not affect existing security parameters between the MTA and the CMS or other application server. Synchronizing the PKINIT exchange with the AP Request/Reply exchange is not required as long as the AS Request/Reply exchange results in a valid, non-expired Kerberos ticket.

The PKINIT Request/Reply messages contain public key certificates, which make them longer than a normal size of a UDP packet. In this case, large UDP packets must be sent using IP fragmentation.

A KDC server should be implemented on a separate host, independent of the Application Server. This would mean, that frequent PKINIT operations from some MTAs will not affect the performance of any of the application servers 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 $\text{PKINIT}_{\text{GP}}$ must be at least 15 minutes.

6.4.2.1.1 PKINIT Request

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

```
PA-PK-AS-REQ ::= SEQUENCE {
  signedAuthPack      [0] ContentInfo,
  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. It is defined in Cryptographic Message Syntax, identified by the SignedData OID:{iso(1) member-body(2) us(840) rsadsi(113549) pkcs(1) pkcs7(7) 2}. SignedData is defined as:

```
SignedData ::= SEQUENCE {
  version                 CMSVersion,
  digestAlgorithms        DigestAlgorithmIdentifiers,
  encapContentInfo        EncapsulatedContentInfo,
  certificates        [0] IMPLICIT CertificateSet OPTIONAL,
  crls                [1] IMPLICIT CertificateRevocationLists
                          OPTIONAL,
  signerInfos          SignerInfos
}
```

- 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 Cryptographic Message Syntax as:

```
EncapsulatedContentInfo ::= SEQUENCE {
  eContentType     ContentType,
  eContent     [0] EXPLICIT OCTET STRING OPTIONAL
}
```

Here eContentType indicates the type of data and for PKINIT must be set to:
{iso(1) org(3) dod(6) internet(1) security(5) kerberosv5(2) pkinit(3) pkauthdata(1)}

eContent is a data structure of type AuthPack encoded inside an OCTET STRING:

```
AuthPack ::= SEQUENCE {
  pkAuthenticator   [0] PKAuthenticator,
  clientPublicValue [1] SubjectPublicKeyInfo OPTIONAL
}
```

The optional clientPublicValue parameter inside the AuthPack must always be present for IPCablecom. (This parameter specifies the client's Diffie-Hellman public value [50].)

```
PKAuthenticator ::= SEQUENCE {
  cusec      [0] INTEGER,
                 -- for replay prevention as in RFC1510
  ctime      [1] KerberosTime,
                 -- for replay prevention as in RFC1510
  nonce      [2] INTEGER,
                 -- zero only if client will accept
                 -- cached DH parameters from KDC;
                 -- must be non-zero otherwise
  pachecksum [3] Checksum
                 -- Checksum over KDC-REQ-BODY
                 -- Defined by Kerberos spec
}
```

The pachecksum field must use the Kerberos checksum type rsa-md5, a plain MD5 checksum over the KDC-REQ-BODY.

The nonce field must be non-zero, indicating that the client does not support the caching of Diffie-Hellman values and their expiration.

- certificates - required by IPCablecom. This field must contain an MTA Device Certificate and an MTA Manufacturer Certificate. This field must not contain any other certificates. All IPCablecom certificates are X.509 certificates for RSA Public keys as specified in clause 8.

- crls - must not be filled in by the MTA.

- signerInfos - must be a set with exactly one member that holds the MTA signature. This signature is a part of a SignerInfo data structure defined within the Cryptographic Message Syntax. All optional fields in this data structure must not be used in IPCablecom. The digestAlgorithm must be set to SHA-1:

  {iso(1) identified-organization(3) oiw(14) secsig(3) algorithm(2) 26}

  and the signatureAlgorithm must be set to rsaEncryption:

  {iso(1) member-body(2) us(840) rsadsi(113549) pkcs(1) pkcs-1(1) 1}

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.

The IKE specification in [24] defines Diffie-Hellman parameters as Oakley groups. Within the IPCablecom PKINIT profile the 2nd Oakley group must be supported and the 1st Oakley group may also be supported. Annex E provides details of the first and second Oakley groups.

When generating Diffie-Hellman private keys, a device must generate a key of length at least 144 bits when the first Oakley group is used and must generate a key of length at least 164 bits when the second Oakley group is used.

For further details of PKINIT, please refer to annex C.

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 must not be used within 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 must not be used within 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 must not be used within IPCablecom.

Upon receipt of a PA-PK-AS-REQ, the KDC must:

1) check the validity of the certificate chain (MTA Device Certificate, MTA Manufacturer Certificate, MTA Root Certificate)

2) check the validity of the signature in the (single) SignerInfo field

3) check the validity of the checksum in the PKAuthenticator

6.4.2.1.2 PKINIT Reply

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

```
PA-PK-AS-REP ::= CHOICE {
  dhSignedData [0] ContentInfo,
  encKeyPack   [1] ContentInfo
}
```

IPCablecom must use 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 annex B. The AS Reply is encrypted with 3-DES CBC, with a Kerberos etype value of des3-cbc-md5 (see clause 6.4.2.2). 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. This PKINIT Reply contains the KDC's Diffie-Hellman public value that is used to generate a shared secret (part of the key agreement). This shared secret is used to encrypt/decrypt the private part of the AS Reply.

- dhSignedData - dhSignedData is identified by the SignedData oid: {iso(1) member-body(2) us(840) rsadsi(113549) pkcs(1) pkcs7(7) 2}. Within SignedData (specified in clause 6.4.2.1.1):

  - 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 pkdhkeydata, where eContentType contains the following OID value: {iso(1) org(3) dod(6) internet(1) security(5) kerberosv5(2) pkinit(3) pkdhkeydata(2)}

    eContent is of type KdcDHKeyInfo (encoded inside an OCTET STRING):

```
KdcDHKeyInfo ::= SEQUENCE {
                   -- used only when utilizing Diffie-Hellman
     subjectPublicKey [0] BIT STRING,
                          -- Equals public exponent (g^a mod p)
                          -- INTEGER encoded as payload of
                          -- BIT STRING
     nonce            [1] INTEGER,
                          -- Binds response to the request
                          -- Exception: Set to zero when KDC
                          -- is using a cached DH value
```

```
  dhKeyExpiration  [2] KerberosTime OPTIONAL
                        -- Expiration time for KDC's cached
                        -- DH value
}
```

- The nonce must be the same nonce that was passed in by the client in the PKINIT Request.
- The subjectPublicKey must be the Diffie-Hellman public value generated by the KDC. The Diffie-Hellman-derived key is used to directly encrypt part of the AS Reply. The requirements on the length of the Diffie-Hellman private exponent are as defined in clause 6.4.2.1.1.
- The dhKeyExpiration must not be present as caching of Diffie-Hellman values is not permitted.

- certificates - required by IPCablecom. This field must contain a KDC certificate. If a Local System CA issued the KDC certificate, then the corresponding Local System CA Certificate must also be present. The Service Provider CA Certificate must also be present in this field. This field may contain the Service Provider Root CA certificate (refer to clause 8.2.1 for validating the Service Provider Root CA certificate if it is included in the PKINIT Reply). This field must not contain any other certificates. If the MTA is configured with a specific service provider name, it must verify that the Service Provider name is identical to the value of the OrganizationName attribute in the subjectName of the Service Provider certificate. If the Local System Certificate is present, then the MTA must verify that the Service Provider name is identical to the value of the OrganizationName attribute in the subjectName of the Local System Certificate. In addition to standard certificate verification rules specified in RFC 2459 [34], an MTA must verify that the KDC certificate includes a subjectAltName extension in the format specified in clause 8.2.3.4.1. The MTA must verify that the extension contains a valid KDC principal name and that the KDC realm in this extension is identical to the server realm name in the encrypted portion of the AS Reply message (EncKDCRepPart).
- crls - this optional field may be filled in by the KDC.
- signerInfos - must be a set with exactly one member that holds the KDC signature. This signature is a part of a SignerInfo data structure defined within the Cryptographic Message Syntax. All optional fields in this data structure must not be used in IPCablecom. The digestAlgorithm must be set to SHA-1:

  {iso(1) identified-organization(3) oiw(14) secsig(3) algorithm(2) 26}

  the signatureAlgorithm must be set to rsaEncryption:

  {iso(1) member-body(2) us(840) rsadsi(113549) pkcs(1) pkcs-1(1) 1}

Upon receipt of a PA-PK-AS-REP, the client must:

1) check the value of the nonce in the eContent field
2) check the validity of the KDC certificate
3) check the validity of the signature in the SignerInfo field

6.4.2.1.2.1 PKINIT Error Messages

In the case that a PKINIT Request is rejected, instead of a PKINIT Reply the KDC must return a Kerberos error message of type KRB_ERROR, as defined in annex C. Any error code that is defined in annex C for PKINIT may be returned.

- The KRB_ERROR must use typed-data of REQ-NONCE to bind the error message to the nonce from the KDC-REQ-BODY portion of the AS-REQ message. 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 KDC do not share a symmetric key.

When a client receives an error message from the KDC, in some cases the present document calls for the client to take some recovery steps and then send a new AS Request or TGS Request. When a client is responding to an error message, it is not a retry and must not be considered to be part of the client's back-off and retry procedure specified in clause 6.4.8. The client must reset its timers accordingly, to reflect that the new request in response to an error message is not a retry.

Although the present document calls for a KDC to return some specific error codes under certain error conditions, in the case when a KDC is repeatedly getting the same error from the same client IP address, it may at some point choose to stop sending back any further replies (errors or otherwise) to this client.

6.4.2.1.2.1.1 Clock Skew Error

When the KDC clock and the client clock are off by more than the limit for a clock skew, an error code KRB_AP_ERR_SKEW must be returned. The value for the maximum clock skew allowed by the KDC must not exceed 5 minutes. The optional client's time in the KRB_ERROR must be filled out, and the client must compute the difference (in seconds) between the two clocks based upon the client and server time contained in the KRB_ERROR message. The client should store this clock difference in non-volatile memory and must use it to adjust Kerberos timestamps in subsequent KDC request messages (AS Request and TGS Request) by adding the clock skew to its local clock value each time. The client must maintain a separate clock skew value for each realm. The clock skew values are intended for uses only within the Kerberos protocol and should not otherwise affect the value of the local clock (since a clock skew is likely to vary from realm to realm).

In the case that a KDC request fails due to a clock skew error, a client must immediately retry after adjusting the Kerberos timestamp inside the KDC Request message.

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 in which a rogue KDC attempts to fool an MTA into accepting an expired KDC certificate.

6.4.2.1.3 Pre-Authenticator for Provisioning Server Location

An AS Request sent by the MTA must include this PROV-SRV-LOCATION pre-authenticator that the KDC can use to locate the Provisioning Server.

The pre-authenticator type must be -1 (according to annex B, the negative type is used for application-specific pre-authenticators). Its ASN.1 encoding is specified as:

```
PROV-SRV-LOCATION ::= GeneralString
                      -- Provisioning Server's FQDN
```

6.4.2.2 Profile for the Kerberos AS Request / AS Reply Messages

As mentioned earlier, the PKINIT Request and Reply are pre-authenticator fields embedded into the AS Request / AS Reply messages. The IPCablecom-specific PROV-SRV-LOCATION pre-authenticator must be used in combination with PKINIT. All other pre-authenticators must not be used in combination with PKINIT.

The optional fields from, 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. The client must not set any of the KDCOptions in the AS-REQUEST, except that the DISABLE-TRANSITED-CHECK option may be set.

The MTA must include its IP address in the optional addresses field of the KDC-REQ-BODY. The KDC must verify that the addresses field in the KDC-REQ-BODY contains exactly one IP address and that it is identical to the IP address in the IP header of the AS Request. After the KDC validates the addresses field, it must include it in the caddr fields of the issued ticket and the AS Reply. The KDC must reject an AS Request that does not include the MTA's IP address. In this case the KDC must return a KDC_ERR_POLICY error code.

If a KDC receives an AS-REQ message in which any of the KDCOptions are set, except for the DISABLE-TRANSITED-CHECK option, the KDC must return an error with the error code KDC_ERR_POLICY.

In the AS Reply, key-expiration, starttime and renew-till optional fields must not be present. The session key contained in the AS-REPLY (which must be identical to the session key in the ticket) must be etype des3-cbc-md5.

The encrypted part of the AS Reply is of the type EncryptedData. The ASN.1 definition of EncryptedData that is used inside multiple Kerberos objects is missing from RFC 4120 [14], annex B. In all cases, EncryptedData must be DER-encoded with EXPLICIT tags as the following ASN.1 structure:

```
EncryptedData   ::= SEQUENCE {
       etype   [0] INTEGER,           -- EncryptionType
       kvno    [1] INTEGER OPTIONAL,  -- service key
                                        -- version number
       cipher  [2] OCTET STRING       -- ciphertext
}
```

When EncryptedData contains ciphertext that is encrypted with a service key, the 'kvno' element must be present and must identify the version of the service key that was used to encrypt the data. When EncryptedData contains ciphertext that is encrypted with a Kerberos session key or with a reply key derived from a PKINIT pre-authenticator, the 'kvno' element must not be present. This is the case for the encrypted portion of the AS Reply.

The encryption type for an encrypted portion of the AS Reply must be set to des3-cbc-md5. In order to generate the value of the 'cipher' element of the EncryptedData, the following data must be concatenated and processed in the following sequence before being encrypted with 3-DES CBC, IV=0:

- 8-byte random byte sequence, called a confounder
- An MD5 checksum, which is the MD5 hash of the concatenation of the three quantities (the confounder + sixteen NULL octets + the text to be encrypted [not including any padding])
- AS Reply part that is to be encrypted
- Random padding up to a multiple of 8

Upon receipt of an AS-REPLY, the client must check the validity of the checksum in the encrypted portion of the AS-REPLY.

6.4.2.3 Profile for Kerberos Tickets

In Kerberos Tickets, authorization-data, starttime and renew-till optional fields must not be present. The optional caddr field must be present when requested in an AS-REQUEST or when present in a TGT of a TGS Request (see annex B). The only ticket flags that are supported within IPCablecom are the INITIAL, PRE-AUTHENT and TRANSITED-POLICY-CHECKED flags. If the KDC receives any request that would otherwise cause it to set any other flag, it must return an error with the error code KDC_ERR_POLICY. The KDC must not generate tickets with any other flags set. The session key contained in the ticket (which must be identical to the session key in the AS-REPLY) must be etype des3-cbc-md5. Since the transited encoding information normally required by PKINIT (see annex B) is not used in IPCablecom, a KDC may choose to leave as a null string the 'contents' field of the TransitedEncoding portion of a ticket issued in response to a PKINIT request.

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 clause 6.4.2.2.

Upon receipt of a ticket for a service, the server must:

1) check the validity of the checksum in the encrypted portion of the ticket

2) check that the ticket has not expired

Currently, all the service keys are pre-shared using an out-of-band mechanism between the KDC and the device providing the service. In the future, IPCablecom may support a method that does not require these keys to be pre-shared.

6.4.3 Symmetric Key AS Request / AS Reply Exchange

In IPCablecom, a Kerberos client may use standard symmetric-key authentication (with a client key) during the AS Request / AS Reply exchange. Also, in IPCablecom, a client not utilizing PKINIT is, at the same time, an Application Server for which other clients might obtain tickets. This means that an IPCablecom entity may utilize the same symmetric key for both client authentication and for decrypting its service tickets.

The Kerberos AS Request / AS Reply exchange, in general, is allowed to occur with no client authentication. The client, in those cases, would authenticate itself later by proving that it is able to decrypt the AS Reply with its symmetric key and make use of the session key.

Such use of Kerberos is not acceptable within IPCablecom. This approach would allow a rogue client to continuously generate AS Requests on behalf of other clients and receive the corresponding AS Replies. Although this rogue client would be unable to decrypt each AS Reply, it will know some of the fields that it should contain. This, and the availability of the matching encrypted AS Replies, would aid an attacker in the discovery of another client's key with cryptanalysis.

Therefore, IPCablecom requires that whenever an AS Request is not using a PKINIT preauthenticator, it must instead use a different preauthenticator, of type PA-ENC-TS-ENC. This preauthenticator is specified as:

```
PA-ENC-TS-ENC ::= SEQUENCE {
  patimestamp [0] KerberosTime,
                  -- client's time
  pausec      [1] INTEGER OPTIONAL,
  pachecksum  [2] CheckSum OPTIONAL
                  -- keyed checksum of
                  -- KDC-REQ-BODY
}
```

The PA-ENC-TS-ENC preauthenticator must be encrypted with the client key using the encryption type des3-cbc-md5, as described in clause 6.4.2.2. All optional fields inside PA-ENC-TS-ENC must be present for IPCablecom. The pachecksum field must be a keyed checksum of type rsa-md5-des3. The checksum must be keyed with the client key. The checksum must be validated by the KDC.

The encrypted timestamp is used by the KDC to authenticate the client. At the same time, the timestamp inside this preauthenticator is used to prevent replays. The KDC checks for replays upon the receipt of this preauthenticator; this is similar to the checking performed by an Application Server upon receipt of an AP Request message.

If the timestamp in the PA-ENC-TS-ENC preauthenticator differs from the current KDC time by more than the acceptable clock skew then KDC must reply with a clock skew error message. The client must respond to this error message as specified in clause 6.4.2.1.2.1.1.

If the realm, target server name (e.g. the name of the KDC), along with the client name, time and microsecond fields from the PA-ENC-TS-ENC preauthenticator match any recently-seen such tuples, the KRB_AP_ERR_REPEAT error must be returned. The KDC must remember any such preauthenticator presented within acceptable clock skew period, so that a replay attempt is guaranteed to fail.

If the Application Server loses track of any authenticator presented within the acceptable clock skew period, it must reject all requests until the acceptable clock skew interval has passed.

Symmetric-key AS Request / AS Reply exchange is illustrated in figure 6.

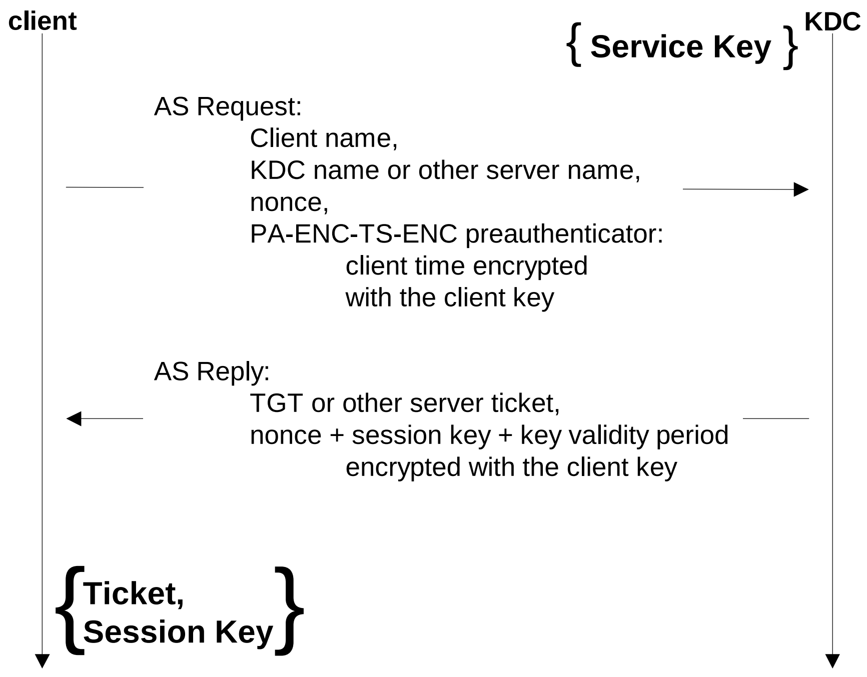

client
KDC
{ Service Key }
AS Request:
Client name,
KDC name or other server name,
nonce,
PA-ENC-TS-ENC preauthenticator:
client time encrypted
with the client key
AS Reply:
TGT or other server ticket,
nonce + session key + key validity period
encrypted with the client key
{Ticket, Session Key}

Figure 6: Symmetric-Key AS Request / AS Reply Exchange

6.4.3.1 Profile for the Symmetric Key AS Request / AS Reply Exchanges

The content of the AS Request / AS Reply messages is the same as in the case of the PKINIT preauthentication (see clause 6.4.2.1) with the exception of the type of the preauthenticator that is used.

In general, clients using a symmetric-key form of the AS Request / AS Reply exchange are not required to always possess a valid TGT or a valid Application Server ticket. A client may obtain both a TGT and Application Server tickets on-demand, as they are needed for the key management with the Application Server.

However, there may be cases where a client is required to quickly switch between servers for load balancing and the additional symmetric-key exchanges with the KDC are undesirable. In those cases, a client may be optimized to obtain tickets in advance, so that the key management would take only a single roundtrip (AP Request / AP Reply exchange.)

In the case that the KDC rejects the AS Request, it returns a KRB_ERROR message instead of the AS Reply, as specified in annex B. The KRB_ERROR must use typed-data of REQ-NONCE to bind the error message to the nonce from the AS-REQ message. This error message must include the optional e-cksum member that would contain an rsa-md5-des3 keyed checksum of the error reply, unless pre-authentication failed to prove knowledge of the shared symmetric key in which case the e-cksum must not be used.

The rsa-md5-des3 checksum must be computed as follows:

1) prepend the message with an 8-byte random byte sequence, called a confounder
2) take an MD5 hash of the result of step 1
3) prepend the hash with the same 8-byte confounder
4) take the 3DES session key from the ticket and XOR each byte with F0
5) use 3DES in CBC mode to encrypt the result of step 3, using the key in step 4 and with IV(initialization vector)=0

Once a client receives an AS Reply, it should save both the obtained ticket and the session key information (found in the enc-part member of the reply) in non-volatile memory. Thus, the client will be able to re-use the same Kerberos ticket after a reboot, avoiding the need to perform the AS Request again.

Kerberos Tickets must not be issued for a period of time that is longer than 7 days (same as for PKINIT exchanges).

Upon receipt of a KRB_ERROR that contains an e-cksum field, the recipient must verify the validity of the checksum.

6.4.4 Kerberos TGS Request / TGS Reply Exchange

In the cases where a client obtained a TGT, that TGT is then used in the TGS Request / TGS Reply exchange to obtain a specific Application Server ticket. This is part of the Kerberos standard, as it is specified in annex B.

A TGS Request includes a KRB_AP_REQ data structure (the same structure used in an AP Request: see clause 6.4.4.1). This data structure contains the TGT as well as an authenticator that is used by the client to prove the possession of the corresponding session key. The TGS Reply has the same format as an AS Reply, except that it is encrypted using a different key - the session key from the TGT.

Figure 7 illustrates the TGS Request / TGS Reply exchange.

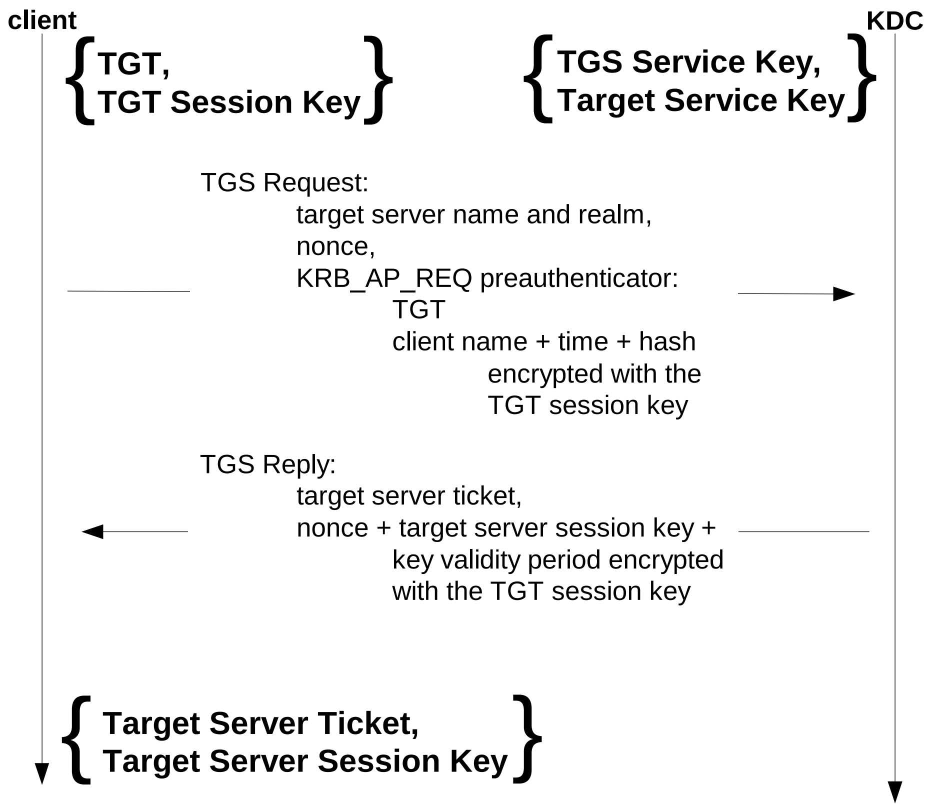


Figure 7: Kerberos TGS Request / TGS Reply Exchange

Figure 7 lists several important parameters in the TGS Request and Reply messages. These parameters are:

- TGS Request
  - Target server (principal) name and realm, nonce - found in the KDC-REQ-BODY Kerberos structure (see annex B).
  - TGS preauthenticator - found in the KDC-REQ Kerberos structure, inside the padata field (see annex B). The preauthenticator type in this case is PA-TGS-REQ.
  - KRB_AP_REQ - the value of the preauthenticator of type PA-TGS-REQ.
  - TGT - inside the KRB_AP_REQ.
  - Client name, time - inside the Kerberos Authenticator structure, which is embedded in an encrypted form in the KRB_AP_REQ.
- TGS Reply
  - Target server ticket - found in the KDC-REP Kerberos structure (see annex B).

- Target server session key, nonce, key validity period - found in the EncKDCRepPart Kerberos structure (see annex B).

In general, the TGS Request / Reply exchange may be performed on-demand - whenever an Application Server ticket is needed to establish Security Parameters. If the client is an MTA and a ticket it currently possesses corresponds to the Provisioning Server in the MIB or a CMS that currently exists in the CMS table, it must initiate the TGS Request / Reply exchange at the time: Ticket_{EXP} - TGS_{GP}. Here, Ticket_{EXP} is the expiration time of the current Application Server ticket and TGS_{GP} is the TGS Grace Period. If the client is an MTA and the ticket it currently possesses does not correspond to the Provisioning Server in the MIB or a CMS that currently exists in the CMS table, the MTA must not initiate a PKINIT exchange.

The validity of the Application Server tickets must not extend beyond the expiration time of the TGT that was used to obtain the server ticket.

6.4.4.1 TGS Request Profile

The optional padata element in the KDC-REQ data structure must consist of exactly one element - a preauthenticator of type PA-TGS-REQ. The value of this preauthenticator is the KRB_AP_REQ data structure. Within KRB_AP_REQ:

1) Options in the ap-options field must not be present.

2) The ticket is the TGT.

3) The encrypted authenticator must contain the checksum field - an MD5 checksum of the ASN.1 encoding of the KDC-REQ-BODY data structure. It must not contain any other optional fields.

4) The authenticator must be encrypted using 3-DES CBC with the following Kerberos etype value des3-cbc-md5 as specified in clause 6.4.2.2.

The optional fields from, enc-authorization-data, additional-tickets and rtime in the KDC-REQ-BODY must not be present in the TGS Request. The optional field cname should not be present. All other optional fields in the TGS Request may be present for IPCablecom. The KDC must reject a TGT that has any ticket flags set, apart from the flags INITIAL, PRE-AUTHENT or TRANSITED-POLICY-CHECKED. If the KDC receives any request that would otherwise cause it to set any flag in the service ticket, apart from the PRE-AUTHENT and TRANSIT-POLICY-CHECKED flags, it must return an error with the error code KDC_ERR_POLICY. The KDC must not generate TGT-based service tickets with any other flags set.

If the TGT contains a caddr field, the KDC must verify that it is a single IP address and that it is identical to the IP address in the IP header of the TGS Request. The KDC must reject TGS Requests from an MTA with a TGT that does not include the MTA's IP address, returning a KDC_ERR_POLICY error code (refer to clause 6.4.4.3).

Upon receipt of a TGS Request, the KDC must:

1) check the validity of the TGT;

2) check the validity of the checksum in the authenticator.

6.4.4.2 TGS Reply Profile

In the TGS Reply, key-expiration, starttime and renew-till optional fields must not be present. The encrypted part of the TGS Reply must be encrypted with the encryption type set to des3-cbc-md5, using the same procedure as described in clause 6.4.2.2.

Upon receipt of a TGS Reply, the client must:

1) use the value of the nonce to bind the reply to the corresponding TGS Request;

2) check the validity of the checksum in the encrypted portion of the TGS Reply.

6.4.4.3 Error Reply

If the KDC is able to successfully parse the TGS Request and the TGT that is inside of it, but the TGS Request is rejected, it must return a Kerberos error message of type KRB_ERROR, as defined in annex B. 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, calculated using the procedure described in clause 6.4.3.1.

The KRB_ERROR must also include typed-data of REQ-NONCE to bind the error message to the nonce from the TGS-REQ message.

Upon receipt of a KRB_ERROR, the client must check the validity of the checksum.

6.4.5 Kerberos Server Locations and Naming Conventions

6.4.5.1 Kerberos Realms

A realm name may use the same syntax as a domain name, however Kerberos Realms must be in all capitals. For a full specification of Kerberos realms, refer to annex B.

6.4.5.2 KDC

Kerberos principal identifier for the local KDC when it is in a role of issuing tickets is always:
krbtgt/<realm>@<realm>, where <realm> is the Kerberos realm corresponding to the particular IPCablecom zone. This is the service name listed inside a TGT.

A Kerberos client must query KDC FQDNs for a particular realm name using DNS SRV records, as specified in [39] and as shown below:

```
<Service Name>.<Protocol>.<Name>    TTL    Class    SRV    Priority    Weight    Port    Target
```

Where:

- The Service Name for Kerberos in IPCablecom must be "_kerberos".
- The Protocol for Kerberos in IPCablecom must be "_udp".
- The Name must be the Kerberos realm name that this record corresponds to.
- TTL, Class, SRV, Priority, Weight, Port, and Target have the standard meaning as defined in [39].

For example, assume the presence of a realm, PACKETCABLE.COM, with two KDCs: kdc1.packetcable.com and kdc2.packetcable.com. These KDCs have different priorities. The DNS SRV records in this case would be:

```
_kerberos._udp.PACKETCABLE.COM.    86400    IN    SRV    0    0    88
kdc1.packetcable.com.

_kerberos._udp.PACKETCABLE.COM.    86400    IN    SRV    1    0    88
kdc2.packetcable.com.
```

To obtain records pertaining to the realm PACKETCABLE.COM, the MTA would send a DNS SRV request for:

```
_kerberos._udp.PACKETCABLE.COM
```

The client, upon receiving a response for a DNS SRV request, must consider the priority/weight as described in the algorithm in [39] and contact the servers in that order. A client must contact the next server based on priority/weight and so on, till all possible server FQDNs and the corresponding IPs are exhausted, if it fails to get a suitable response from the first server listed (refer to clause 6.4.8 for timeout procedures).

For example, after the above DNS SRV records are retrieved, the client will try kdc1.packetcable.com first, based on its priority. (Priority for kdc1.packetcable.com is 0, while priority for kdc2.packetcable.com is 1: a lower priority number means a higher priority.)

When an IPCablecom KDC is requesting information from a Provisioning Server (e.g. the mapping of an MTA MAC address to its corresponding FQDN) it must use a principal name of type NT- PRINCIPAL (1) with a single component "kdcquery" (without quotes).

In an ASCII representation, the principal identifier is as follows:

```
kdcquery@<realm>
```

where <realm> is the Kerberos realm of the KDC.

6.4.5.3 CMS

A CMS Kerberos principal identifier must be constructed from the CMS FQDN as follows:

cms/<FQDN>@<realm>

where <FQDN> is the CMS's FQDN (in lower case) and <realm> is its Kerberos realm.

For example, a CMS with an FQDN 'iptel-cms1.company1.com' and with a realm name 'COMPANY1.COM' would have the principal identifier:

cms/iptel-cms1.company1.com@COMPANY1.COM

The Kerberos PrincipalName data structure (inside the Kerberos messages) is defined as follows:

```
PrincipalName ::= SEQUENCE {
  name-type   [0] INTEGER,
  name-string [1] SEQUENCE OF GeneralString
}
```

Within this data structure, name-type must be NT-SRV-HST (which has the value of 3 according to the Kerberos specification). The name-string element of the data structure must have exactly two components, where the first component has the string value "cms" (without the quotes) and the second component is the CMS's FQDN in lower case.

For the full syntax of Kerberos principal names, refer to annex B.

For the purpose of setting up an IPsec connection between the CMS and RKS, the first component of the CMS principal name must be of the form "cms:<ElementID", where the <ElementID> is described in clause 6.4.5.5.

In the case of a combined network element that integrates the functions of multiple logical elements within the IPCablecom reference architecture (e.g. a single network element that provides both CMS and MGC functionality), the principal name may include all server functions as specified in clause 6.4.5.5.

6.4.5.4 Provisioning Server

When an IPCablecom MTA Provisioning Server is acting in the role of an SNMP manager, it must use a principal name of type NT-SRV-HST (3) with the following two components:

1) "mtaprovsrvr" (without quotes)

2) the FQDN of the Provisioning Server (in lower case)

In ASCII representation, the Provisioning Server's principal identifier must be as follows:

mtaprovsrvr/<Prov Server FQDN>@<realm>

where <realm> is the Kerberos realm of the Provisioning Server.

When an IPCablecom Provisioning Server is providing a service (to the KDC) that maps each MTA MAC address to its corresponding FQDN, it must use a principal name of type NT-SRV-HST (3) with the following two components:

1) "mtafqdnmap" (without quotes)

2) the FQDN of the Provisioning Server (in lower case)

In ASCII representation, the principal identifier must be as follows:

mtafqdnmap/<Prov Server FQDN>@<realm>

where <realm> is the Kerberos realm of the Provisioning Server.

6.4.5.5 Names of Other Kerberized Services

All Kerberized services within IPCablecom, except for the KDC krbtgt service (see 6.4.5.2), must be assigned a service principal name of type KRB_NT_SRV_HST (Value=3), which has the following form according to the Kerberos specification:

<service name>/<FQDN>

This means that the first component of the service principal name is the service name in lower case, and the last is either an FQDN in lower-case or an IP address of the corresponding host. If a specific host has an assigned FQDN, its principal name includes an FQDN and not an IP address. When a KDC receives a ticket request for a service on this host with an IP address instead of an FQDN as the second component of the service principal name, the KDC must reject such a request.

When a KDC database contains a service with a principal name that has an IP address as the second component, all ticket requests for this service must use the same service principal name with the same IP address as the second component. When a KDC receives a ticket request for this service with an FQDN as the second component of the service principal name, the KDC must reject such a request. (This scenario could happen if a service principal is defined in the KDC database at the time when the corresponding host does not have an FQDN, and then later an FQDN for this host is defined as well.)

When an IP address is used, it must be formatted as follows:

[A.B.C.D]

where A, B, C and D are components of an IPv4 address expressed as decimal numbers. The components of an IP address must be separated by a period '.' and the IP address must be surrounded by square brackets.

The following is an example of a principal name based on an IP address:

df/[192.35.65.4]

Figure 3 shows a number of interfaces for which the necessary security is provided by IPsec. In addition to supplying the required key management using IKE with pre-shared keys, some vendors may choose to implement, and operators to deploy, a Kerberized key management scheme for these interfaces.

The present document requires that the RKS verifies billing event messages by ensuring that the Element ID contained in the message matches correctly the IP address at the far end of the IPsec Security Associations. In order to ensure that the RKS is able to maintain this mapping when Kerberized key management is used to generate the Security Associations, devices that communicate with the RKS include their Element ID in their principal name. This information is then passed to the RKS in the cname field of the ticket that the KDC issues; this ticket is passed to the RKS in the AP-REQ that is used to initiate the IPsec Security Associations.

The first component of the principal name for the various IPCablecom devices must be as follows:

1) BP: bp[:<ElementID>]

2) CMTS: cmts[:<ElementID>]

3) DF: df[:<ElementID>]

4) MG: mg[:<ElementID>]

5) MGC: mgc[:<ElementID>]

6) MP: mp[:<ElementID>]

7) MPC: mpc[:<ElementID>]

8) RKS: rks[:<ElementID>]

9) SG: sg[:<ElementID>]

where:

<ElementID> is the identifier that appears in billing event messages and it must be included in a principal name of every server that is capable of generating event messages.

Element ID is defined as an 5-octet right-justified, space-padded ASCII-encoded numerical string [6]. When converting the Element ID for use in a principal name, any spaces must be converted to ASCII zeroes (0x48).

For example, a CMTS that has the Element ID "   311" will have a principal name whose first component is "cmts:00311". Similarly, a DF with no Element ID will have a principal name whose first component is "df".

Components that contain combined elements (such as a CMS with an integrated MGC) must indicate this in the principal name by including all component names, joined with the character "&", in the first component of the principal name. The following is an example of a principal name for a combined CMS and MGC with a single IP address:

cms:00210&mgc:00211/[192.35.65.4]

If the combined component uses a single ElementID, the principal name would be:

cms:00210&mgc:00210/[192.35.65.4]

6.4.6 MTA Principal Names

An MTA principal name must be of type NT-SRV-HST with exactly two components, where the first component must be the string "mta" (not including the quotes) and the second component must be the FQDN of the MTA:

mta/<MTA FQDN>

where <MTA FQDN> is the FQDN of the MTA in lower case.

For example, if an MTA FQDN is "mta12345.mso1.com" and its realm is "MSO1.COM", the principal identifier would be:

mta/mta12345.mso1.com@MSO1.COM

6.4.7 Mapping of MTA MAC Address to MTA FQDN

The MTA authenticates itself with the MTA Device Certificate in the AS Request, where the certificate contains the MTA MAC address but not its FQDN. In order to authenticate the MTA principal name (containing the FQDN), the KDC must map the MTA MAC address (from the MTA Device certificate) to the MTA FQDN, in order to verify the principal name in the AS Request.

The protocol for retrieving the MTA FQDNs is Kerberos-based. The Provisioning Server must listen for the request on UDP port 2246 and must return the response to the UDP port from which the request was transmitted on the client:

1) MTA FQDN Request - sent from the KDC to the Provisioning Server, containing the MTA MAC address and the hash of the MTA public key. This message consists of the Kerberos KRB_AP_REQ concatenated with KRB_SAFE.

2) MTA FQDN Reply - a reply to the KDC by the Provisioning Server, containing the MTA FQDN. This message consists of the Kerberos KRB_AP_REP concatenated with KRB_SAFE.

3) MTA FQDN Error Reply - an error reply in response to the MTA FQDN Request. This message is the Kerberos KRB_ERROR.

The format of each of these messages is specified in the clauses below.

6.4.7.1 MTA FQDN Request

The KDC must first verify the digital signature and certificate chain in the PKINIT Request, before sending out an MTA FQDN Request message to determine the MTA MAC address to FQDN mapping.

In the case where the PKINIT Request and certificate signatures are all valid but the manufacturer certificate is revoked, the KDC may still proceed with the MTA FQDN Request. In this case, the KDC must provide the revocation time in the MTA FQDN Request.

The MTA FQDN Request must be formatted as follows.

Table 6: MTA FQDN Request Format

Field Name	Length	Description
KRB_AP_REQ	Variable	DER-encoded, the length is in the ASN.1 header
KRB_SAFE	Variable	DER-encoded

In the KRB_AP_REQ, only the following option is supported:

- MUTUAL-REQUIRED - mutual authentication required. This option must always be set.
- All other options are not supported.

The encrypted authenticator in the KRB_AP_REQ must contain the following field, which is optional in Kerberos:

- seq-number - random value generated by the KDC.

All other optional fields within the encrypted authenticator are not supported within IPCablecom. In clause 6.5.2.2 there is a requirement that the recipient of a KRB_AP_REQ accepts certain negative values of seq-number; that requirement does not apply when processing the KRB_AP_REQ embedded in a received MTA FQDN message. The authenticator itself must be encrypted using 3-DES CBC with the Kerberos etype value des3-cbc-md5 with the session key from the ticket that is contained in this KRB_AP_REQ object. The encryption method for des3-cbc-md5 is specified in clause 6.4.2.2.

KRB_SAFE must contain the following field, which is optional in Kerberos:

- seq-number - same value as in the KRB_AP_REQ, to tie KRB_SAFE to KRB_AP_REQ and avoid replay attacks.

All other optional fields within KRB_SAFE are not supported within IPCablecom. The keyed checksum within KRB_SAFE must be of type rsa-md5-des3 and must be computed with the session key in the accompanying KRB_AP_REQ. The method for computing an rsa-md5-des3 keyed checksum is specified in clause 6.4.3.1.

The data that is wrapped inside KRB_SAFE must be formatted as follows.

Table 7: KRB_SAFE Format

Field Name	Length	Description
Message Type	1 byte	1 = MTA FQDN Request
Enterprise Number	4 bytes	Network byte order, MSB first 1 = IPCablecom
Protocol Version	1 byte	2 for this version
MTA MAC Address	6 bytes	MTA MAC Address
3 MTA Pub Key Hash	20 bytes	SHA-1 hash of DER-encoded SubjectPublicKeyInfo
Manufacturer Cert Revocation Time	4 bytes	0 = MTA Manufacturer cert not revoked Otherwise, this is UTC time, number of seconds since midnight of Jan 1, 1970, in network byte order

Once the KDC has sent an MTA FQDN Request, it must save the nonce value that was contained in the seq-number field in order to validate a matching MTA FQDN Reply.

If the KDC times out before getting a reply it must give up and simply drop the PKINIT request with no error code returned. The KDC must not retry in this case, since it would still have to handle retries of PKINIT Request from the MTA. At the same time, after a time out the KDC should increase its time out value on the next request to the same Provisioning Server using an exponential back-off algorithm.

The Provisioning Server receiving this message must validate the KRB_AP_REQ and verify that it is not a replay using the procedure specified in the Kerberos standard annex B, also described in clause 6.5.2. After the KRB_AP_REQ has been validated, the Provisioning Server must also verify the KRB_SAFE component: that the checksum keyed with the session key is valid and that the seq-number field matches the KRB_AP_REQ.

If the Manufacturer Cert Revocation Time field is 0 and the Provisioning Server supports the storage of MTA public key hashes, then it must update the MTA public key hash in its database. If the public key hash has changed or is saved for the first time, the Provisioning Server must also record the time this update (to the MTA public key hash) is performed.

If the Manufacturer Cert Revocation Time field is non-zero, the Provisioning Server must validate that the public key hash has not changed from the previous update and that the revocation time is after the last update to the MTA public key hash. If not - the error code KRB_MTAMAP_ERR_PUBKEY_not_TRUSTED must be returned. If the Provisioning Server does not support storage of MTA public key hashes and the Manufacturer Cert Revocation Time field is non-zero, the same error code must be returned.

6.4.7.2 MTA FQDN Reply

The MTA FQDN Reply must be formatted as follows.

Table 8: MTA FQDN Format

Field Name	Length	Description
KRB_AP_REP	Variable	DER-encoded, the length is in the ASN.1 header
KRB_SAFE	Variable	DER-encoded

The encrypted part of the KRB_AP_REP must contain the following field, which is optional in Kerberos:

- seq-number - echoes the value in the KRB_AP_REQ.

All other optional fields within the encrypted part of the KRB_AP_REP are not supported within IPCablecom. It must be encrypted using 3-DES CBC with the Kerberos etype value des3-c

bc-md5 and must be computed with the session key from the preceding KRB_AP_REQ. The encryption method for des3-cbc-md5 is specified in clause 6.4.2.2.

KRB_SAFE must contain the following field, which is optional in Kerberos:

- seq-number - same value as in the KRB_AP_REP, to tie KRB_SAFE to KRB_AP_REP and avoid replay attacks.

All other optional fields within KRB_SAFE are not supported within IPCablecom. The keyed checksum within KRB_SAFE must be of type rsa-md5-des3 and must be computed with the session key from the preceding KRB_AP_REQ. The method for computing an rsa-md5-des3 keyed checksum is specified in clause 6.4.3.1.

The data that is wrapped inside KRB_SAFE must be formatted as follows.

Table 9: KRB_SAFE Data Format

Field Name	Length	Description
Message Type	1 byte	2 = MTA FQDN Reply
Enterprise Number	4 bytes	Network byte order, MSB first. 1 = IPCablecom
Protocol Version	1 byte	2 for this version
MTA FQDN	variable	MTA FQDN
MTA IP Address	4 bytes	MTA-IP Address (MSB first)

After the KDC receives this reply message, it must validate the integrity of both the KRB_AP_REP and KRB_SAFE objects (see annex B) and must also verify that the value of the seq-number field is the same for both. If this integrity check fails, the KDC must immediately discard the reply and proceed as if the message had never been received (e.g. if the KDC was waiting for a valid MTA FQDN Reply it should continue to do so).

The Provisioning Server may set the MTA IP Address field of the MTA FQDN Reply to zero. If the KDC receives an MTA FQDN REPLY with a non-zero MTA IP Address field, it must compare it to the IP address contained in the AS Request. If this check fails, then the KDC must not respond to the AS Request.

6.4.7.3 MTA FQDN Error

If the Provisioning Server is able to successfully parse the KRB_AP_REQ and the ticket that is inside of it, but the MTA FQDN Request is rejected, it must return an error message.

All errors must be returned as a KRB_ERROR message, as specified in annex B. It must include typed-data of REQ-SEQ to bind the error message to the sequence number from the authenticator in the KRB_AP_REQ. Also, 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 and must be computed with the session key from the preceding KRB_AP_REQ, as specified in clause 6.4.3.1. In the case that the client time field inside KRB_AP_REQ differs from the Provisioning Server's clock by more than the maximum allowable clock skew, a clock skew error must be handled as specified in clause 6.5.2.3.2.

If the error is application-specific (not a Kerberos-related error), then KRB_ERROR must include typed-data of type TD-APP-DEFINED-ERROR (value 106). The value of this typed-data is specified in annex B as follows:

```
AppSpecificTypedData ::= SEQUENCE {
  oid        [0] OPTIONAL OBJECT IDENTIFIER,
                 -- identifies the application
  data-value [1] OCTET STRING
                 -- application specific data
}
```

Inside AppSpecificTypedData the oid field must be set to:

enterprises (1.3.6.1.4.1) cableLabs (4491) clabProjects (2) clabProjIPCablecom (2) pktcSecurity (4) errorCodes (1) FQDN (3)

The data-value field must correspond to the following typed-data value:

```
PktcKrbMtaMappingError ::= SEQUENCE {
  e-code [0] INTEGER,
  e-text [1] GeneralString OPTIONAL,
  e-data [2] OCTET STRING OPTIONAL
}
```

The e-code field must correspond to one of the following error code values:

KRB_MTAMAP_ERR_not_FOUND	1	MTA MAC Address not found
KRB_MTAMAP_ERR_PUBKEY_not_TRUSTED	2	MTA public key is not trusted
KRB_MTAMAP_VERSION_UNSUP	3	Unsupported Version Number
KRB_MTAMAP_MSGTYPE_UNKNOWN	4	Unrecognized Message Type
KRB_MTAMAP_ENTERPRISE_UNKNOWN	5	Unrecognized Enterprise Number
KRB_MTAMAP_NOT_YET_VALID	6	MTA not yet valid
KRB_MTAMAP_ERR_GENERIC	7	Generic MTA name mapping error

The optional e-text field can be used for informational purposes (i.e. logging, network troubleshooting) and the optional e-data field is reserved for future use to transport any application data associated with a specific error.

Upon receipt of a KRB_ERROR from the Provisioning Server, the KDC must check the validity of the checksum. If the KRB_ERROR passes the validity check, the KDC must send a corresponding KRB_ERROR to the MTA (as specified in clause 6.4.2.1.2), in response to the PKINIT Request. The application specific MAC-FQDN error codes must be mapped to Kerberos error codes in the error reply to the MTA according to table 10.

Table 10: Mapping of KRB_MTAMAP_ERR to KRB_ERR

KRB_MTAMAP_ERR_NOT_FOUND	KDC_ERR_C_PRINCIPAL_UNKNOWN
KRB_MTAMAP_ERR_PUBKEY_NOT_TRUSTED	KDC_ERR_CLIENT_REVOKED
KRB_MTAMAP_VERSION_UNSUP	KRB_ERR_GENERIC
KRB_MTAMAP_MSGTYPE_UNKNOWN	KRB_ERR_GENERIC
KRB_MTAMAP_ENTERPRISE_UNKNOWN	KRB_ERR_GENERIC
KRB_MTAMAP_NOT_YET_VALID	KDC_ERR_CLIENT_NOTYET
KRB_MTAMAP_ERR_GENERIC	KRB_ERR_GENERIC

6.4.8 Server Key Management Time Out Procedure

The Kerberos client must implement a retransmission strategy using exponential back-off with configurable initial and maximum retransmission timer values for any KDC or application server requests that have not been acknowledged by the server. The Kerberos client must update the client timestamp field with the current time-of-day reading for each such retry. During an exponential back-off, when a previous time out value was T_i, then the next time out value, value T_{i+1}, must satisfy the following criteria:

$$1{,}5 \times T_i \leq T_{i+1} \leq 2{,}5 \times T_i$$

After successfully processing an AS Request or TGS Request and generating a corresponding reply, the KDC must save:

- The AS Request or TGS Request (e.g. the full AS Request / TGS Request or a hash of the AS Request / TGS Request).
- The full KDC reply.

The KDC must maintain this information for all requests with the client time field that is within the time window ($T - \Delta T_{MAX}$, $T + \Delta T_{MAX}$), where T is the current time and ΔT_{MAX} is the maximum clock skew that is allowed by KDC policy.

The KDC may also save:

- The client principal identifier.
- The information that uniquely identifies the client pre-authentication field in the AS Request (PKINIT or encrypted timestamp in the case of non public key AS Request) or TGS Request (PA-TGS-REQ).

The KDC may maintain this information for all requests with the client time field that is within the time window ($T - \Delta T_{MAX}$, $T + \Delta T_{MAX}$), where T is the current time and ΔT_{MAX} is the maximum clock skew that is allowed by KDC policy. If the AS Request or TGS Request is identical to the one previously received, the KDC must respond with the same reply message. If only the principal name and pre-authenticator (PKINIT, encrypted timestamp or PA-TGS-REQ) match, then the KDC must perform one of the following:

- If the received AS Request or TGS Request passes all other error checks, the KDC may reply with a cached reply message.
- Reject this message as a replay.

The MTA may have learned several IP addresses for a KDC or application server (refer to clause 6.4.5.2 for more information on obtaining IP addresses from Realm Names and forming a local list of IP addresses based on prioritization). If the number of retransmissions for a KDC IP address has reached its maximum configured value and there are more IP addresses for the same KDC that have not been tried, then the MTA must direct the retransmissions to the remaining alternate addresses in its local list. Each time that the MTA switches to a new KDC IP address for retransmissions, it must start a new exponential back-off procedure. If there are no more KDC IP addresses to try, then the MTA should actively query the name server in order to detect the possible change of KDC network interfaces, regardless of the Time To Live (TTL) associated with the DNS record to see if any other IP addresses have become available. If there are new IP Addresses discovered, the MTA must go through the retransmission strategy again for the newly discovered IP Addresses.

For Kerberized key management with application servers, when an application layer is informed that key management with a particular IP address failed, it is normally up to the application layer to select the next IP address. The switch over algorithm between multiple IP addresses mapped to the same FQDN is specified by each corresponding application protocol. For example, in the case of the Kerberized key management between the MTA and the CMS, refer to the NCS specification [2]. There are also cases when key management is performed independent of the application layer, e.g. to pre-establish security associations during MTA initialization. In those cases, it is up to a specific MTA implementation to decide if to fail over and how to fail over to another application server IP address.

An application server may not respond to application messages (e.g. NCS messages) from the MTA. This may occur if the MTA has valid security parameters with the application server, but the security parameters on the server have been lost or corrupted (e.g. the CMS rebooted and lost all IPsec Security Associations).

In the case of NCS signalling, an MTA must no longer use any previously established IPsec SAs with a particular CMS each time the NCS backoff and retry algorithm places an MTA endpoint controlled by that CMS into a DISCONNECTED state. After an MTA endpoint has moved to a DISCONNECT state, it will start sending RSIP/disconnect NCS messages which will need to be protected by newly established IPsec SAs.

6.4.9 Service Key Versioning

The service key that is shared between a KDC and an application server, to encrypt/decrypt service tickets, is a versioned key (refer to annex B). This key may be changed either due to a routine key refresh, or because it was compromised. When the Service key is changed, the application server must retain the older key for a period of time that is at least as long as the ticket lifetime used when issuing service tickets (i.e. up to 7 days). In the case of a routine service key change, the application server 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 application server will prevent against many MTAs from suddenly flooding a KDC with PKINIT Requests for new tickets.

If a service key is changed because it has been compromised, the application server 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 application server must respond as specified in annex B with a KRB_AP_ERR_BADKEYVER error. The application server 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 6.5.2.3.1). 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 service ticket which is no longer valid and fetch a new one from its KDC.

6.5 Kerberized Key Management

6.5.1 Overview

This clause specifies how Kerberos tickets are used to perform key management between a client and an Application Server, where a client is able to get a Kerberos ticket for the server but not the other way around.

The same protocol described here applies in a symmetric case - where both sides of a key management interface are able to get a ticket for each other, i.e. each side is both a client and a server. In the symmetric case only the AP Request and AP Reply messages apply.

The Kerberos session key is used in the AP Request and AP Reply messages that are exchanged in order to re-establish security parameters. Subkeys from the AP-REQ and AP-REP are used to derive all of the secret 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 are always valid security parameters between the client and the Application Server. It may also occur on-demand, where the security parameters are allowed to time out and are re-established the next time that application traffic needs to be sent over a secure link.

The UDP port used for all key management messages between the client and the Application Server must be 1293 (on both devices).

A recipient of any Kerberized Key Management message that does not fully comply with the IPCablecom requirements must reject the message.

6.5.2 Kerberized Key Management Messages

Figure 8 illustrates an AP Request / AP Reply exchange.

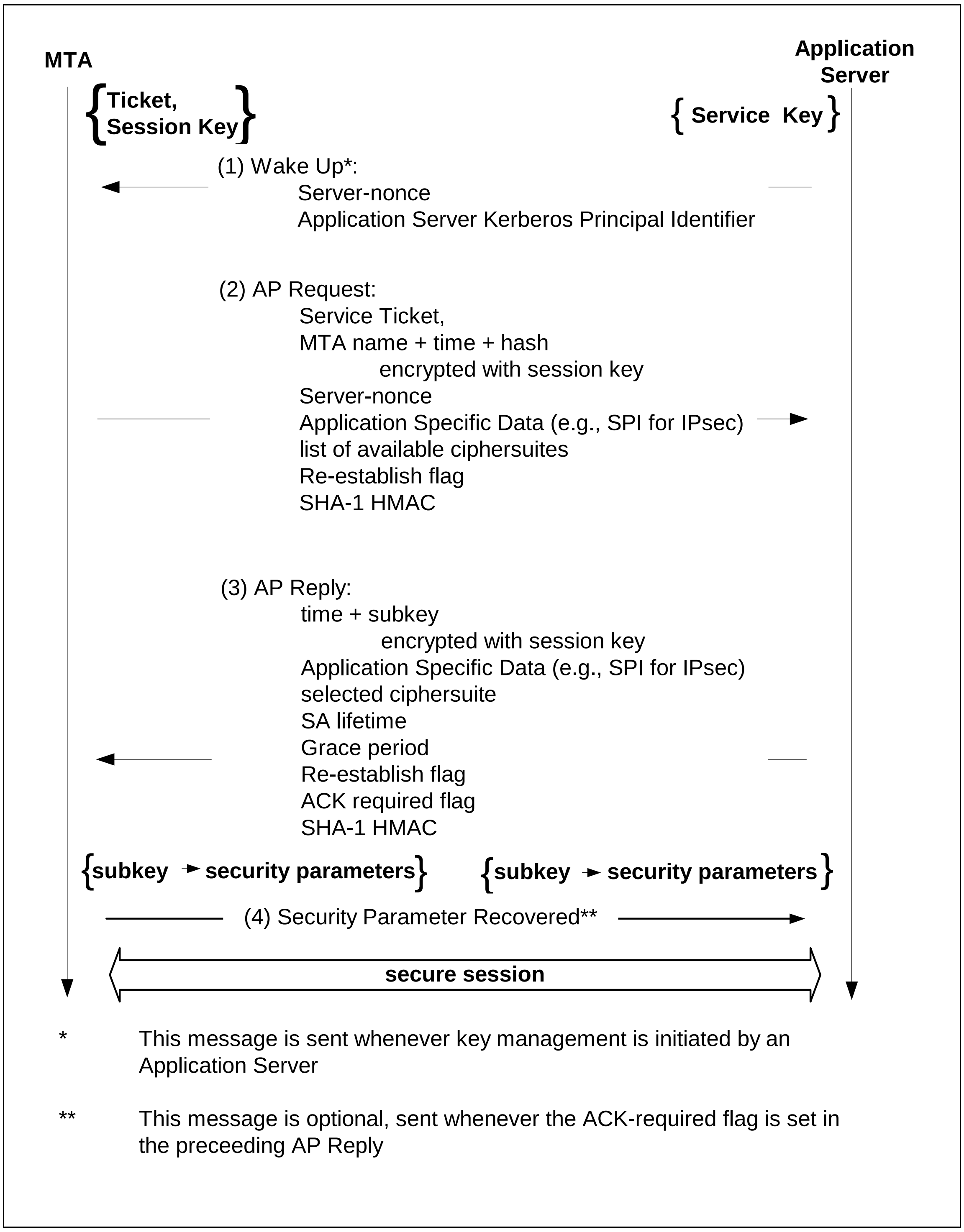

MTA
Application Server
{Ticket, Session Key}
{Service Key}
(1) Wake Up*:
Server-nonce
Application Server Kerberos Principal Identifier
(2) AP Request:
Service Ticket,
MTA name + time + hash
encrypted with session key
Server-nonce
Application Specific Data (e.g., SPI for IPsec)
list of available ciphersuites
Re-establish flag
SHA-1 HMAC
(3) AP Reply:
time + subkey
encrypted with session key
Application Specific Data (e.g., SPI for IPsec)
selected ciphersuite
SA lifetime
Grace period
Re-establish flag
ACK required flag
SHA-1 HMAC
{subkey → security parameters}
{subkey → security parameters}
(4) Security Parameter Recovered**
secure session
* This message is sent whenever key management is initiated by an Application Server
** This message is optional, sent whenever the ACK-required flag is set in the preceeding AP Reply

Figure 8: Kerberos AP Request / AP Reply Exchange

(1) Wake Up - An Application Server sends this message when it initiates a new key management exchange.

To prevent denial-of-service attacks, this message includes a Server-nonce field - a random value generated by the Application Server. The Client includes the exact value of this Server-nonce in the subsequent AP Request.

This message also contains the Server Kerberos Principal Identifier, used by the Client to find or to obtain a correct Kerberos ticket for that Application Server.

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.
- Domain of Interpretation (DOI) - 1 byte value. Specifies the target protocol for which security parameters are established.

Table 10a

DOMAIN OF INTERPRETATION VALUES	
VALUE	**TARGET PROTOCOL**
1	IPsec
2	SNMPv3

- Protocol Version - 1 byte. The high order nibble is the major version number, and the lower order nibble is the minor version number. For IPCablecom, the major number must be 1, and the minor number must be 0.
- Server-nonce - a 4-byte random binary string. Its value must not be all 0's.
- Server Kerberos Principal Identifier - a printable, null-terminated ASCII string, representing the Kerberos Principal Identifier of the Application Server, as defined in clause 6.4.5.

Once the Application Server has sent a Wake Up, it must save the Server-nonce. The Application Server must keep this nonce in order to validate a matching AP Request. In the case of a time out, the Application Server must adhere to the exponential retry backoff procedure described in clause 6.4.8. The Application Server must begin each retry by re-sending a Wake Up message with a new server-nonce value. When the "Timeout Procedure" has completed without success, the Application Server must discard the server-nonce from the last retry, after which it will no longer accept a matching AP Request.

(2) AP Request - must be sent by the Client in order to establish a new set of security parameters. Any time that the Client receives a Wake Up message from a valid application server that is listed as part of client configuration data, it must respond with the AP Request message specified below. If a client receives a Wake Up message from an unknown application server, the client must not respond.

In addition, the present document specifies the use of this message by the Client to periodically establish a new set of security parameters with the Application Server - see clause 6.5.4.2. It also specifies the use of this message by the Client to establish a new set of security parameters with the Application Server, when the Client somehow loses the security parameters (e.g. after a reboot) - see clause 6.5.3.5.

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

The Client sends an AP Request that includes a ticket and an authenticator, encrypted with the session key. The Application Server 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.
- Domain of Interpretation (DOI) - 1 byte value. Specifies the target protocol for which security parameters are established. See table above.
- Protocol Version - 1 byte. The high order nibble is the major version number, and the lower order nibble is the minor version number. For IPCablecom, the major number must be 1, and the minor number must be 0.
- KRB_AP_REQ - DER encoding of the KRB_AP_REQ Kerberos message, as specified in annex B.

- Server-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 Server-nonce field in the Wake Up message. If this AP Request is in response to a Rekey, clause 6.5.2.1, then the value must be identical to that of the Server-nonce field in the Rekey message. Otherwise, the value must be all 0's.
- Application-Specific Data - additional information that must be communicated by the client to the server, dependent on the target protocol for which security is being established (e.g. IPsec or SNMPv3).
- List of ciphersuites available at the Client:

  Number of entries in this list (1 byte)

  Each entry has the following format:

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

  The actual values of the authentication algorithms and encryption transform Ids are dependent on the target protocol.
- Re-establish flag - a 1-byte Boolean value. When the value is TRUE (1), the Client is making an attempt to automatically establish a new set of Security Parameters before the old ones expire. 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 Application Server), it must verify the value of this HMAC. If this integrity check fails, the Application Server must immediately discard the AP Request and proceed as if the message had never been received (e.g. if the Application Server was waiting for a valid AP Request it should continue to do so).

Once the client has sent an AP Request, it must save the nonce value that was contained in the seq-number field (a different nonce from the server-nonce specified above) along with the Server Kerberos Principal Identifier in order to validate a matching AP Reply. If the client generated this AP Request on its own, it must adhere to the exponential retry backoff procedure described in clause 6.4.8.

If the AP Request was generated in response to a message sent by the Application Server (Wake Up or Rekey), then the client must save the nonce and Server Kerberos Principal Identifier until the time specified by the appropriate Key Management MIB variables (pktcMtaDevProvSolicitedKeyTimeout for Prov Server, pktcMtaDevCmsSolicitedKeyTimeout for CMS).

After the timeout has been exceeded or when the "Timeout Procedure" has completed without success, the client must discard this (nonce, Server Kerberos Principal Identifier) pair, after which it will no longer accept a matching AP Reply.

If the MTA generated an AP Request on its own and has reached the maximum number of retries with a particular application server IP address failing to get an AP Reply, it must retry with alternate application server IP addresses as specified in clause 6.4.8.

In the case that the Server-nonce is 0 (not filled in) and the Application Server is currently waiting for a reply to a Wake Up or Rekey message from a client at this IP address, it must reject the AP Request and not reply to the client. If the Application Server is not waiting for a reply to a Wake Up or Rekey message, it must verify that this AP Request is not a replay using the procedure specified in the Kerberos standard (annex B):

- If the timestamp in the AP Request differs from the current Application Server time by more than the acceptable clock skew then Application Server must reply with an error message specified in clause 6.5.2.3.2.
- If the realm, Application Server name, along with the Client 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 may be returned.
- The Application Server must remember any authenticator presented within the acceptable clock skew, so that a replay attempt is guaranteed to fail.
- If the Application Server loses track of any authenticator presented within the acceptable clock skew, it must reject all requests until the interval has passed.

In the case that the Server-nonce is not 0, the Application Server may follow the above procedure in order to fully conform with the Kerberos specification (annex B). In this case, the above procedure is not required because matching the Server-nonce in the Wake Up or Rekey message against the Server-nonce in the AP Request also prevents replays.

(3) AP Reply - Sent by the Application Server in response to AP Request.

The AP Reply must include a randomly generated subkey (inside the Kerberos KRB_AP_REP message), encrypted with the same session key.

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.
- Domain of Interpretation (DOI) - 1 byte value. Specifies the target protocol for which security parameters are established, see table 10a.
- Protocol Version - 1 byte. The high order nibble is the major version number, and the lower order nibble is the minor version number. For IPCablecom, the major number must be 1, and the minor number must be 0.
- KRB_AP_REP - DER encoding of the KRB_AP_REP Kerberos message, as specified in annex B.
- Application-Specific Data - additional information that must be communicated by the server to the client, dependent on the target protocol for which security is being established (e.g. IPsec or SNMPv3).
- Selected ciphersuite for the target protocol, using the same format as defined for AP Request. The number of entries in the list must be one.
- Security parameters lifetime - a 4-byte value, MSB first, indicating the number of seconds from now, when these security parameters are due to expire.
- Grace period - a 4-byte value in seconds, MSB first. This indicates to the client to start creating a new set of security parameters (with a new AP Request / AP Reply exchange) when the timer gets to within this period of their expiration time.
- Re-establish flag - a 1-byte Boolean value. When the value is TRUE (1), a new set of security parameters must be established before the old one expires. When the value is FALSE (0), the old set of security parameters must be allowed to expire.
- 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 Security Parameter 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 Client) it must:

- verify the value of HMAC field in AP Reply. If HMAC integrity check fails, the Client must immediately discard the AP Reply.
- verify that the AP Reply Source IP Address matches the AP Request Destination IP Address in the list of outstanding AP Requests. The Client must immediately discard the AP Reply, which cannot be matched for the corresponding AP Request.
- verify that the nonce value contained in the seq-number field in AP Reply matches the one in the corresponding AP Request. The Client must immediately discard the AP Reply if seq-number field value in AP Reply does not match.

If the AP Reply is discarded, the Client must proceed as if the message had never been received (e.g. if the Client was waiting for a valid AP Reply it should continue to do so).

Once the Application Server has sent an AP Reply with the ACK-required flag set, it must compute the expected value in the Security Parameter Recovered message and save it for an appropriate timeout period during which it will accept a matching Security Parameter Recovered Message. Once the appropriate timeout period is exceeded, the Application Server must discard the saved values and no longer accept a matching Security Parameter Recovered Message.

Each time the Application Server times out waiting for the Security Parameter Recovered message, it must continue with the exponential back-off algorithm until all retries have been exhausted, as specified in clause 6.4.8. The Application Server must begin each retry by re-sending a Wake Up message with a new server-nonce value.

(4) Security Parameter Recovered - Sent by the Client to the Application Server to acknowledge that it received an AP Reply and successfully set up new Security Parameters. This message is only sent when ACK-required flag is set in the AP Reply.

This message must consist of the concatenation of the following:

- Key Management Message ID - 1 byte value. Always set to 0x04.
- Domain of Interpretation (DOI) - 1 byte value. Specifies the target protocol for which security parameters are established, see table 10a.
- Protocol Version - 1 byte. The high order nibble is the major version number, and the lower order nibble is the minor version number. For IPCablecom, the major number must be 1, and the minor number must be 0.
- HMAC - a 20-byte SHA-1 HMAC of the preceding AP Reply message. The 20-byte key for this HMAC is determined by taking a SHA-1 hash of the subkey from the AP Reply.

If the receiver (Application Server) gets a bad Security Parameter Recovered message that does not match an AP Reply, the Application Server must discard it and proceed as if this Security Parameter Recovered message was never received.

6.5.2.1 Rekey Messages

The Rekey message replaces the Wake Up message and provides better performance, whenever a receiver (Application Server) wants to trigger the establishment of a Security Parameter with a specified Client. The 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 Security Parameter with a Client, without a significant impact to the ongoing voice communication.

The use of the Rekey message eliminates the need for the AP Reply message, thus reducing the key management overhead to a single roundtrip. This is illustrated in figure 9.

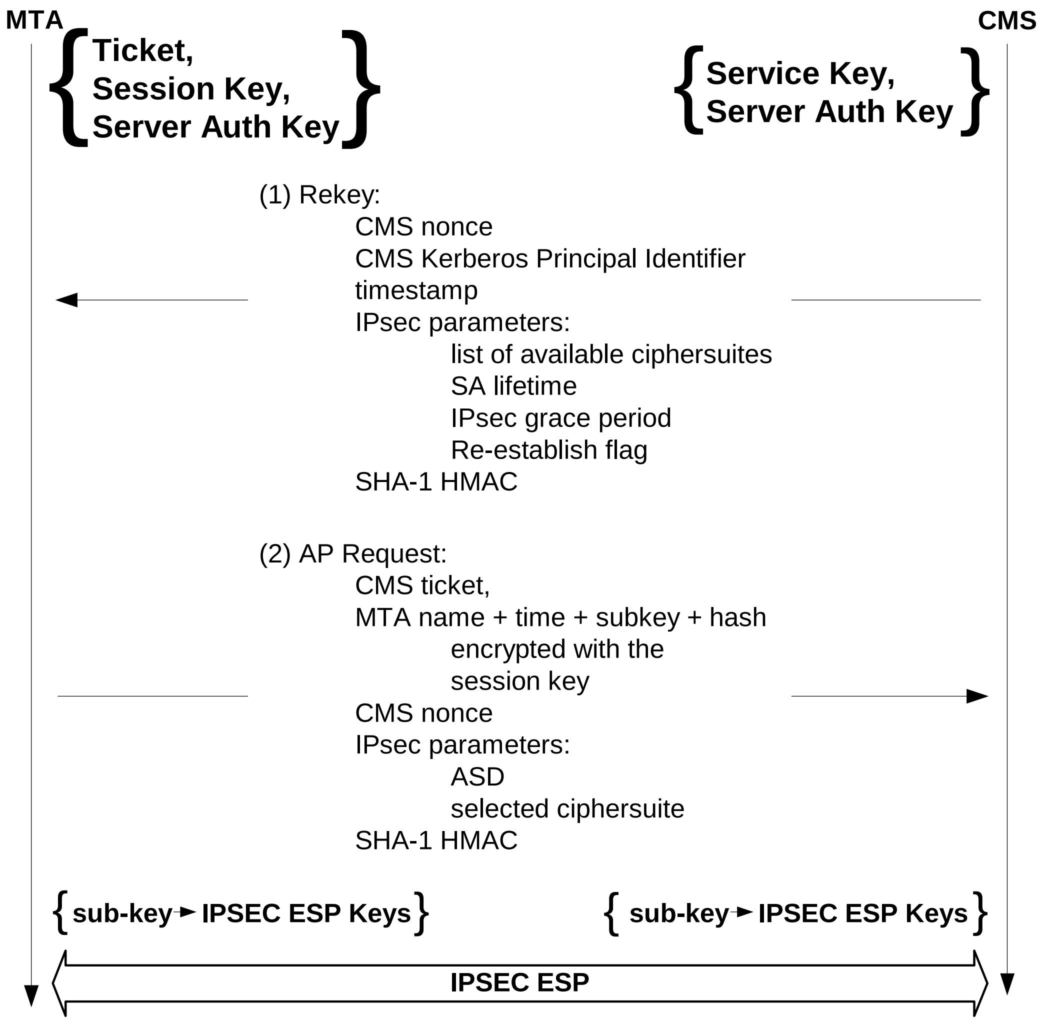


Figure 9: Rekey Message to Establish a Security Parameter

The messages listed in figure 9 are defined as follows:

(1) Rekey - sent by the Application Server to establish a new set of Security Parameters. It must be a concatenation of the following:

- Key Management Message ID - 1 byte value. Always set to 0x05.
- Domain of Interpretation (DOI) - 1 byte value. Specifies the target protocol for which security parameters are established, see table 10a.
- Protocol Version - 1 byte. The high order nibble is the major version number, and the lower order nibble is the minor version number. For IPCablecom, the major number must be 1, and the minor number must be 0.
- Server-nonce - a 4-byte random binary string. Its value must not be all 0's.
- Server Kerberos Principal Identifier - a printable, null-terminated ASCII string, representing the Kerberos Principal Identifier of the Application Server, as defined in clause 6.4.5. This allows the Client 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.
- Application-Specific Data - additional information that must be communicated by the server to the client, dependent on the target protocol for which security is being established (e.g. IPsec).

- List of ciphersuites available at the server - see above specification for the AP Request message.
- Security parameters lifetime - a 4-byte value, MSB first. This indicates the number of seconds from now, when this set of security parameters is due to expire.
- Grace period - a 4-byte value in seconds, MSB first. This indicates to the client to start creating a new set of security parameters (with a new AP Request / AP Reply exchange) when the timer gets to within this period of their expiration time.
- Re-establish flag - a 1-byte Boolean value. When the value is TRUE (1), a new set of security parameters must be established before the old one expires. When the value is FALSE (0), the old set of security parameters must be allowed to expire.
- 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 the following name pair (client principal name, server principal name). This key must be updated at the Application Server right after it sends an AP Reply message. It must be set to a (20-byte) SHA-1 hash of the Kerberos session key used in that AP Reply. The Client must also update 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.)

It is possible, that the Application Server sends a Rekey message as soon as it sends an AP Reply (from another IP address), and before the Client is able to derive the new Server Authentication Key. In that case, the Client will not authenticate the Rekey message and the Application Server will have to retry.

Similarly, after sending an AP Reply the Application Server might immediately send an IP packet using the just established Security Parameter, when the Client is not yet ready to receive it. In this case, the Client will reject the packet and the Application Server will have to retransmit.

Both of these error cases could be completely avoided with a 3-way handshake (a Client acknowledging an AP Reply with a Security Parameter Recovered message).

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

Once the Application Server has sent a Rekey, it must save the server-nonce in order to validate a matching AP Request. In the case of a time out, the Application Server must adhere to the exponential retry backoff procedure described in clause 6.4.8. The Application Server must begin each retry by re-sending a Rekey message with a new server-nonce value. When the "Timeout Procedure" has completed without success, the Application Server must discard the server-nonce from the last retry, after which it will no longer accept a matching AP Request.

When this Rekey message is received and validated by the Client, all previously existing outgoing Security Parameters with this Application Server IP address must be removed at this time. If the Client previously had a timer set for automatic refresh of Security Parameters with this Application Server IP address, that automatic refresh must be reset or disabled.

The Client must verify that this Rekey message is not a replay using the procedure similar to the one for AP Request in the Kerberos standard annex B:

- If $|T_{CMS} - (T_{MTA} + \text{Skew})| >$ The acceptable Clock Skew then the Client must drop the message. Here, T_{CMS} is the timestamp in the Rekey message and T_{MTA} is the reading of the MTA local clock. Skew is the saved difference between the Application Server and MTA clock. PktcSrvrToMtaMaxClockSkew is currently in the MTA MIB (see [25] and [48]) as the variable pktcMtaDevCmsMaxClockSkew.
- If the Server-nonce, principal name and timestamp fields match any recently seen (within the pktcSrvrToMtaMaxClockSkew) Rekey messages, then the Client must drop the message.

(2) AP Request - must be sent by the Client as a response to a Rekey message. Unlike the AP Request message described above, this one must also include 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.

The format of the AP Request is as specified above in clause 6.5.2. The only difference is that the list of ciphersuites here must contain exactly one entry - the ciphersuite selected by the client from the list provided in the Rekey message.

Right before the client sends out this AP Request, it must establish the security parameters with the corresponding server IP address. If the corresponding Rekey message had the Re-establish flag set, the client must be prepared to automatically re-establish new security parameters, as specified in clause 6.5.

Once this AP Request is received and verified by the Application Server, the server must also establish the security parameters.

6.5.2.2 IPCablecom Profile for KRB_AP_REQ / KRB_AP_REP Messages

In the KRB_AP_REQ, only the following option is supported:

- MUTUAL-REQUIRED - mutual authentication required. When this option is set, the server must respond with an AP Reply message. When this option is not set, the AP Reply message must not be sent in reply.

  All other options must not be set. If an application server receives a request containing the unsupported option USE-SESSION-KEY, it must return an error with the error code KRB_AP_ERR_METHOD. If an application server receives a request containing any other unsupported options, it must return an error with the error code KRB_ERR_GENERIC.

  When MUTUAL-REQUIRED is set, the encrypted authenticator in the KRB_AP_REQ must contain the following field, which is optional in Kerberos:

  - seq-number must contain a pseudo-random number generated by the client (to be used as a nonce).
  - The server must accept otherwise-valid KRB-AP-REQ messages that contain a seq-number in the range -2^31 to -1.

When MUTUAL-REQUIRED is not set, the encrypted authenticator must contain the following field that is optional in Kerberos.

- subkey - used to generate security parameters for the target protocol. The subkey type must be set to -1. The actual subkey length is dependent on the target protocol.

When MUTUAL-REQUIRED is set, the target protocol is IPsec and the client is an MTA, the client may include the subkey field; in the case that the target protocol is IPsec and the client is other than an MTA, the client should include the subkey field. For IPsec, the subkey, if present, must contain a pseudo-random number of length 46 octets generated by the client.

Other optional fields in the authenticator must not be present. If the authenticator contains the authorization-data field, the application server must return an error with the error code KRB_ERR_GENERIC. If the authenticator contains any other optional fields (apart from subkey and authorization-data), the application server must ignore those fields.

The negative key type is used to indicate that it is application-specific and not defined in the Kerberos specification. When the Kerberos specification is updated to include this key type, the IPCablecom specification will be updated accordingly.

The authenticator itself must be encrypted using 3-DES CBC with the Kerberos etype value des3-cbc-md5 as it is specified in clause 6.4.2.2.

In the encrypted part of the KRB_AP_REP, the optional subkey field must be used for IPCablecom. Its type and format must be the same as when it appears in the KRB_AP_REQ (see above).

The optional seq-number must be present, and must 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 must be encrypted with the Kerberos etype value des3-cbc-md5 as specified in clause 6.4.2.2.

6.5.2.3 Error Handling

6.5.2.3.1 Error Reply

If the Application Server 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 an error message. This error message must be formatted as the concatenation of the following fields:

- Key Management Message ID - 1 byte value. Always set to 0x06.

- Domain of Interpretation (DOI) - 1 byte value. Specifies the target protocol for which security parameters are established. See clause 6.5.2.
- Protocol Version - 1 byte value. The high order nibble is the major version number and the lower order nibble is the minor version number. For IPCablecom the major version number must be 1 and the minor version number must be 0.
- KRB_ERROR - Kerberos error message as specified in annex B. It must include typed-data of REQ-SEQ to bind the error message to the sequence number from the authenticator in the AP-REQ message. The value encapsulated by the REQ-SEQ typed data must be the same as the value of the seq-number that was sent by the client in the KRB_AP_REQ. Also, 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 clause 6.4.3.1.

If the error is application-specific (not a Kerberos-related error), then the KRB_ERROR must include typed-data of type TD-APP-DEFINED-ERROR (value 106). The value of this typed-data is the following ASN.1 encoding (specified in annex B):

```
AppSpecificTypedData ::= SEQUENCE {
  oid        [0] OPTIONAL OBJECT IDENTIFIER,
                 -- identifies the application
  data-value [1] OCTET STRING
                 -- application specific data
}
```

Both the oid and the data-value fields inside AppSpecificTypedData are specified separately for each DOI.

Upon receiving this error reply, the Client must verify both the keyed checksum and the REQ-SEQ field, to make sure that it matches the seq-number field from the authenticator in the AP Request.

If the Application Server 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 Client. In case of a line error, the Client will time out and re-send its AP Request. If the verification has failed, then the MTA must ignore this error message and continue waiting for the reply as if the error message was never received.

When a client receives an error message, in some cases the present document calls for the client to take some recovery steps and then send a new AP Request message. When a client is responding to an error message, it is not a retry and must not be considered to be part of the client's back-off and retry procedure specified in clause 6.4.8. The client must reset its timers accordingly, to reflect that the AP Request in response to an error message is not a retry.

Although the present document calls for an application server to return some specific error codes under certain error conditions, in the case when a server is repeatedly getting the same error from the same client IP address, it may at some point choose to stop sending back any further replies (errors or otherwise) to this client.

6.5.2.3.2 Clock Skew Error

When the Application Server clock and the client clock are off by more than the limit for a clock skew, an error code KRB_AP_ERR_SKEW must be returned. The value for the maximum clock skew allowed by the Application Server must not exceed 5 minutes. The optional client's time in the KRB_ERROR must be filled out, and the client must compute the difference (in seconds) between the two clocks based upon the client and server time contained in the KRB_ERROR message. The client should store this clock difference in non-volatile memory and must use it to adjust Kerberos timestamps in subsequent AP Request messages by adding the clock skew to its local clock value each time. The client must maintain a separate clock skew value for each realm and may share the same clock skew between the KDC and various application servers within that realm. The clock skew values are intended for uses only within the Kerberos protocol and should not otherwise affect the value of the local clock (since a clock skew is likely to vary from realm to realm).

In the case that an AP Request failed due to a clock skew error, a client must immediately retry after adjusting the Kerberos timestamp inside the AP Request message.

Additionally, the Client 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 Client check against a maximum time offset protects against an attack, where a rogue KDC attempts to fool a Client into accepting an expired KDC certificate (later, during the next PKINIT exchange).

6.5.2.3.3 Handling Ticket Errors After a Wake Up

6.5.2.3.3.1 KRB_AP_ERR_BADKEYVER after a Wake Up

This clause addresses a scenario when an application server sends a Wake Up to a client and subsequently receives an AP Request that contains a ticket that is encrypted using an obsolete service key (results in the KRB_AP_ERR_BADKEYVER error code). This error normally requires the client to get another ticket and retry but in this particular scenario the client has to retry in the middle of a key management transaction.

In this scenario, the application server must reply to the invalid AP Request with the KRB_ERROR message with the KRB_AP_ERR_BADKEYVER error code. Subsequent to the reply, the server must wait for another AP Request and must use the same time out value that it would normally use when waiting for an AP Request. The client, upon getting back the above error code must attempt to obtain a new ticket from the KDC (if the client has not done so already while waiting for server's reply) and if successful, must send another AP Request to the application server. If the client is unsuccessful in obtaining another ticket, it must not reply. If the server times out waiting for the second AP Request, it must proceed as if it timed out waiting for the original AP Request.

If the application server is able to validate the second AP Request, it must then proceed as specified in clause 6.5.3. If the second AP Request again results in the KRB_AP_ERR_BADKEYVER error, the server must abort key management with this client and not reply.

6.5.2.3.3.2 KRB_AP_ERR_SKEW After a Wake Up

An application server is not required to check for a clock skew in this case, but if it does generate the KRB_AP_ERR_SKEW, the same procedure must be followed as in clause 6.5.2.3.3.1, except that the client must retry after adjusting the timestamp (see clause 6.5.2.3.2) instead of getting a new ticket.

6.5.3 Kerberized IPsec

This clause specifies the Kerberized key management profile specific to IPsec ESP in transport mode. IPsec uses the term Security Association (SA) to refer to a set of security parameters. IPsec Security Associations are always uni-directional and they must always be established in pairs within IPCablecom.

An MTA must establish SAs with the IP address from where the corresponding Kerberized IPsec key management message (AP-REP or REKEY) has been received. Note that a CMS can notify an MTA that it is listening for NCS messages on a different port. Also, both the CMS and the MTA can send NCS messages from different ports, and the response must be sent to the port from which the message was sent. Kerberized Key Management does not allow for the negotiation of source or destination ports. Therefore SAs established to protect NCS signalling need to support multiple ports. One way to accomplish this is to establish two separate policies, outbound and inbound, in the IPsec Security Policy Database (see [19]). Table 11 illustrates an example policy that would support changes in port numbers. Note that this table only illustrates inbound and outbound policies for NCS signalling between a specific MTA and a specific CMS. The table is not a complete IPsec Security Policy Database. Other entries would be required to support communications over different protocols with the same host (e.g. Kerberized Key Management), communications with other hosts, or default policies for unknown hosts.

Table 11: Example IPsec Security Policy Database Entries for NCS Signalling between MTA and CMS

Direction	Policy	Source IP	Source Port	Destination IP	Destination Port
Inbound - this applies to messages being received	Apply IPsec ESP	Remote IP address	Wildcard - any port	Local IP address	Bind to local port(s) that NCS messages will be sent from, and the provisioned NCS listening port.
Outbound - this applies to messages being sent	Apply IPsec ESP	Local IP address	Bind to local port(s) that messages will be sent from.	Remote IP address	Wildcard - any port

The DOI value for IPsec must be set to 1.

The ASD (Application-Specific Data) field in the AP Request key management message must be the SPI (Security Parameter Index) for the client's inbound Security Association. It is a 4-byte integer value, MSB first.

The ASD (Application-Specific Data) field in the AP Reply and Rekey key management messages must be the SPI (Security Parameter Index) for the server's inbound Security Association. It is a 4-byte integer value, MSB first

The subkey for IPsec must be a 46-byte value, defined as follows:

- If the AP-REQ does not include a subkey, the 46-octet subkey from AP-REP is taken as the subkey for IPsec.
- If the AP-REQ does include a subkey but no AP-REP (in the case of Rekey) is sent, then the 46-octet AP-REQ subkey is used as the subkey for IPsec.
- Otherwise, both the AP-REQ and the AP-REP messages include 46-octet subkeys, and their bit-by-bit XOR is the 46-byte subkey for IPsec.

An MTA must not perform Kerberized Key Management or establish IPsec Security Associations with a CMS when the pktcMtaDevCmsIpsecCtrl flag for that CMS is set to false in the pktcMtaDevCmsTable. Note that this flag may only be set in the MTA configuration file and cannot be updated using SNMPv3. In the case of an NCS Redirect or any other dynamic method for associating a new CMS with an MTA endpoint where there is not an entry in the pktcMtaDevCmsTable for the new CMS, the MTA must perform Kerberized Key Management and establish IPsec Security Associations with the new CMS.

The CMS must be capable of disabling its Kerberized Key Management interface. The CMS must not perform Kerberized Key Management or establish IPsec Security Associations when so configured.

6.5.3.1 Derivation of IPsec Keys

After the Application Server sends out an AP Reply message, it is ready to derive a new set of IPsec keys. Similarly, after the Client 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 Kerberos subkey.

The size of the Kerberos subkey must be 46 bytes (the same as with the SSL or TLS pre-master secret).

The IPsec ESP keys must be derived in the following order:

1) Message authentication key for Client->Application Server messages
2) Encryption key for Client->Application Server messages
3) Message authentication key for Application Server->Client messages
4) Encryption key for Application Server->Client messages

For specific authentication and encryption algorithms that may be used by IPCablecom for IPsec, refer to clause 6.1.2.

The derivation of the required keying material must be based on running a one-way pseudo-random function F(S, "IPsec Security Association") recursively until the right number of bits has been generated. Here, S is the Kerberos subkey and the ASCII string "IPsec Security Association" is taken without quotes and without a terminating null character. F is defined in clause 9.6.

6.5.3.2 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 Client and the Application Server handle the re-establishment of IPsec Security Associations (re-establish flag was TRUE in the AP Reply). When the re-establishment of IPsec SAs is required there must always be at least one SA available for each direction and there must not be an interruption in the call signalling.

6.5.3.2.1 Periodic Re-establishment of IPsec SAs at the Client

If the re-establish flag is set, the Client 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 Client must send an AP Request as specified in clause 6.5. The destination IP Address of the AP Request message must be the destination IP Address of the outbound IPsec SA that is about to expire. After the Client 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 Kerberos subkey, from which the IPsec keys are derived as specified in clause 6.5. The expiration time for the outgoing SA must be set to TEXP, while the expiration time for the incoming SA must be set to TEXP + GPIPsec.

2) From this point forward, the new SA must be used for sending messages to the Application Server. The old SA that the Client used for sending Signalling messages to the Application Server may be explicitly removed at this time, or it may be allowed to expire (using an IPsec timer) at the time TEXP.

3) Continue accepting incoming Signalling messages from the Application Server on both the old and the new incoming SAs, until the time TEXP + GPIPsec. After this time, the old incoming SA must expire. If a Client receives a Signalling message from the Application Server using a new incoming SA at an earlier time, it may at that time remove the old incoming SA.

If the client fails to get any reply from the server and has to retry one or more times with another AP Request, the re-establish flag must be set to FALSE in each retry. This implies that when CMS processes a retry, it will remove any existing outgoing IPsec SAs, including the ones that may have been created after the processing of the initial AP Request, and proceed as if it is processing the SAs on demand (see clause 6.5.3.5.1).

6.5.3.2.2 Periodic Re-establishment of IPsec SAs at the Application Server

When an AP Request message is received with re-establish flag set, the Application Server must perform the applicable processing steps in clause 6.5.2. If the client is an MTA, the Application Server must also verify that the source IP address in the received datagram of the AP Request message is the same IP address as was used when the initial SA was established. The Application Server must ignore the AP Request if the IP addresses do not match.

In addition, the Application Server must perform the following steps, in the specified order, immediately before an AP Reply is returned.

1) Create new IPsec SAs, based on the negotiated ciphersuite, SPIs and on the established Kerberos subkey, from which the IPsec keys are derived as specified in clause 6.5.

2) Send back an AP Reply.

3) Continue sending Signalling messages to the Client using an old outgoing SA until the time TEXP. During the same period, accept incoming messages from either the old or the new incoming SA.

4) At the time TEXP both the old incoming and the old outgoing SAs must expire. At the time TEXP, the Application Server must switch to the new SA for outgoing Signalling messages to the Client. If for some reason the new IPsec SAs were not established successfully, there would not be any IPsec SAs that are available after this time.

6.5.3.3 Expiration of IPsec SAs

An IPsec SA is defined with an expiration time T_{EXP} and a grace period $GP_{IPsec.}$ This clause specifies how both the Client and the Application Server must handle the expiration of IPsec Security Associations (re-establish flag was FALSE in the AP Reply).

At the Client:

- Outgoing SA expires at T_{EXP}
- Incoming SA expires at $T_{EXP} + GP_{IPsec}$

At the Application Server:

- Outgoing SA expires at T_{EXP}
- Incoming SA expires at $T_{EXP} + GP_{IPsec}$

Whenever an IPsec SA has been expired and a Signalling message needs to be sent by either the Client or the Application Server, the key management layer must be signalled to establish a new IPsec SA. It is established using the same procedures as the ones specified in clause 6.5.3.5.

6.5.3.4 Initial Establishment of IPsec SAs

When a Client is rebooted, it does not have any current IPsec SAs established with the Application Server, since IPsec SAs are not saved in non-volatile memory. In order to re-establish them, it must go through the recovery procedure that is described in clause 6.5.3.5.

6.5.3.5 On-demand Establishment of IPsec SAs

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

6.5.3.5.1 Client Loses an Outgoing IPsec SA

If a client attempts to send a Signalling message to the Application Server without a valid IPsec SA, the IPsec layer in the Client will realize the SA is missing and return an error back to the Signalling application. In this case, the following recovery steps must be taken at the key management layer:

1) The Client first makes sure that it has a valid Kerberos ticket for the Application Server. If not, it must first perform a PKINIT exchange as specified in clause 6.4.2.

2) Client sends a new AP Request to the Application Server and gets back an AP Reply, as specified in clause 6.5.2. After the receipt of an AP Reply the Client must:

   - create new IPsec SAs;

   - remove any old outgoing IPsec SAs;

   - be prepared to use both of the newly created IPsec SAs.

3) If the Kerberos ticket includes the optional caddr field and the caddr does not contain a matching source IP address for the AP Request datagram, the Application Server must ignore the request.

4) The Application Server must not set the ACK-required flag in the AP Reply. Right after sending out an AP Reply, the Application Server must be prepared to both send and receive messages on the newly created SAs.

5) After receiving this AP Request (with Re-establish flag = FALSE), the Application Server must remove any existing outgoing IPsec SAs that it might already have for this Client.

The key management application running on the Client must send an explicit signal to the Signalling application when it completes the re-establishment of the IPsec SAs.

6.5.3.5.2 Client Loses an Incoming IPsec SA

When the Client receives an IP packet from an Application Server on an unrecognized IPsec SA, the Client must ignore this error and the packet must be dropped.

6.5.3.5.3 Application Server Loses an Outgoing IPsec SA

When an Application Server attempts to send a Signalling message to the Client, and the IPsec layer in the Application Server realizes a valid SA is missing, the IPsec layer must return an error back to the Signalling application.

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

In this case, the following recovery steps must be taken at the key management layer:

1) Application Server sends a Wake Up message to the Client.

2) The Client makes sure that it has a valid Kerberos ticket for the Application Server. If not, it must first obtain it from the KDC.

3) Client sends a new AP Request to the Application Server, as specified in clause 6.5.2. If the Kerberos ticket includes the optional caddr field and the caddr does not contain a matching source IP address for the AP Request datagram, the Application Server must ignore the request.

4) For each AP Request, the Client generates a nonce and puts it into the seq-number field. As specified in clause 6.5.2, the Client will save this nonce for a period of time specified by the pktcMtaDevCmsSolicitedKeyTimeout MIB object and wait for a matching AP Reply (this is not the same nonce as the Server-nonce received in the Wake Up). However, after this timeout, the Client must not retry and must abort an attempt to establish an IPsec SA in response to a received Wake Up.

   Once the Client gets back a matching AP Reply, it will be in the format specified in clause 6.5.2. The ACK-required flag in the AP Reply must be set, to insure that the Client replies with the SA Recovered message in the following step.

   If this Client previously had any outgoing IPsec SAs with this Application Server IP address, they must be removed at this time. If the Client previously had a timer set for automatic refresh of IPsec SAs with this Application Server IP address, that automatic refresh must be reset or disabled. The Client may start using both of the newly created SAs. If the AP Reply had the Re-establish flag set, the Client must be prepared to automatically re-establish new IPsec SAs, as specified in clause 6.5.3.2.

   The Application Server can receive Signalling messages from the Client on the new incoming SA, but cannot yet start using an outgoing SA for sending messages to the Client.

5) Immediately after the Client establishes the new IPsec SAs, it must send a SA Recovered message to the Application Server.

6) Upon receipt of this message, the Application Server must immediately activate the new outgoing SA for sending Signalling messages to the Client.

The key management application running on the Application Server must send an explicit signal to the Signalling application when it completes the re-establishment of the IPsec SAs.

6.5.3.5.4 Application Server Loses an Incoming IPsec SA

When the Application Server receives an IP packet from a Client on an unrecognized IPsec SA, the Application Server must ignore this error and the packet must be dropped. In this case, any attempt at recovery (e.g. establishing a new SA) is prone to denial-of-service attacks.

6.5.3.6 IPsec-Specific Errors Returned in KRB_ERROR

Inside AppSpecificTypedData the oid field must be set to: enterprises (1.3.6.1.4.1) cableLabs (4491) clabProjects (2) clabProjPacketCable (2) pktcSecurity (4) errorCodes (1) ipSec (1).

The data-value field must correspond to the following typed-data value:

```
PktcKrbIpsecError ::= SEQUENCE {
  e-code [0] INTEGER,
  e-text [1] GeneralString OPTIONAL,
  e-data [2] OCTET STRING OPTIONAL
}
```

The e-code field must correspond to one of the following error code values:

KRB_IPSEC_ERR_NO_POLICY	1	No IPsec policy defined for request
KRB_IPSEC_ERR_NO_CIPHER	2	No support for requested ciphersuites
KRB_IPSEC_NO_SA_AVAIL	3	No IPsec SA available (i.e. SAD is full)
KRB_IPSEC_ERROR_GENERIC	16	Generic KRB IPsec error

The optional e-text field can be used for informational purposes (i.e. logging, network troubleshooting) and the optional e-data field is reserved for future use to transport any application data associated with a specific error.

6.5.4 Kerberized SNMPv3

This clause specifies the Kerberized key management profile specific to SNMPv3, see [28]. In the case of SNMPv3, the security parameters are associated with the usmUserName (SNMPv3 user name), the agent's usmUserEngineID (SNMPv3 engine ID) and the manager's usmUserEngineID.

Multiple SNMP managers on different hosts but with the same user name are considered as unique Kerberos principals. Still, the SNMPv3 keys generated by any one of these SNMP managers must be shared across all the managers - as long as they apply to the same SNMPv3 user name and the same SNMPv3 engine ID (of the agent).

The security parameters consist of a single authentication key, a single privacy (encryption) key, SNMPv3 boot count and engine time. SNMPv3 privacy can be turned off by selecting a NULL encryption transform.

The DOI value for SNMPv3 must be set to 2.

The ASD field in the AP Request message must be set to the concatenation of the following:

Table 12: Required Format for Data in the AP Request

Attribute	Length
Agent's snmpEngineID Length	1 byte
Agent's snmpEngineID	variable
Agent's snmpEngineBoots	4 bytes, network byte order
Agent's snmpEngineTime	4 bytes, network byte order
usmUserName Length	1 byte
usmUserName	variable

The ASD field in the AP Reply message must be set to the concatenation of the following:

Table 13: Required Format for Data in the AP Reply

Attribute	Length
Manager's snmpEngineId Length	1 byte
Manager's snmpEngineId	variable
Manager's snmpEngineBoots	4 bytes, network byte order
Manager's snmpEngineTime	4 bytes, network byte order
usmUserName Length	1 byte
usmUserName	variable

For IPCablecom MTAs, the usmUserName contains in it the MTA MAC address (see [4]). The manager must verify that this MAC address and the MTA FQDN specified in the MTA principal name match. The manager must also verify that any SNMP INFORM message containing a MAC address from the MTA contains a correct MAC address - the same one that is in the usmUserName. The usmUserName field inside the application-specific data field in the AP Reply must be the same as the one in the preceding AP Request.

The Rekey message is not used for SNMPv3 key management.

The subkey for SNMPv3 must be a 46-byte value.

6.5.4.1 Derivation of SNMPv3 Keys

After the server sends out an AP Reply message, it is ready to derive a new set of SNMPv3 keys. Similarly, after the client receives this AP Reply, it is ready to derive the same set of keys for SNMPv3. This clause specifies how the SNMPv3 keys are derived from the Kerberos subkey.

The size of the Kerberos subkey must be 46 bytes.

The derived SNMPv3 keys must be as follows, in the specified order:

- SNMPv3 authentication key
- SNMPv3 privacy key

For specific authentication and encryption algorithms that may be used by IPCablecom for SNMPv3, refer to clause 6.3.

The derivation of the required keying material must use a one-way pseudo-random function F(S, "SNMPv3 Keys") recursively until the right number of bits has been generated. Here, S is the subkey and the string "SNMPv3 Keys" is taken without quotes and without a terminating null character. F is defined in clause 9.6.

6.5.4.2 Periodic Re-establishment of SNMPv3 Keys

Periodic re-establishment of SNMPv3 keys, where the next set of keys is created before the old one expired, is currently not supported by IPCablecom. The re-establish flag in the AP Reply key management message must be set to FALSE.

6.5.4.3 Expiration of SNMPv3 Keys

Expiration of SNMPv3 keys is currently not supported by IPCablecom. The values of the Security Parameters Lifetime and Grace Period fields in the AP Reply must be set to 0.

6.5.4.4 Initial Establishment of SNMPv3 Keys

When a client is rebooted, it may not have any saved SNMPv3 keys established with the SNMP Manager. In order to re-establish them, it goes through the recovery procedure that is described in clause 6.5.4.5.1.

6.5.4.5 Error Recovery

This clause describes the recovery steps that must be taken in the case that SNMPv3 keys are somehow lost and need to be re-established.

6.5.4.5.1 SNMP Agent Wishes to Send with Missing SNMPv3 Keys.

In the case of SNMP, an SNMP agent is not responsible for re-establishing SNMPv3 keys because it does not send unsolicited requests to the Provisioning Server after the initial provisioning is done. Still, an SNMP agent could attempt to re-establish SNMPv3 keys after it gets an SNMPv3 authentication error back from the SNMP manager. If the SNMP agent determines that it has incorrect SNMPv3 keys, it must perform the following steps before it is able to send out an SNMP message:

1) The agent first makes sure that it has a valid Kerberos ticket for the Application Server. If not, it must first obtain it as specified in clause 6.5.2.

2) The agent sends a new AP Request to the manager and gets back an AP Reply, as specified in clause 6.5.2. After the receipt of the AP Reply the agent is prepared to use the newly created SNMPv3 keys. In this scenario, the SNMP manager must not set an ACK-required flag in the AP Reply. Right after sending out an AP Reply, the manager is prepared to both send and receive messages with the new SNMPv3 keys. After receiving this AP Request (with Re-establish flag = FALSE), the manager must remove its previous set of SNMPv3 keys that it might already have for this agent (and for this SNMPv3 user name).

It is possible that the SNMP manager already initiated key management (with a Wake Up) but instead receives an unsolicited AP Request from the agent (with server-nonce = 0). This unlikely scenario might occur if the manager and the agent decide to initiate key management at about the same time. In this case, the SNMP manager must ignore the unsolicited AP Request message and continue waiting for the one that is in response to a Wake Up.

6.5.4.5.2 SNMP Agent Receives with Missing SNMPv3 Keys

If SNMP agent receives a request from SNMP manager and is unable to find SNMPv3 keys for the specified USM User Name, the agent must process the SNMP message according to [28] and [38].

6.5.4.5.3 SNMP Manager Wishes to Send with Missing SNMPv3 Keys

SNMP manager attempts to send a message to the agent and does not find the desired user's SNMPv3 keys (or considers the existing SNMPv3 keys invalid or compromised). In this case, the following recovery steps must be taken at the key management layer:

1) Manager sends a Wake Up message to the agent.

2) The agent makes sure that it has a valid Kerberos ticket for the manager. If not, it must first obtain it from the KDC.

3) Agent sends a new AP Request to the manager, as specified in clause 6.5.2. For each AP Request, the agent generates a nonce and puts it into the seq-number field. As specified in clause 6.5.3.5.3, the agent will save this nonce for a period of time specified by the pktcMtaDevProvSolicitedKeyTimeout MIB object and wait for a matching AP Reply (this is not the same nonce as the server-nonce received in the Wake Up). However, after this timeout, the agent must not retry and must abort an attempt to establish SNMPv3 keys in response to a received Wake Up.

   Once the agent gets back a matching AP Reply, it will be in the format specified in clause 6.5.2. The ACK-required flag in the AP Reply must be set, to insure that the agent replies with the SA Recovered message in the following step.

   If this agent previously had SNMPv3 keys for the specified SNMPv3 user, they must be removed at this time.

4) After the receipt and validation of the AP Reply, the agent sends SA Recovered message to the manager. At this time the agent will be ready to use the new SNMPv3 keys and will enable SNMPv3 security.

5) Upon receipt of the SA Recovered message, the manager will immediately activate the new set of SNMPv3 keys and will enable SNMPv3 security.

It is possible that the SNMP agent already initiated key management (with an unsolicited AP Request) but instead receives a Wake Up from the manager. This unlikely scenario might occur if the manager and the agent decide to initiate key management at about the same time. In this case, the SNMP agent must abort waiting for the reply to the unsolicited AP Request message and instead generate a new AP Request in response to the Wake Up.

If an SNMP agent receives a second Wake Up message from a different SNMP manager (FQDN or IP address) before the first key management session has been completed, the SNMP agent must ignore the second Wake Up message.

6.5.4.6 SNMPv3-Specific Errors Returned in KRB_ERROR

Inside AppSpecificTypedData the oid field must be set to:

> enterprises (1.3.6.1.4.1) cableLabs (4491) clabProjects (2) clabProjPacketCable (2) pktcSecurity (4) errorCodes (1) snmpv3 (2).

The data-value field must correspond to the following typed-data value:

```
PktcKrbSnmpv3Error ::= SEQUENCE {
  e-code [0] INTEGER,
  e-text [1] GeneralString OPTIONAL,
  e-data [2] OCTET STRING OPTIONAL
}
```

The e-code field must correspond to one of the following error code values:

KRB_SNMPV3_ERR_USER_NAME	1	Unrecognized SNMPv3 user name
KRB_SNMPV3_ERR_NO_CIPHER	2	No support for requested ciphersuites
KRB_ SNMPV3_ERR_ENGINE_ID	3	Invalid SNMPv3 Engine ID Specified
KRB_ SNMPV3_ERROR_GENERIC	16	Generic KRB SNMPv3 error

The optional e-text field can be used for informational purposes (i.e. logging, network troubleshooting) and the optional e-data field is reserved for future use to transport any application data associated with a specific error.

6.6 End-to-End Security for RTP

RTP security is currently fully specified in clause 7.6.2.1. 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 14: RTP Packet Transform Identifiers

Transform ID	Value	Key Size (in bits)	Must Support	Description
RTP_ENCR_NULL	0x50	N/A	yes	Encryption turned off
RTP_AES	0x51	128	yes	AES-128 in CBC mode with 128-bit block size
RTP_XDESX_CBC	0x53	192	no	DESX-XEX-CBC
RTP_DES_CBC_PAD	0X54	128	no	DES-CBC-PAD
RTP_3DES_CBC	0X56	128	no	3DES-EDE-CBC
reserved	0x57-59	-	-	

The RTP_AES and RTP_ENCR_NULL Transform IDs must be supported. AES-128 [35] must be used in CBC mode with a 128-bit block size and an Initialization Vector (IV) generated in accordance with clause 7.6.2.1.2.2.2. AES-128 requires 10 rounds of cryptographic operations [35].

Table 15: RTP IPCablecom Authentication Algorithms

Authentication Algorithm	Value	Key Size (in bits)	Must Support	Description
AUTH_NULL	0x60	0	yes	Authentication turned off
reserved	0x61	-	-	
RTP_MMH_2	0x62	variable (see clause 7.6.2.1.2.1.1)	yes	2-byte MMH MAC
reserved	0x63	-	-	
RTP_MMH_4	0x64	variable (see clause 7.6.2.1.2.1.1)	yes	4-byte MMH MAC
reserved	0x65	-	-	

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

6.7 End-to-End Security for RTCP

RTCP security is currently fully specified in clause 7.6.2.2. Key Management for RTCP requires that both the (encryption) Transform ID and the Authentication Algorithm be specified. This clause lists the Transform IDs and Authentication Algorithms that are available for RTCP security.

Table 16: RTCP Packet Transform Identifiers

Transform ID	Value	Key Size (in bits)	Must Support	Description
RTCP_ENCR_NULL	0x70	0	yes	Encryption turned off.
AES-CBC	0x71	128	yes	AES-128 in CBC mode with 128-bit block size
XDESX-CBC	0x72	192	no	DESX-XEX-CBC
DES-CBC-PAD	0x73	128	no	DES-CBC-PAD
3DES-CBC	0x74	128	no	3DES-EDE-CBC
reserved	0x75-7f	-	-	

The AES-CBC and RTCP_ENCR_NULL Transform IDs must be supported. AES-128 [35] must be used in CBC mode with a 128-bit block size and a randomly generated Initialization Vector (IV). AES-128 requires 10 rounds of cryptographic operations [35].

Table 17: RTCP Authentication Algorithms

Transform ID	Value	Key Size (in bits)	Must Support	Description
RTCP_AUTH_NULL	0x80	N/A	yes	Authentication turned off
HMAC-SHA1-96	0x81	160	yes	First 12 bytes of the HMAC-SHA1 as described in [23]
HMAC-MD5-96	0x82	128	no	First 12 bytes of the HMAC-MD5 as described in [37]
reserved	0x83-8f	-	-	

The HMAC-SHA1-96 and RTCP_AUTH_NULL authentication algorithm must be supported.

6.8 BPI+

All E-MTAs and S-MTAs must use DOCSIS® 1.1 compliant cable modems that implement BPI+ [9]. Baseline Privacy Plus (BPI+) provides security services to the DOCSIS® 1.1 data link layer traffic flows running across the cable access network, i.e. between CM and CMTS. These services are message confidentiality and access control. The BPI+ security services operating in conjunction with DOCSIS® 1.1 provide cable modem users with data privacy across the cable network and protect cable operators from theft of service.

The protected DOCSIS® 1.1 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.

When employing BPI+, the CMTS 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 DOCSIS® MAC management messages that CMs use to establish QoS service flows. BPI+ employs a client/server key management protocol in which the CMTS (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 DOCSIS® 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 DOCSIS® MAC frame's packet data.
- A key management protocol (Baseline Privacy Key Management or "BPKM" provides the secure distribution of keying data from CMTS to CMs. Through this key management protocol, CM and CMTS synchronize keying data; in addition, the CMTS uses the protocol to enforce conditional access to network services.

Baseline Privacy Plus does not provide any security services beyond the DOCSIS® 1.1 cable access network. The majority of IPCablecom signalling and media traffic flows, however, take paths that traverse the managed IP "back haul" networks, which lie behind CMTSs. Since DOCSIS® 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 Client media flows, at the application/transport layer through RTP application layer security. Thus, IPCablecom does not rely on BPI+ to provide security services to its component protocol interfaces.

6.9 TLS

6.9.1 Overview

The TLS protocol [17] provides privacy and data integrity over a reliable transport layer protocol such as TCP. The protocol is composed of two layers: the TLS Record Protocol and the TLS Handshake Protocol. The TLS Record Protocol is used to securely encapsulate upper layer protocols, while the TLS Handshake Protocol provides the key management functionality required to establish TLS sessions.

In IPCablecom, TLS is used to secure SIP based signalling between SIP endpoints such as the CMS and EBPs.

6.9.2 IPCablecom Profile for TLS with SIP

Unless specified within the present document, IPCablecom SIP interfaces requiring TLS must be compliant with the TLS specification [17] and any requirements specified in [40] relating to its usage in SIP.

TLS [17] supports the negotiation and use of compression methods. However, since these methods are not specified within TLS [17], compression must not be used in IPCablecom.

6.9.2.1 TLS Ciphersuites

In TLS, the ciphersuite includes the authenticated key agreement (AKE) method used in the TLS handshake, as well as encryption and authentication ciphers used to secure the record layer. Ciphersuites are negotiated with the TLS client presenting a list of supported ciphersuites in the Client Hello message, and the server responding with the selected ciphersuite in the Server Hello message.

Table 18 describes the TLS ciphersuites defined in [17] and [41] supported by IPCablecom:

Table 18: TLS Ciphersuites

TLS Ciphersuite	Support	AKE Method	Encryption	Auth.
TLS_RSA_WITH_AES_128_CBC_SHA	must	RSA	AES-128 CBC	SHA
TLS_DHE_RSA_WITH_AES_128_CBC_SHA	must	Ephemeral Diffie-Hellman with RSA signatures	AES-128 CBC	SHA
TLS_RSA_WITH_3DES_EDE_CBC_SHA	should	RSA	3DES CBC	SHA
TLS_DHE_RSA_WITH_3DES_EDE_CBC_SHA	should	Ephemeral Diffie-Hellman with RSA signatures	3DES CBC	SHA

6.9.2.2 IPCablecom TLS Certificates

TLS is a client-server based protocol with optional client authentication. However, in IPCablecom, mutual authentication using RSA based certificates must be used. The TLS server must send a Certificate Request to the client. Both the TLS client and server certificates must conform to the IPCablecom Server Certificate as specified in clause 8.2.3.4.4.

IPCablecom Server Certificates include a server identifier (based on FQDN or IP address) embedded within the CN of the Subject Name field. Before accepting or continuing with a TLS connection, the TLS server or client must validate the remote server identifier to ensure it matches the IP address used for the TCP/TLS connection, in addition to any other local policy (i.e. provisioned list of allowed remote TLS endpoints based on FQDN or IP address).

In addition to the CableLabs® Service Provider Root certificate, a TLS implementation may support a list of trusted CAs (Certificate Authorities) to facilitate inter-working between IPCablecom domains (i.e. between Server Providers).

6.9.2.3 Connection Persistence and Re-Use

Since TCP connection and TLS session establishment (which relies on TCP) can be quite costly both in terms of performance and network latency, they are not suited for on-demand SIP signalling. As such, TLS sessions should be kept persistent as much as possible and SIP connection re-use should be supported.

6.9.2.4 Session Caching

In TLS, it is possible to resume a previous session if it has been cached on both the TLS client and server. Resuming sessions drastically speeds up the session establishment, as fewer messages are exchanged and authentication is based on symmetric key cryptography.

In IPCablecom, TLS session caching should be supported. A TLS client initiating a TLS session must attempt to resume a cached session if it has retained a session for the remote server. The duration for which a TLS client or server must retain a cached session is a local policy and implementation specific.

7 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 5.2 summarizes 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 security analysis in clause 5.3.3 is organized by functional categories. 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):

Table 19: RTP - RTCP Security Profile Matrix

	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 keys distributed over secured MTA-CMS links AES-128 encryption algorithm Optional 2-byte or 4-byte MAC based on MMH algorithm IPCablecom supports ciphersuite negotiation.	Application Layer Security via RTCP IPCablecom Security Profile keys distributed over secured MTA-CMS links RTCP ciphersuites are negotiated separately from the RTP ciphersuites and include both encryption and message authentication algorithms. Keys are derived from the end-end secret using the same mechanism as used for RTP encryption

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 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.

7.1 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 [4].

Figure 10 illustrates only the flows involved with the Secure provisioning processes. The provisioning specification lays out in detail these Secure Provisioning flows along with two non-secure MTA provisioning flows called Basic and Hybrid. The Secure Provisioning flows involving security mechanisms are described in this clause. Refer to the provisioning specification for the non-secure flows [4].

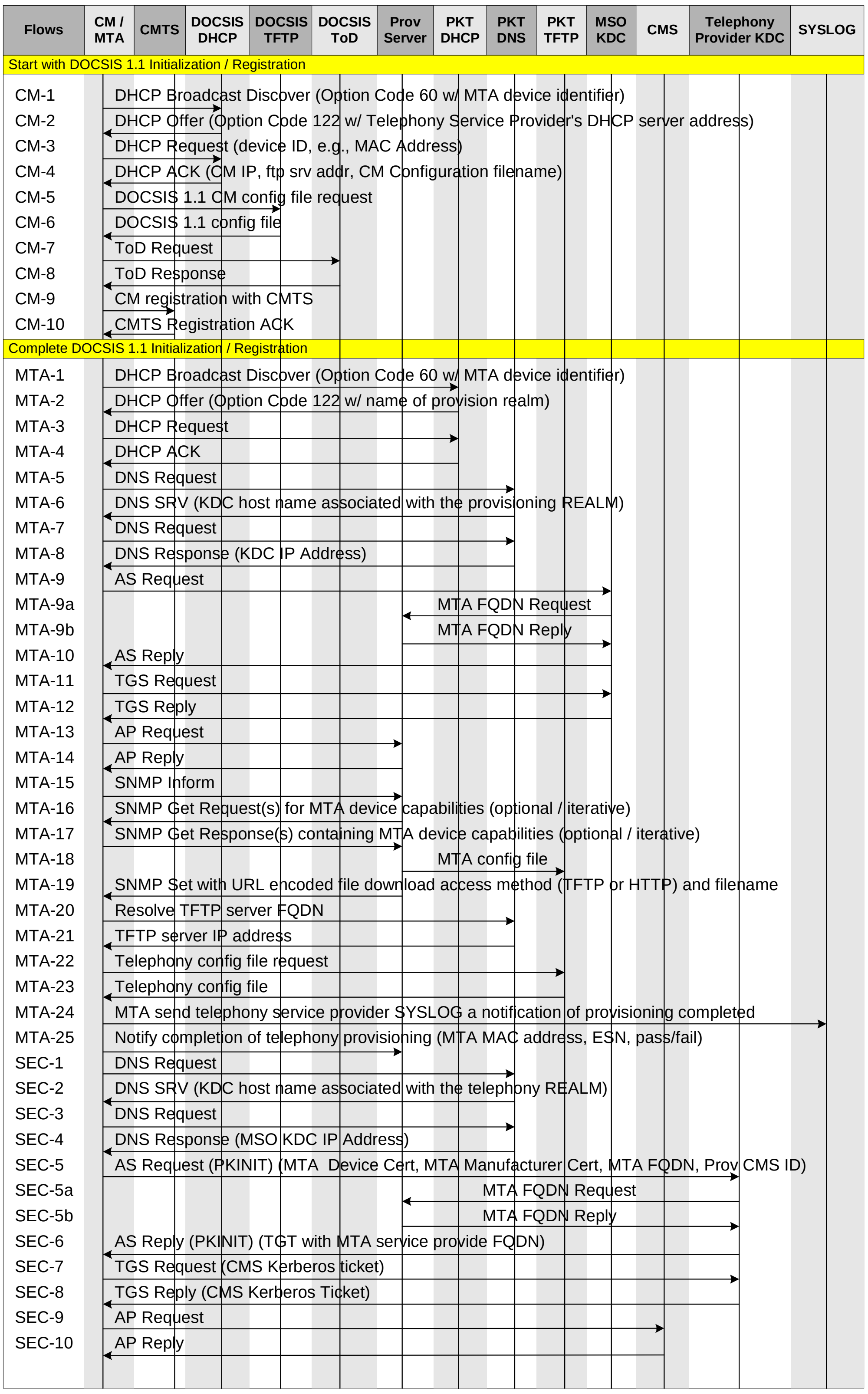

Flows
CM / MTA
CMTS
DOCSIS DHCP
DOCSIS TFTP
DOCSIS ToD
Prov Server
PKT DHCP
PKT DNS
PKT TFTP
MSO KDC
CMS
Telephony Provider KDC
SYSLOG
Start with DOCSIS 1.1 Initialization / Registration
CM-1 DHCP Broadcast Discover (Option Code 60 w/ MTA device identifier)
CM-2 DHCP Offer (Option Code 122 w/ Telephony Service Provider's DHCP server address)
CM-3 DHCP Request (device ID, e.g., MAC Address)
CM-4 DHCP ACK (CM IP, ftp srv addr, CM Configuration filename)
CM-5 DOCSIS 1.1 CM config file request
CM-6 DOCSIS 1.1 config file
CM-7 ToD Request
CM-8 ToD Response
CM-9 CM registration with CMTS
CM-10 CMTS Registration ACK
Complete DOCSIS 1.1 Initialization / Registration
MTA-1 DHCP Broadcast Discover (Option Code 60 w/ MTA device identifier)
MTA-2 DHCP Offer (Option Code 122 w/ name of provision realm)
MTA-3 DHCP Request
MTA-4 DHCP ACK
MTA-5 DNS Request
MTA-6 DNS SRV (KDC host name associated with the provisioning REALM)
MTA-7 DNS Request
MTA-8 DNS Response (KDC IP Address)
MTA-9 AS Request
MTA-9a MTA FQDN Request
MTA-9b MTA FQDN Reply
MTA-10 AS Reply
MTA-11 TGS Request
MTA-12 TGS Reply
MTA-13 AP Request
MTA-14 AP Reply
MTA-15 SNMP Inform
MTA-16 SNMP Get Request(s) for MTA device capabilities (optional / iterative)
MTA-17 SNMP Get Response(s) containing MTA device capabilities (optional / iterative)
MTA-18 MTA config file
MTA-19 SNMP Set with URL encoded file download access method (TFTP or HTTP) and filename
MTA-20 Resolve TFTP server FQDN
MTA-21 TFTP server IP address
MTA-22 Telephony config file request
MTA-23 Telephony config file
MTA-24 MTA send telephony service provider SYSLOG a notification of provisioning completed
MTA-25 Notify completion of telephony provisioning (MTA MAC address, ESN, pass/fail)
SEC-1 DNS Request
SEC-2 DNS SRV (KDC host name associated with the telephony REALM)
SEC-3 DNS Request
SEC-4 DNS Response (MSO KDC IP Address)
SEC-5 AS Request (PKINIT) (MTA Device Cert, MTA Manufacturer Cert, MTA FQDN, Prov CMS ID)
SEC-5a MTA FQDN Request
SEC-5b MTA FQDN Reply
SEC-6 AS Reply (PKINIT) (TGT with MTA service provide FQDN)
SEC-7 TGS Request (CMS Kerberos ticket)
SEC-8 TGS Reply (CMS Kerberos Ticket)
SEC-9 AP Request
SEC-10 AP Reply

Figure 10: IPCablecom Provisioning Flows

As part of the provisioning process, the MTA performs Kerberos key management (AS Request/AS Reply and AP Request/AP Reply, and optional TGS Request/TGS Reply).

Table 20 describes the execution of the Kerberos key management step during MTA Provisioning.

Table 20: Kerberos Key Management During MTA Provisioning

Flow Step	Security Requirement	Life Time	Step Bypass Permitted
MTA-9/MTA-10 - AS Request/AS Reply (see clause 6.4.1)	TGT ticket if using TGS Request, Provisioning Server Ticket if otherwise.	Max. 7 days.	This step must not be performed if the MTA already possesses a valid ticket for the Provisioning Server.
MTA-11/MTA 12 - TGS Request/TGS Reply (see clause 6.4.4)	Applies when a TGT is used. Obtains a Provisioning Server Ticket.	Lifetime set to expire no later than the expiration time of the TGT ticket.	This step must not be performed if the MTA already possesses a valid ticket for the Provisioning Server.
MTA-9a/MTA-9b - MTA FQDN Request/MTA FQDN Reply (see clause 6.4.7)	MTA FQDN Request and Reply are protected using Kerberos tickets.		These steps will not occur if MTA-9 is skipped. Otherwise, this step cannot be bypassed.
MTA-13/MTA-14 - AP Request/AP Reply (see clause 6.5.2 and clause 6.5.4)	Initial SNMPv3 authentication and privacy keys for the MTA. The user name for the MTA is specified as "MTA-Prov-xx:xx:xx:xx:xx:xx". Where xx:xx:xx:xx:xx:xx represents the MAC address of the MTA. AP Req /AP Rep messages do not specify the SNMPv3 key expiration time in the protocol, but the SNMP Manager may still set up expiration time locally; after the keys expire the manager can send a Wake Up message to create a new set of SNMPv3 keys.	Expiration is not supported by IPCablecom.	None - new SNMPv3 keys and User Ids are created each time the MTA is reinitialized. It is assumed that SNMPv3 keys and User Ids are not saved in NVRAM. Also note that this step is used for Engine ID determination and SNMPv3 time synchronization - the two sides exchange initial values for SNMPv3 boots and engine time parameters.

An MTA must get a new ticket before performing Kerberized Key Management with a particular Application Server if the ticket(s) it currently possesses is not valid. A ticket would no longer be valid if the KDC REALM or Application Server FQDN changes, if the MTA's IP address has changed, or if the current time, adjusted by the time offset for that REALM or Application Server, does not fall within the ticket validity period.

The PKINIT$_{GP}$ for the Provisioning Server's realm is specified in the MTA MIB inside the realm table. When the MTA implementation requests a TGT in an AS Request and when the MTA needs to obtain tickets for one or more CMSs in the same realm as the Provisioning Server, the PKINIT$_{GP}$ value specified in the MIB must be used to refresh the TGT. In all other cases, the AS Request for the TGT in the Provisioning Server's realm or for the Provisioning Server's ticket directly may be issued on-demand.

The TGS Grace Period is not specified for the key management between the MTA and the Provisioning Server. The TGS Request for the Provisioning Server's ticket may be issued on-demand.

7.1.1 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.

As defined in [4], the IPCablecom architecture supports three provisioning flow:

- Basic Flow.
- Hybrid Flow.
- Secure Flow.

The Basic and Hybrid Flows are completely insecure flows (i.e. there are no mechanisms in the flows that would prevent a user from provisioning their own MTA). The Basic and Hybrid Flows also do not provide a means to secure the SNMP management interface on the MTA. Service providers that choose to deploy MTAs with one of these insecure flows must accept that there are security risks. For example, a Denial-of-Service attack could be mounted by sending SNMP TRAPs and INFORMs to the cable operator's management system. The management system would have to process them, even though they are unauthenticated. Unfortunately, the inclusion of these insecure flows also poses security risks for Service Providers that choose to deploy MTAs with the Secure Flow.

MTAs that support the insecure flows may be provisioned by a user, even if the service provider is using the Secure Flows. Unauthorized provisioning of an MTA allows a user to provide their own configuration file. The MTA could then be used to communicate normally with a CMS. Alternatively, un-authorized provisioning of an MTA could be used to bypass service provider controls on secure software download (in the case of the S-MTA) and provide a software image that has some perceived value (such as security vulnerability).

With respect to the Secure Flow, support for SNMPv2c coexistence for network management operations also introduces vulnerabilities to service providers that use the Secure Flow (unauthenticated TRAPs and INFORMs could be sent). The best way to address these vulnerabilities is to disable SNMPv2c coexistence.

Therefore it is recommended, as always, that service providers use multiple layers of security to ensure that their CMSs and back-office systems are protected against rogue MTAs.

7.1.1.1 Security Services

7.1.1.1.1 MTA-DHCP Server

Authentication and Message Integrity is desirable on this interface, in order to prevent denial-of-service attacks that cause an MTA to be improperly configured. Securing DHCP is considered an operational issue to be evaluated by each network operator. It is possible to use access control through the local DHCP relay inside the local loop. IPsec can be used for security between the DHCP relay and the DHCP server.

7.1.1.1.2 MTA-SNMP Manager

This clause applies to all SNMPv3 messages between the MTA and an SNMPv3 Manager. Within the IPCablecom architecture, the Provisioning Server includes the SNMPv3 Manager function, although SNMPv3 traffic occurs both during and after the provisioning phase.

Authentication: the identity of the MTA that is sending configuration parameters and faults to the SNMP manager must be authenticated, to prevent denial of service attacks. For example, the Provisioning Server may be tricked into continuously creating bogus configuration files or into creating a configuration file based on incorrect MTA capabilities that in effect disable that MTA.

Also, during the provisioning sequence the MTA is told (via an SNMP Set) the parameters needed to find, authenticate and decrypt its configuration file. If this SNMP Set were forged, it would disrupt the MTA provisioning sequence.

Message Integrity: 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: may be used to protect sensitive MTA configuration data. IPCablecom currently does not specify any such sensitive MTA parameters and so confidentiality is optional.

Access Control: write access to the MTA configuration parameters must be allowed only to the authorized OSS users, to prevent denial of service/misconfiguration attacks. Read access can be enforced in conjunction with confidentiality, which is optional (see above on confidentiality).

Note that DHCP is used to configure the MTA with the Kerberos realm name, which points it to a particular KDC. DHCP also configures the MTA with the location of the Provisioning Server. Since IPCablecom currently does not specify DHCP security, by faking DHCP responses it is possible to point MTAs to a wrong Provisioning Server and to a wrong KDC that permits security establishment with that Provisioning Server. (The MTA would only authenticate that wrong KDC if the CableLabs® Service Provider Root CA signed the KDC certificate.) So, it is possible to bypass access control, but the attack has to be orchestrated by another cable operator that had also been certified by IPCablecom.

7.1.1.1.3 MTA-Provisioning Server, via TFTP Server

Authentication: required to prevent denial-of-service attacks that cause an MTA to be improperly configured.

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

Confidentiality: optional, it is up to the Provisioning Server to decide whether or not to encrypt the file.

Access Control: not required at the TFTP Server. If needed, MTA configuration file is encrypted with the Provisioning Server-MTA shared key.

Non-Repudiation: not required.

7.1.1.2 Cryptographic Mechanisms

7.1.1.2.1 Call Flows MTA-15, 16, 17: MTA-SNMP Manager: SNMP Inform/Get Requests/Responses

All SNMP traffic between the MTA and the SNMP Manager in both directions is protected with SNMPv3 security [28] during the Secure Provisioning process. IPCablecom requires that SNMPv3 message authentication is always turned on with privacy being optional (see clause 6.3). The only SNMPv3 encryption algorithm is currently DES-CBC. This is the limitation of the SNMPv3 IETF standard, although stronger encryption algorithms are desirable. See clause 6.3 for the list of SNMPv3 cryptographic algorithms supported by IPCablecom.

7.1.1.2.2 Call Flow MTA-18: Provisioning Server-TFTP Server: Create MTA Config File

This clause describes the MTA Config file creation in the Secure Provisioning Flow. In this flow, the Provisioning Server builds an MTA device configuration file. This file must contain the following configuration info for each endpoint (port) in the MTA:

- CMS name (FQDN format)
- Kerberos Realm for this CMS
- Telephony Service Provider Organization Name
- PKINIT Grace Period

This file must be authenticated and may be encrypted. If the configuration file is encrypted then the SNMPv3 privacy must be used in order to transport the configuration file encryption key securely. Once the Provisioning Server builds the configuration file, it will perform the following steps:

1) The Provisioning Server decides to encrypt the file, it creates a configuration file encryption key and encrypts the file with this key. The encryption algorithm must be the same as the one that is used for SNMPv3 privacy. It then stores the key and the cipher. The file must be encrypted using the following procedure:

    a) prepend the file contents with a random byte sequence, called a confounder. The size of the confounder must be the same as the block size for the encryption algorithm. In the case of DES it is 8 bytes.

    b) append random padding to the result in (a). The output of this step is of length that is a multiple of the block size for the encryption algorithm.

    c) encrypt the result in (b) using IV=0. The output of this step is the encrypted configuration file.

2) It creates a SHA-1 hash of the configuration file and stores it. If the file was encrypted, the hash is taken over the encrypted file.

3) It sends the following items to the MTA in the SNMP SET in the flow MTA-19:

    a) pktcMtaDevConfigKey, which is the configuration file encryption key MIB variable generated in step 1.

    b) pktcMtaDevConfigHash, which is the SHA-1 of the configuration file MIB variable generated in step 2.

    c) Name and location of the configuration file.

Steps 1 and 2 must occur only when a configuration file is created or an existing file is modified. If the pktcMtaDevConfigKey is set, then the MTA must use this key to decrypt the configuration file. Otherwise, MTA must assume that the file is not encrypted. SNMPv3 provides authentication when the pktcMtaDevConfigHash is set and therefore the configuration file is authenticated indirectly via SNMPv3.

In the event that SNMPv3 privacy is selected during the key management phase, but is using a different algorithm than the one that was selected to encrypt the configuration file (or the configuration file was previously in the clear), the configuration file must be re-encrypted and the TFTP server directory must be updated with the new file. Similarly, if the Provisioning Server decides not to encrypt the file this time, after it was previously encrypted, the TFTP server directory must be updated with the new file.

MTA endpoints may also be configured for IP Telephony service while the MTA is operational. In that case the same information that is normally assigned to an endpoint in a configuration file must be assigned with SNMP Set commands.

7.1.1.2.3 Call Flows MTA-19, 20 and 21: Establish TFTP Server Location

This set of call flows is used to establish the IP address of the TFTP server from where the MTA will retrieve its configuration file. Although flow MTA-19 is authenticated via SNMPv3, MTA-20 and 21 are not authenticated.

Flow MTA-21 allows for denial-of-service attacks, where the MTA is pointed to a wrong TFTP server (IP address). The MTA cannot be fooled in accepting the wrong configuration file since checking the hash of the file authenticates the file - this denial-of-service attack will result in failed MTA provisioning.

The denial-of-service threats, where responses to DNS queries are forged, are currently not addressed by IPCablecom. It is mainly because DNS security (DNSSEC) is not yet available as a commercial product and would cause significant operational difficulty in the conversion of the DNS databases.

7.1.1.2.4 Call Flows MTA-22, 23: MTA-TFTP Server: TFTP Get/Get Response

The TFTP get request is not authenticated and thus anyone can request an MTA configuration file. This file does not contain any sensitive data and may be encrypted with the Provisioning Server-MTA shared key if the Provisioning Server chooses to. In this case no one except the MTA 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 contents of the configuration file are listed in clause 7.1.1.2.2.

7.1.1.2.5 Security Flows

For each CMS specified in the pktcMtaDevCmsTable table with pktcMtaDevCmsIpsecCtrl value set to true and assigned to a provisioned MTA endpoint, the MTA must perform the following security flows after the provisioning process and prior to any NCS message exchange. For each CMS specified in pktcMtaDevCmsTable with pktcMtaDevCmsIpsecCtrl set to false, the MTA must not perform the following flows and must send and receive NCS messages without IPsec (i.e. NCS packets are sent in the "clear").

Table 21: Post-MTA Provisioning Security Flows

Sec Flow	Flow Description	If Step Fails, Proceed Here
Get Kerberos tickets associated with each CMS with which the MTA communicates.		
SEC-1	DNS SRV Request The MTA requests the Telephony KDC host name for the Kerberos realm. This step must not be performed if the MTA already possesses a valid ticket for the CMS.	SEC-1
SEC-2	DNS SRV Reply Returns the Telephony KDC host name associated with the provisioning REALM. If the KDC's IP Address is included in the Reply, proceed to SEC-5. This step must not be performed if the MTA already possesses a valid ticket for the CMS.	SEC-1
SEC-3	DNS Request The MTA now requests the IP Address of the Telephony KDC. This step must not be performed if the MTA already possesses a valid ticket for the CMS.	SEC-1
SEC-4	DNS Reply The DNS Server returns the IP Address of the Telephony KDC. This step must not be performed if the MTA already possesses a valid ticket for the CMS.	SEC-1
SEC-5	AS Request For each different CMS assigned to voice communications endpoints, the MTA requests a TGT or a Kerberos Ticket for the CMS by sending a PKINIT REQUEST message to the KDC. This request contains the MTA Device Certificate and the MTA FQDN. This step must not be performed if the MTA already possesses a valid ticket for the CMS.	Report alarm. Abort establishment of signalling security.
SEC-5a	MTA FQDN Request The KDC requests the MTA's FQDN from the Provisioning Server. This step will not occur if the MTA skips SEC-5.	
SEC-5b	MTA FQDN Reply The Provisioning Server replies to the KDC request with the MTA's FQDN. This step will not occur if the MTA skips SEC-5.	
SEC-6	AS Reply The KDC sends the MTA a PKINIT REPLY message containing the requested Kerberos Ticket. This step must not be performed if the MTA already possesses a valid ticket for the CMS.	Proceed to SEC-5 or abort signalling security depending upon error conditions.
SEC-7	TGS Request In the case where the MTA obtained a TGT in SEC-6, it now obtains the Kerberos ticket for the TGS request message. This step must not be performed if the MTA already possesses a valid ticket for the CMS.	Report alarm. Abort establishment of signalling security.
SEC-8	TGS Reply Response to TGS Request containing the requested CMS Kerberos Ticket. This step must not be performed if the MTA already possesses a valid ticket for the CMS.	Proceed to SEC-7/SEC-5 or abort signalling security depending upon error conditions.
SEC-9	AP Request The MTA requests a pair of IPsec simplex Security Associations (inbound and outbound) with the assigned CMS by sending the assigned CMS an AP REQUEST message containing the CMS Kerberos Ticket.	Report alarm. Abort establishment of signalling security.
SEC-10	AP Reply The CMS establishes the Security Associations and then sends an AP REPLY message with the corresponding IPsec parameters. The MTA derives IPsec keys from the subkey in the AP Reply and establishes IPsec SAs.	Proceed to SEC-9/SEC-7/SEC-5 or abort signalling security depending upon error conditions.

Several tables in the MTA MIB are used to control security flows SEC-1 through SEC-10 (see table 21).

The CMS table (pktcMtaDevCmsTable) and the realm table (pktcMtaDevRealmTable) are used for managing the MTA security signalling. The realm table defines the domains for the CMSs. The CMS table defines the CMSs within the domains. An endpoint is associated with one CMS at any given time. The following restrictions must be adhered to:

a) The realm table in the configuration file must at a minimum include an entry for the realm that is identified in DHCP option 122, suboption 6.

b) There must be a realm table entry for each CMS table entry. Multiple CMS table entries may utilize the same realm table entry.

c) Each MTA endpoint defined in the NCS endpoint table (pktcNcsEndPntConfigTable) must be configured with a CMS FQDN (pktcNcsEndPntConfigCallAgentId) that is also present in the CMS table (pktcMtaDevCmsFqdn).

d) All members of a CMS cluster defined by the same FQDN must use the same configuration for establishing Security Associations as defined in pktcMtaDevCmsTable.

e) If NCS signalling selects a CMS (with an N: parameter selection) that is not defined by an entry in the CMS table, the same realm and CMS parameters, with the exception of the CMS FQDN and pktcMtaDevCmsIpsecCtrl, are used as defined in the current CMS table entry. The pktcMtaDevCmsIpsecCtrl flag for the new CMS must be set to true.

The use of the security-relevant MIB tables immediately following step MTA-25 is as follows:

1) The MTA finds a list of CMSs with which it needs to establish IPsec SAs. This list must include every CMS that is assigned to a configured endpoint, as specified by the NCS MIB table pktcNcsEndPointConfigTable. This list of CMSs must include only CMSs that are listed in the pktcMtaDevCmsTable.

2) For each CMS in the above list, the MTA must attempt to establish IPsec Security Associations as follows:

   a) Find the corresponding CMS table entry.

   b) If the MTA does not already possess a valid ticket for the specified CMS, use the pktcMtaDevCmsKerbRealmName parameter in the CMS table entry to index into pktcMtaDevRealmTable. Then, using the parameters associated with that realm, perform steps SEC-1 through SEC-6 and optionally SEC-7 and SEC-8 in order to obtain the desired CMS ticket.

   c) Perform IPsec key management according to flows SEC-9 and SEC-10. This step may occur at any time after step b. above, but it must occur before any signalling messages are exchanged with that CMS.

   The CMS table entry contains various timing parameters used in steps SEC-9 and SEC-10. In the case of time outs or other errors, the MTA may retry using the timing parameters specified in the CMS table entry.

   The above steps must also apply when an additional MTA endpoint is activated (see [4]) or when an endpoint is configured (via SNMP sets) for a new CMS in the NCS MIB (see [26] and [47]).

3) Any time before an MTA endpoint sends a signalling message to a particular CMS, it must ensure that the respective Security Association is present. If the MTA is unable to establish IPsec SAs with a CMS that is associated with a configured endpoint (by the NCS MIB), it must set the NCS MIB variable pktcNcsEndPntStatusError to noSecurityAssociation (2).

After the initial establishment of the IPsec Security Associations for CMSs, the MTA MIB is utilized in subsequent key management as follows:

When the MTA receives a Wake Up message, it must respond with an AP Request when the corresponding CMS FQDN is found in the pktcMtaDevCmsTable and must not respond otherwise.

NOTE: Establishment of IPsec Security Associations due to a Wake Up does not result in any call signalling traffic between the MTA and the CMS.

7.1.1.2.5.1 Call Flows SEC-5,6: Get a Kerberos Ticket for the CMS

The MTA uses PKINIT protocol to get a Kerberos Ticket for the specified CMS (see clause 6.4.3). After the KDC receives a ticket request, it retrieves the MTA FQDN from the provisioning server so that it can verify the request before replying with a ticket. The Telephony KDC issues the Kerberos Ticket 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 CMS. Note that there is no requirement that the MTA obtain all the tickets from a single KDC.

7.1.1.2.5.2 Call Flows SEC-7,8,9: 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 pairs of SAs will be established (one set for each CMS).

When a single Kerberos ticket is issued for clustered Call Agents, it is used to establish more than one pair of IPsec SAs.

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 initiate Kerberized key management 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).

7.1.1.3 Key Management

7.1.1.3.1 MTA - SNMP Manager

Key Management for the MTA-Provisioning SNMPv3 user must use the Kerberized key management protocol as it is specified in clause 6.5.4. The MTA and the Provisioning Server must support this key management protocol. Additional SNMPv3 users may be created with the standard SNMPv3 cloning method [28] or with the same Kerberized key management protocol.

In order to perform Kerberized key management, the MTA must first locate the KDC. It retrieves the provisioning realm name from DHCP and then uses a DNS SRV record lookup to find the KDC FQDN(s) based on the realm name (see clause 6.4.5.1). When there is more than one KDC (DNS SRV record) found, DNS assigns a priority (and possibly a weighting) to each one. The MTA will choose a KDC based on the DNS priority and weight labelling and will go through the list until it finds a KDC that is able to respond.

7.1.1.3.2 MTA - TFTP Server

The optional encryption key for the MTA configuration file is passed to the MTA with an SNMP Set command (by the Provisioning Server) shown in the provisioning flow MTA-19. SNMPv3 security is utilized to provide message integrity and privacy. In the event that SNMPv3 privacy is not enabled, the MTA configuration file must not be encrypted and the file encryption key must not be passed to the MTA.

The encryption algorithm used to encrypt the file must be the same as the one used for SNMPv3 privacy. The same file encryption key may be re-used on the same configuration file while the MTA configuration file contents are unchanged. However, if the MTA configuration file changes or if a different encryption algorithm is selected for SNMPv3 privacy, the Provisioning Server must generate a new encryption key, must re-encrypt the configuration file and must update the TFTP Server with the re-encrypted file.

7.1.1.4 MTA Embedded Keys

The MTA device must be manufactured with a public/private RSA key pair and an X.509 device certificate that must be different from the BPI+ device certificate.

7.1.1.5 Summary Security Profile Matrix - Device Provisioning

The following matrix applies only to the Secure Provisioning Flow and SNMPv3.

Table 22: Security Profile Matrix - MTA Device Provisioning

	SNMP	TFTP (MTA - TFTP server)
authentication	Yes	Yes: authentication of source of configuration data.
access control	Yes: write access to MTA configuration is limited to authorized SNMP users. Read access can also be limited to the valid users when confidentiality is enabled.	Yes: write access to the TFTP server must be limited to the Provisioning Server but is out of scope for IPCablecom. Read access can be optionally indirectly enabled when the MTA configuration file is encrypted.
integrity	Yes	Yes
confidentiality	Optional	Optional (of MTA configuration information during the TFTP-get).
non-repudiation	No	No
security mechanisms	SNMPv3 authentication and privacy. Kerberized key management protocol defined by IPCablecom.	Hash of the MTA configuration file is sent to the MTA over SNMPv3, providing file authentication. When the file is encrypted, the key is also sent to the MTA over SNMPv3 (with SNMPv3 encryption turned on).

7.1.2 Subscriber Enrolment

The subscriber enrolment process establishes a permanent customer billing account that uniquely identifies the MTA to the CMS via the endpoint ID, which contains the MTA's FQDN. 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 validate 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), 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 7.6 and to clause 6.6 on media streams.

7.2 Quality of Service (QoS) Signalling

7.2.1 Dynamic Quality of Service (DQoS)

7.2.1.1 Reference architecture for embedded MTAs

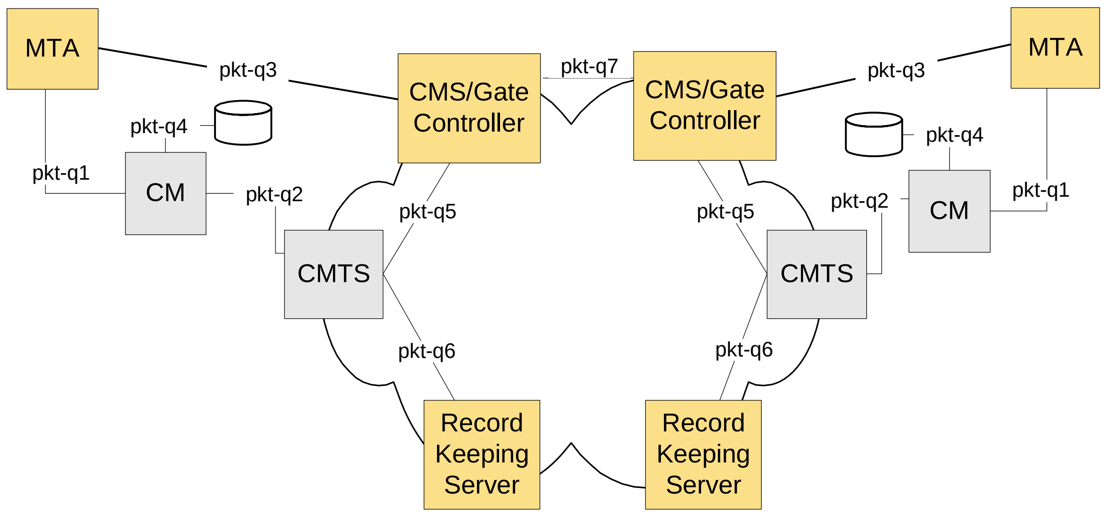


Figure 11: QoS Signalling Interfaces in IPCablecom Network

7.2.1.2 Security Services

7.2.1.2.1 CM-CMTS DOCSIS® 1.1 QoS Messages

Refer to the DOCSIS® 1.1 RFI specification [8].

7.2.1.2.2 Gate Controller - CMTS 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.

7.2.1.3 Cryptographic Mechanisms

7.2.1.3.1 CM-CMTS DOCSIS® 1.1 QoS Messages

The DOCSIS® 1.1 QoS messages are specified in the DOCSIS® 1.1 RFI specification [8].

7.2.1.3.1.1 QoS Service Flow

A Service Flow is a DOCSIS® MAC-layer transport service that provides unidirectional transport of packets either to upstream packets transmitted by the CM or to downstream packets transmitted by the CMTS. A service 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 CMTS, these attributes include details of how the CM requests mini-slots and the expected behaviour of the CMTS upstream scheduler.

DOCSIS® defines a Classifier, which consists of some packet matching criteria (IP source address, for example), a Classifier priority, and a reference to a service flow. If a packet matches the specified packet matching criteria, it is then delivered on the referenced service flow.

Downstream Classifiers are applied by the CMTS to packets it is transmitting, and Upstream Classifiers are applied at the CM and may be applied at the CMTS 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. A CMTS controlling downstream service flows will limit an MTA's downstream bandwidth according to QoS allocations. If the CMTS 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 to flood the network with unauthorized IP packets.

CMTSs should use classifiers to police upstream packets (including verifying source IP addresses) arriving over the HFC access network.

For more information regarding the use of packet Classifiers, refer to the DOCSIS® 1.1 RFI specification [8].

7.2.1.3.2 Gate Controller - CMTS COPS Messages

To download a QoS policy for a particular communications connection, the Gate Controller function in the CMS must send COPS messages to the CMTS. These COPS messages must be both authenticated and encrypted with IPsec ESP. Refer to clause 6.1.2 on the details of how IPsec ESP is used within IPCablecom and for the list of available ciphersuites.

7.2.1.4 Key Management

7.2.1.4.1 Gate Controller - CMTS COPS Messages

Key management for this COPS interface is either IKE or Kerberos. Implementations must support IKE with pre-shared keys. Implementations may support IKE with X.509 certificates and they may support Kerberos using symmetric keys. For more information on the IPCablecom use of IKE, refer to clause 6.2.2. For more information on the IPCablecom use of Kerberos with symmetric keys, refer to clauses 6.4.3 and 6.5.

When the Gate Controller detects a failure of all COPS connections associated with a particular outgoing IPsec SA, it must delete all associated SAs (IKE and IPsec SAs if IKE is used as the Key management protocol or only IPsec SAs if Kerberos is used as the Key management protocol).

Subsequently, every N times (1 ≤ N ≤ 10) that the Gate Controller tries to recover the connection, the SAs must be removed.

7.2.1.4.2 Security Profile Matrix Summary

Table 23: Security Profile Matrix - DQoS

	COPS (CMTS-CMS)
Authentication	Yes
access control	Yes
Integrity	Yes
Confidentiality	yes
non-repudiation	No
security mechanisms	IPsec with encryption and message integrity IKE or Kerberos

7.3 Billing System Interfaces

7.3.1 Security Services

7.3.1.1 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.

7.3.1.2 CMTS-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.

7.3.1.3 MGC - 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.

7.3.2 Cryptographic Mechanisms

Both message integrity and privacy must be provided by IPsec ESP, using any of the ciphersuites that are listed in clause 6.1.2.

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 16 ASCII 0s. 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 Element ID of the CMS, CMTS or the MGC. The RKS must verify each billing event by ensuring 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 7.3.3 on how this map is maintained. A combined element (such as a combined CMS/MGC) may use the same IP address and Security Association to convey Event Messages from both elements. Additionally, both elements may use the same Element ID. Refer to clause 7.3.3.1 for information on how to maintain a map of multiple elements and Element IDs.

7.3.2.1 RADIUS Server Chaining

RADIUS servers may be chained. This means that when the local RADIUS server that is directly talking to the CMS or CMTS 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.

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

7.3.3 Key Management

7.3.3.1 Key CMS - RKS Interface

The CMS and the RKS must negotiate a shared secret (CMS-RKS Secret) using IKE or Kerberos with symmetric keys (implementations must support IKE with pre-shared keys; they may support IKE with X.509 certificates and they may support Kerberos using symmetric keys). For more information on the IPCablecom use of IKE, refer to clause 6.2.2. For more information on the IPCablecom use of Kerberos with symmetric keys, refer to clauses 6.4.3 and 6.5.

The key management protocol must run asynchronous to billing event generation, and will guarantee that there is always a valid, non-expired CMS-RKS Secret.

An RKS must maintain a mapping between an IP address and an Element ID for each host with which it has IPsec Security Associations. How this mapping is created depends on the IPsec key management protocol:

1) IKE with Pre-Shared Keys. One way to implement this mapping is to provide a local database of which Element ID(s) are associated with the source IP address.

2) IKE with Certificates. As specified in clause 8.2.3.4.3, a certificate of a server that sends billing event messages to an RKS contains its Element ID(s) in the CN attribute of the distinguished name. During IKE phase 1, the RKS must save a mapping between the IP address and its Element ID(s) that is contained in the certificate.

3) Kerberized Key Management. As specified in clause 6.4.5.5, a principal name of each server that reports billing event messages to the RKS includes its Element ID(s). After an RKS receives and validates an AP Request message, it must save a mapping between the IP address and its Element ID(s) that is contained in the principal name.

When an event message arrives at the RKS, the RKS must retrieve a source IP address based on the Element ID, using the mapping established during key management. The RKS must ensure that this address is the same as the source IP address in the IP packet header.

7.3.3.2 CMTS - RKS Interface

The CMTS and the RKS must negotiate a shared secret (CMTS-RKS Secret) using IKE or Kerberos (implementations must support IKE with pre-shared keys; they may support IKE with X.509 certificates and they may support Kerberos using symmetric keys). For more information on the IPCablecom use of IKE, refer to clause 6.2.2. For more information on the IPCablecom use of Kerberos with symmetric keys, refer to clauses 6.4.3 and 6.5.

The key management protocol must be running asynchronous to billing event generation, and will guarantee that there is always a valid, non-expired CMTS-RKS Secret.

An RKS maintains a mapping between an IP address and an Element ID for each host with which it has IPsec Security Associations, as specified in clause 7.3.3.1. This includes the CMTS.

When a billing event arrives at the RKS, the RKS must retrieve a source IP address based on the Element ID, using the mapping established during key management. The RKS must ensure that this address is the same as the source IP address in the IP packet header

7.3.3.3 MGC - RKS Interface

The MGC and the RKS must negotiate a shared secret (MGC-RKS Secret) using IKE or Kerberos (implementations must support IKE with pre-shared keys; they may support IKE with X.509 certificates and they may support Kerberos using pre-shared keys). For more information on the IPCablecom use of IKE, refer to clause 6.2.2. For more information on the IPCablecom use of Kerberos with symmetric keys, refer to clauses 6.4.3 and 6.5.

The key management protocol must be running asynchronous to billing event generation, and will guarantee that there is always a valid, non-expired MGC-RKS Secret.

An RKS maintains a mapping between an IP address and an Element ID for each host with which it has IPsec Security Associations, as specified in clause 7.3.3.1. This includes the MGC.

When an event message arrives at the RKS, the RKS must retrieve a source IP address based on the Element ID, based on the mapping established during key management. The RKS must ensure that this address is the same as the source IP address in the IP packet header.

7.3.4 Billing System Summary Security Profile Matrix

Table 24: Security Profile Matrix - RADIUS

	RADIUS Accounting (CMS - RADIUS Server/RKS)	**RADIUS Accounting (CMTS - RADIUS Server/RKS)**	**RADIUS Accounting (MGC - RADIUS Server/RKS)**
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 ESP with encryption and message integrity enabled. key management using IKE or Kerberos	IPsec ESP with encryption and message integrity enabled. key management using IKE or Kerberos	IPsec ESP with encryption and message integrity enabled. key management using IKE or Kerberos

7.4 Call Signalling

7.4.1 Network Call Signalling (NCS)

7.4.1.1 Reference Architecture

Figure 12 shows the network components and the various interfaces to be discussed in this clause.

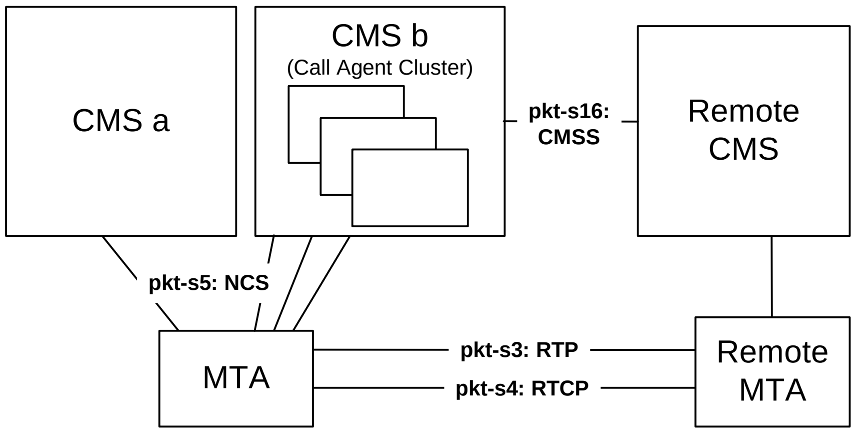


Figure 12: NCS Reference Architecture

Figure 12 shows a CMS containing a cluster of Call Agents, which are identifiable by one CMS FQDN. 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-s3 and pkt-s4 (RTP bearer channel and RTCP) are described in clause 6.6 of the present document. The protocol interface pkt-s16 (CMS to CMS) is SIP with IPCablecom extensions, as specified in [32].

When a call is made between two endpoints in different zones, the call signalling has to traverse the path between two different CMSs. The signalling protocol between CMSs is SIP with IPCablecom specific extensions. See [32] for more details. Initially, the initiating CMS may not have a direct signalling path to a terminating CMS. The call routing table of the initiating CMS may point it to an intermediate SIP proxy. That SIP proxy, in turn, may point to another SIP proxy. In general, we make no assumptions about the number of SIP proxies in the signalling path between the CMSs.

Once the two CMSs have discovered each other's location, they have the option to continue SIP signalling directly between each other. The SIP proxies that route traffic between Domains are called Exterior Border Proxies (EBPs). EBPs enforce access control on all signalling messages routed between domains. They also provide application level security on sensitive information contained within SIP messages. While not depicted in figure 12, CMSS may also be used between a CMS and an MGC.

As SIP proxies and CMSs may be in different IPCablecom domains (and consequently different trust domains), there must be a signalling path and trust relationship between two domains, before any direct SIP signalling can take place. IPCablecom Server certificates are used for TLS mutual authentication in CMSS and provide the trust infrastructure for SIP signalling. A CMS or EBP, may be configured to only trust specific Service Provider CA certificates and/or FQDNs (i.e. access list) of external CMSs and EBPs. Generally, trust between different Service Provider domains is be provided by the EBPs.

7.4.1.2 Security Services

The same set of requirements applies to both CMS-MTA and CMS-CMS signalling interfaces.

Authentication: signalling messages should be authenticated, in order to prevent a third party masquerading as either an authorized MTA, CMS, MGC, or SIP Proxy.

Confidentiality: NCS messages carry dialled numbers and other customer information, which must not be disclosed to a third party. Thus confidentiality of signalling messages should be required. The signalling messages carry media stream keying material that must be kept private on each signalling hop, and should also be kept private end-to-end between the initiating and target CMSs, to avoid exposure at SIP signalling proxies. There is no standard, well-supported mechanism to support end-to-end privacy of keying material, however, so only hop-by-hop confidentiality is supported in IPCablecom.

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

7.4.1.3 Cryptographic Mechanisms

IPsec ESP must be used to secure the NCS signalling between the CMS and MTA. IPsec keys must be derived using the mechanism described in clause 6.5.3.1.

TLS must be used to secure the SIP signalling (CMSS) between CMSs and between CMSs and SIP proxies (EBPs).

The first SIP signalling roundtrip between the initiating and target CMSs may transit through any number of intermediate SIP signalling proxies. Since TLS is applied separately on each signalling hop, the contents of the SIP signalling message is decrypted and re-encrypted at each SIP signalling proxy. The full contents of the SIP signalling message, including media stream keying material, are available in the clear at each intermediate SIP signalling proxy.

7.4.1.3.1 MTA-CMS Interface

Each signalling message coming from the MTA and containing the MTA domain name (included in the NCS endpoint ID field) must be authenticated by the CMS. This domain name is an application-level NCS 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 to FQDN map for each MTA IP address that has a current SA. This map must be built during the key management process described in the following clause and does not need to reside in permanent storage.

7.4.1.3.2 CMS-CMS, CMS-MGC, CMS-SIP Proxy and SIP Proxy - SIP Proxy Interfaces

When a CMS or MGC or a SIP Proxy receives a SIP signalling message, it should map the source IP address to the identity (FQDN) of the CMS or SIP Proxy and to the local policy associated with that FQDN. This lookup would utilize an IP address to FQDN map for all MGCs and SIP Proxies that have current TLS sessions with this host. This map is built during key management described in the following clause and does not need to reside in permanent storage.

7.4.1.4 Key Management

7.4.1.4.1 MTA-CMS Key Management

The MTA must use Kerberos with PKINIT to obtain a CMS service ticket (see clause 6.4.3). The MTA should first obtain a TGT (Ticket Granting Ticket) via the AS Request/AS Reply exchange with the KDC (authenticated with PKINIT). In the case that the MTA obtained a TGT, it performs a TGS Request/TGS Reply exchange to obtain the CMS service ticket (see clause 6.4.4).

After the MTA has obtained a CMS ticket, it must execute a Kerberized key management protocol (that utilizes the CMS ticket) with the CMS to create SAs for the pkt-s10 interface. This Kerberized key management protocol is specified in clause 6.5. Clause 6.5 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 NCS message. Rather, the MTA will obtain the Kerberos ticket from the KDC when started and will refresh it based on the timeout parameter. Similarly, the MTA will obtain the sub-key (and thus IPsec ESP keys) based on the IPsec timeout parameters. In addition, when the IPsec ESP keys are timed out and the MTA needs to transmit data to the CMS, it will perform key management with the CMS and obtain the new keys. It is also possible for the IPsec SAs to expire at the CMS while it has data to send to the MTA. In this case, clause 6.5.3.5.3 describes the technique for the CMS to initiate key management and establish new Security Associations.

7.4.1.4.1.1 Call Agent Clustering

At the time that the CMS receives a Kerberos ticket for establishing an IPsec SA, it must extract the MTA FQDN from the MTA principal name in the ticket and map it to the IP address. This map is later used to authenticate the MTA endpoint ID in the NCS signalling messages.

In the case a CMS, or an application server, 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 KDC and multiple AP Request/Reply sequence for each Call Agent in the cluster. The Kerberos messages are specified in clause 6.4.4.

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 a IPsec SA with that MTA (see clause 6.5.2.1).

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 SA with this optimized approach is a single roundtrip time. This is illustrated in figure 13.

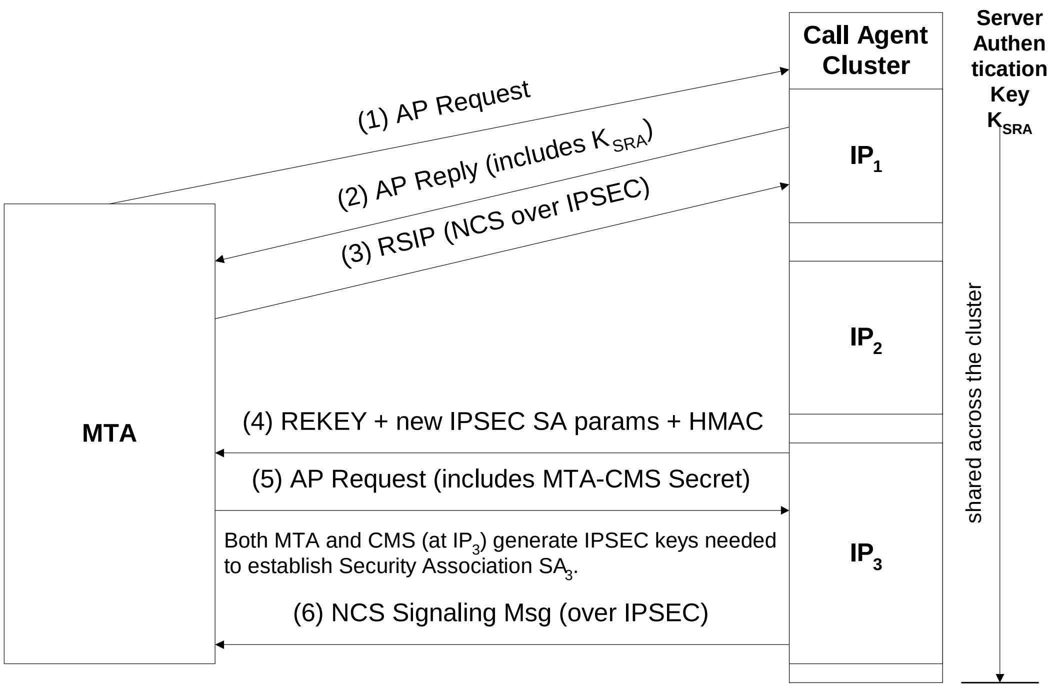

Call Agent Cluster
Server Authen tication Key K_{SRA}
(1) AP Request
(2) AP Reply (includes K_{SRA})
(3) RSIP (NCS over IPSEC)
IP_1
IP_2
IP_3
MTA
shared across the cluster
(4) REKEY + new IPSEC SA params + HMAC
(5) AP Request (includes MTA-CMS Secret)
Both MTA and CMS (at IP_3) generate IPSEC keys needed to establish Security Association SA_3.
(6) NCS Signaling Msg (over IPSEC)

Figure 13: Key Management for NCS Clusters

In figure 13, 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:

(4) The CMS at IP_3 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, and 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}.

(5) 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 6.5.3.

7.4.1.4.1.2 MTA Controlled by Multiple CMSs

In the case a single MTA is controlled by multiple CMSs and each CMS is associated with a different Kerberos realm, the MTA will need to execute multiple PKINIT Request/Reply exchanges with the KDC, one for each realm, optionally followed by a TGS Request/Reply exchanges. Then, an MTA would execute multiple AP Request/Reply exchanges in order to create the Security Associations with the individual CMSs.

7.4.1.4.1.3 Transferring from one CMS to another via NCS signalling

When control of an MTA endpoint is transferred from one CMS to another via NCS signalling, the following steps are taken:

1) The new CMS might not have been included in the CMS table. In that case, the corresponding table entry must be locally created. Refer to clause 7.1.1.2.5 for instructions on how to create the new CMS table entry.

2) If the MTA does not already have IPsec SAs established with this CMS (e.g. via an earlier Wake Up), it must attempt to establish them at this time.

3) If the MTA now possesses valid IPsec Security Associations with the new CMS, the NCS signalling software is notified and the Security Association can be utilized. Further signalling traffic for this affected endpoint related to the prior CMS Security Association must not be sent.

7.4.1.4.2 CMS-CMS, CMS-MGC, CMS-SIP Proxy, SIP Proxy-SIP Proxy Key Management

When a CMS, MGC, or a SIP Proxy has data to send to another CMS, MGC, or SIP Proxy and does not already have a TLS Session with that host, it must first establish a TLS session with the other CMS, MGC, or SIP Proxy (see clause 6.9).

A CMS or a SIP Proxy should create TLS sessions ahead of time (before they are needed) whenever possible and maintain persistent connections.

7.4.1.4.2.1 Example of Inter-Domain Call Setup with TLS Sessions

The following example diagram, depicts a typical SIP signalling flow for an inter-domain call setup in which EBPs are used (refer to [32] for further details on CMSS call flows). It illustrates several points in the end-end call setup where different TLS sessions and TCP connections may be required and also emphasizes the importance of connection persistence and re-use to minimize TCP connection and TLS session establishment during the call setup (i.e. in order to minimize performance impacts and call setup delays). It should be noted that in a peering relationship between two SIP User Agents (i.e. CMSs and/or EBPs) often results in two TCP connections, one for SIP transactions initiated in each direction. This is due to the fact most TCP connections are initiated using ephemeral source ports and SIP transactions are initiated by sending SIP requests to a User Agent's well-known SIP port. As for securing each TCP connection with TLS, TLS clients typically cache TLS sessions based on specific remote IP address and port pairs, therefore it is unlikely TLS session caching using a common TLS master key can be used for both of the TLS sessions.

The inter-domain signalling flow begins with CMS "A" sending an INVITE to EBP "A" (CMS "A" is initiating a SIP INVITE transaction and will signal the well-known SIP port on EBP "A"). A TLS session is required and may need to be established for the TCP connection if one does not already exist (which can be re-used) for this new transaction. Similarly, a TLS session is required for each hop in this INVITE transaction, and may require a TLS session to be established between EBP "A" and EBP "B", and also between EBP "B" and CMS "B".

The 183 (Session Progress) response from CMS "B" is routed back through the EBPs to CMS "A" using the previously established TLS sessions. Once CMS "A" receives this 183 response, it sends a PRACK (Provisional ACK) directly to CMS "B". This PRACK is a SIP request which initiates a new transaction, and requires a TLS session be established with CMS "B"'s well-known SIP port. CMS "B" sends a 200 OK response back to CMS "A" (using the same TLS session and TCP connection) as a the final response to this PRACK transaction. Upon receiving a response to its initial INVITE, CMS "A" will also send an UPDATE request to CMS "B" over the previously established TLS session, to indicate resource reservation has been completed. CMS "B" will respond with a 200 OK, completing this UPDATE transaction.

Upon receiving the UPDATE, CMS "B" reserves any necessary resource and sends back a 180 (Ringing) provisional response to CMS "A" over the previously established TLS session. This provisional response will also initiate PRACK / 200 OK transaction between the two CMSs, over the same TLS session.

Once the terminating end answers the call, CMS "B" sends the 200 OK final response to the INVITE. However, this response is sent back via the EBPs using the same TLS sessions used for the INVITE. CMS "A" will acknowledge receipt of this 200 OK response by sending an ACK (a SIP request) directly to CMS "B". This ACK is sent using the previously established TLS session used for the first PRACK.

Once the call is established, the example illustrates the case where the terminating end goes on hook. CMS "B" initiates a BYE / 200 OK transaction by sending a BYE (SIP request) to CMS "A". As this BYE request is sent to the well-known SIP signalling port of CMS "A", it is very likely CMS "B" will need to use a different TCP connection and TLS session than the ones used for sending SIP requests from CMS "A" to CMS "B" (assuming CMS "A" is using an ephemeral port for its TCP connection to CMS "B").

As can be seen from this example, two TCP connections with two distinct TLS sessions may be required between two CMSs. It is important to support persistent and re-usable connections and TLS session caching in order to minimize impacts on CMS performance and call latency.

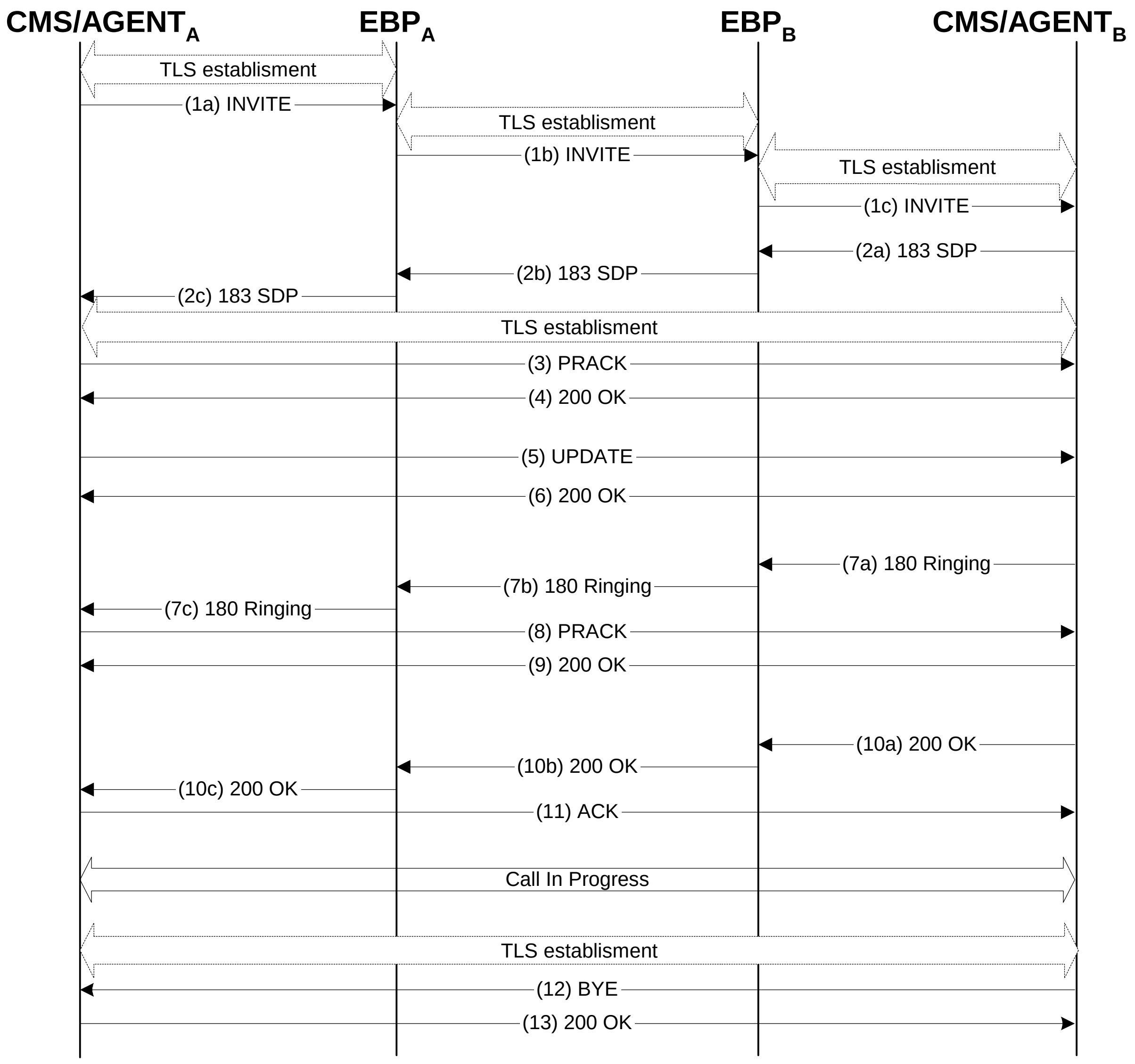


Figure 14: CMS - CMS Signalling Flow with Security

7.4.2 Call Signalling Security Profile Matrix

Table 25: Security Profile Matrix - Network Call Signalling

	MTA-CMS	**CMS-CMS**	**CMS-SIP Proxy / SIP Proxy-SIP Proxy**
authentication	optional	yes	yes
access control	optional	yes	yes
integrity	optional	yes	yes
confidentiality	optional	yes	yes
non-repudiation	no	no	no
security mechanisms	IPsec ESP with encryption and message integrity enabled. Authentication via Kerberos with PKINIT Kerberized key management defined by IPCablecom Security may be disabled through the provisioning process	TLS with encryption and message integrity. Authentication via X.509 certificates (or symmetric keys when TLS session caching is used)	TLS with encryption and message integrity. Authentication via X.509 certificates (or symmetric keys when TLS session caching is used)

7.5 PSTN Gateway Interface

7.5.1 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).

These components are described in detail in [5].

7.5.1.1 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.

7.5.1.2 Media Gateway

Media Gateways (MG) provide the bearer connectivity between the PSTN and the IPCablecom IP network.

7.5.1.3 Signalling Gateway

IPCablecom provides support for SS7 signalling gateways. The SG contains the SG to MGC interface. Refer to [5] for more detail on signalling gateways.

The SS7 Signalling Gateway performs the following security-related functions:

- Isolates the SS7 network from the IP network. Guards the SS7 network from threats such as Information Leakage, integrity violation, denial-of-service, and illegitimate use.
- Provides mechanism for certain trusted entities ("TCAP Users") within the IPCablecom network, such as Call Agents, to query external PSTN databases via TCAP messages sent over the SS7 network.

7.5.2 Security Services

7.5.2.1 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 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.

7.5.3 Cryptographic Mechanisms

7.5.3.1 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 6.1.2 for details of how IPsec ESP is used within IPCablecom and for the list of available ciphersuites.

7.5.4 Key Management

7.5.4.1 MGC - MG Interface

Key management for the MGC-MG interface is either IKE or Kerberos. Implementations must support IKE with pre-shared keys. Implementations may support IKE with X.509 certificates and they may support Kerberos using symmetric keys. For more information on the IPCablecom use of IKE, refer to clause 6.2.2. For more information on the IPCablecom use of Kerberos with symmetric keys, refer to clauses 6.4.3 and 6.5.

The key management protocol ensures that there is always a valid, non-expired MGC - MG secret.

7.5.5 MGC-MG Summary Security Profile Matrix

Table 26: Security Profile Matrix - TGCP

	TGCP (MG - MGC)
authentication	yes
access control	yes
integrity	yes
confidentiality	yes
non-repudiation	no
security mechanisms	IPsec IKE or Kerberos

7.6 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.

7.6.1 Security Services

7.6.1.1 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 and/or MGs should 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 8 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 2 or 4 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 7.6.2.3.1).

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.

7.6.1.2 RTCP

Authentication: see the above clause.

Encryption: within IPCablecom, RTCP messages are not permitted to contain the identity of the RTCP termination endpoint. Snooping on RTCP messages, therefore, does not reveal any subscriber-specific information but may reveal network usage and reliability statistics. RTCP encryption is optional.

Message Integrity: RTCP signalling messages (e.g. BYE) can be manipulated to cause denial-of-service attacks and alteration of reception statistics. To prevent these attacks, message integrity should be used for RTCP.

7.6.2 Cryptographic Mechanisms

MTAs and MGs must have an ability to negotiate a particular encryption and authentication algorithm. If media security parameters are negotiated and RTP encryption is on (Transform ID is not RTP_ENCR_NULL), each media RTP packet must be encrypted for privacy. If RTP encryption is on, encryption must be applied to the RTP payload and must not be applied to the RTP header. Security must not be applied to RTP packets if the negotiated RTP ciphersuite is AUTH_NULL and RTP_ENCR_NULL. 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 7.6.2.3.1.

7.6.2.1 RTP Messages

Figure 15 shows the format of an encoded RTP packet. IPCablecom must adhere to the RTP packet format as defined by RFC 1889 [10] and RFC 1890 [42] after being authenticated and decrypted (where the MAC bytes, if included, are stripped off as part of the authentication).

The packet's header consists of 12 or more octets, as described in [10]. 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 [10]):

0					1	2 3
0 1	2	3	4 5 6 7	8	9 0 1 2 3 4 5	6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
V=2	P	X	CC	M	PT	Sequence Number
Timestamp						
Synchronization Source (SSRC) Identifier						
Contributing Source (CSRC) Identifier						

Figure 15: RTP Packet 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.

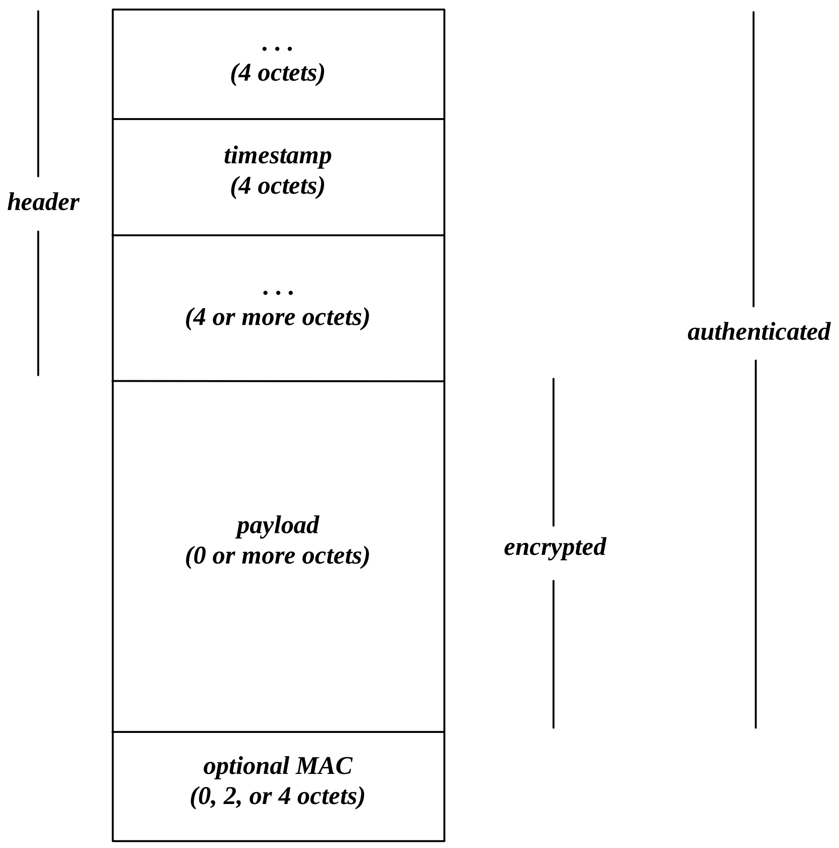


Figure 16: Format of Encoded RTP Packet

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. Mf 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 [2].

  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/sec. 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 within the duration of one codec frame. It is assumed that an event packet can have a payload as large as that of a voice packet, but no longer. In the case of a block cipher, the cryptographic keys do not change after midstream codec changes. When a codec change does not require a corresponding key change, the value of N_e must be calculated as follows:

$$N_e = \text{MAX}\ \{ \text{NcK} \}\ \text{for } K = 1, \ldots N$$

where N1, N2, … NK are the different frame sizes for codecs that are supported by a particular endpoint.

Otherwise, $N_e = N_c$, where N_c is the frame size for the current codec.

- 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.

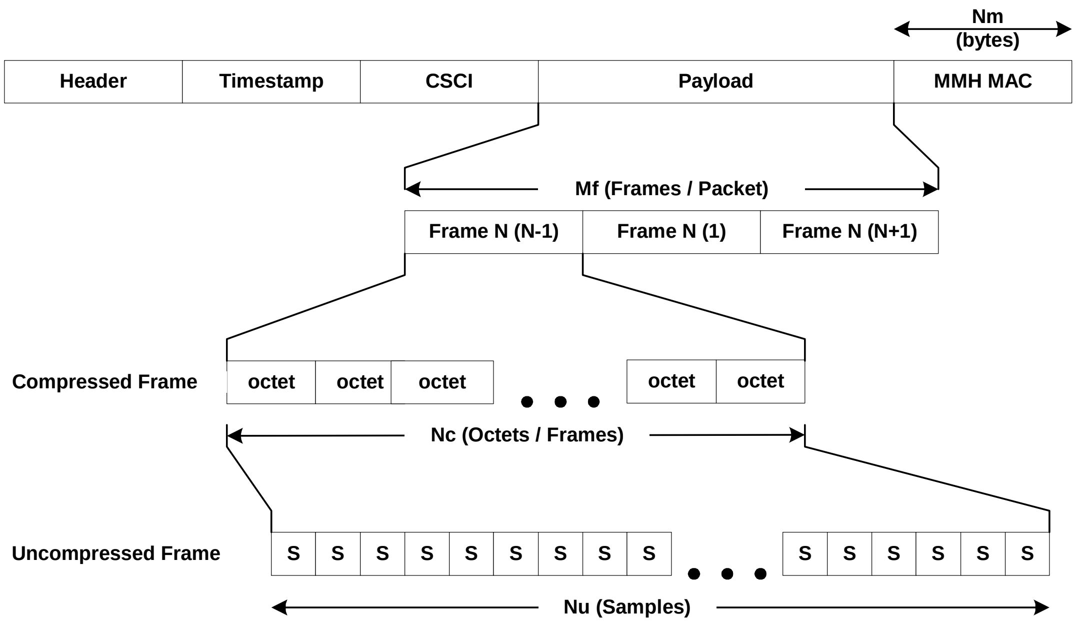


Figure 17: RTP Packet Profile Characteristics

7.6.2.1.1 RTP Timestamp

According to RFC1889 [10], the timestamp field is a 32-bit value initially chosen at random. 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: $((N_f*N_u) + (\text{RTP Initial Timestamp}))$ modulo 2^{32}, where N_f is the frame number of the first frame included in the packet.

7.6.2.1.2 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 clause 7.6.2.3.3.

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

7.6.2.1.2.1 Encryption and MMH MAC Option

7.6.2.1.2.1.1 Deriving an MMH MAC Key

The MMH MAC Key size must be determined before generating the MMH MAC Key. The following algorithm specifies how to derive the MMH MAC Key when being used with block ciphers.

$$\text{MMH MAC key size} = (M_f * N_e) + N_h + N_m - 2 + P$$

where M_f is the maximum number of frames per packet; N_e is maximum number of octets in one frame of compressed audio; N_h is the maximum number of octets in the RTP header, as defined in clause 7.6.2.1; and N_m is the number of octets in the MAC. Therefore, $(M_f * N_e) + N_h$ represents the maximum size of an RTP packet, and $N_m - 2$ represents the additional two octets that are added to the key size when a four octet MMH MAC is used. (The key size is the same as the maximum RTP packet size when a two octet MMH MAC is used.) P is 0 or 1, as needed to make the MMH MAC key size an even number so that it is a multiple of the word size (2 bytes) used in the MMH MAC algorithm.

The number of octets in the RTP header ranges from 12 to 72, inclusive, depending on the number of CSRC identifiers that are included [10]. 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.

Since the key derivation procedure generates the MMH MAC key last (see clause 7.6.2.3.3.1), it is not necessary to generate a complete MMH MAC key at the start of the RTP session. Implementations may generate less than the full MMH MAC key and generate the rest later, as needed. For example, instead of using a value of N_e that reflects all possible codecs supported by an endpoint, an implementation might initially derive an MMH key of size $(M_f * N_c) + N_h + N_m - 2 + P$, where N_c is the frame size for the currently selected codec. Later, after a codec change that results in a larger value of N_c, additional bytes for the MMH key may be generated.

7.6.2.1.2.1.2 RTP Timestamp Wrap-around

Let us say that the initial RTP timestamp value is T_0. A timestamp wrap-around occurs when:

- an RTP packet with sequence number i has a timestamp value $2^{32} - \xi_1$ for $0 < \xi_1 <= \Delta T_{MAX}$, where ΔT_{MAX} is the maximum difference between two consecutive RTP timestamps.
- an RTP packet with a sequence number i+1 has a timestamp value ξ_2 for $0 <= \xi_2 < \Delta T_{MAX}$.

The wrap-around point is between the RTP packets i and i+1.

Each endpoint must keep a count N_{WRAP} of RTP timestamp wrap-arounds, with a range from 0 to 2^{16}-1 and initialized to zero at the start of the connection N_{WRAP} must be incremented by the sender right after the wrap-around point. N_{WRAP} must also be incremented by the receiver before it decrypts any RTP packets after the wrap-around point.

7.6.2.1.2.2 Block Cipher Encryption of RTP Packets

The AES Block Cipher must be supported for encryption of RTP packets. The following clauses specify how to support any Block Cipher, including AES.

7.6.2.1.2.2.1 Block Termination

If an implementation supports block ciphers, residual block termination (RBT) must be used to terminate streams that end with less than a full block of data to encrypt (see clause 9.3).

7.6.2.1.2.2.2 Initialization Vector

An Initialization Vector (IV) is required when using a block cipher in CBC mode to encrypt RTP packet payloads. The size of an IV is the same as the block size for the particular block cipher. For example, the IV size for DESX and 3-DES is 64 bits, while for AES-CBC it is 128 bits. In order to calculate the IV each endpoint must keep track of N_{WRAP} - the count of timestamp wrap-arounds during this RTP session, see clause 7.6.2.1.2.1.2. The IV must be calculated new for each RTP packet as specified below:

1) Take the first N bits of the header, where N = min(cipher block size, RTP header size).

2) In the result of the previous step replace the first 16 bits of the header with the 16-bit value of NWRAP, MSB first.

3) Pad the result of previous step with 0's on the right, so that the resulting bit string is equal in size to the cipher block size.

4) XOR the result of the previous step with the RTP Initialization Key (defined in clause 7.6.2.3.3.1). The size of the RTP Initialization Key is the same as the cipher block size.

5) Encrypt the result of the previous step using the same block cipher that is used to encrypt RTP packets, but in ECB mode. The result of this step is the Initialization Vector for this RTP packet.

7.6.2.1.2.2.3 MMH-MAC Pad Derivation When Using a Block Cipher

The MMH-MAC algorithm requires a one-time pad for each RTP packet. The MMH-MAC Pad must be derived by performing the MMH Function on the Block Cipher's IV. For a 2-byte MMH-MAC, use the MMH Function described in clause 9.7.1.1; for a 4-byte MMH-MAC, use the MMH Function described in clause 9.7.1.2.

The IV is calculated according to clause 7.6.2.1.2.2.2 for block ciphers that require an IV. Even if the block cipher does not require an IV, one must be derived according to clause 7.6.2.1.2.2.2 and used as the basis of the MMH-MAC Pad derivation.

A key is also required by the MMH digest function in order to calculate the pad. The MMH MAC key derived in clause 7.6.2.3.3.1 must be truncated according to clause 9.7.2.2 and must then be used as the key to the MMH digest. Accordingly, the MMH MAC key is truncated to:

$$<\text{size of IV}> + N_m - 2$$

Where <size of IV> is 16 bytes for AES, N_m is the size of the MMH MAC in bytes, as defined in clause 7.6.2.1, and N_m - 2 represents the additional two octets that are added to the key size when a four octet MMH MAC is used). (The truncated key size is the same as the IV size when a two octet MMH MAC is used.)

7.6.2.1.3 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 7.6.2.3.3.

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

7.6.2.1.3.1 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 items (1) and (2) 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;

    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.

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

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

7.6.2.1.3.2 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.

7.6.2.2 RTCP Messages

7.6.2.2.1 RTCP Format

RFC 1889 [10] defines the packet format of RTCP messages.

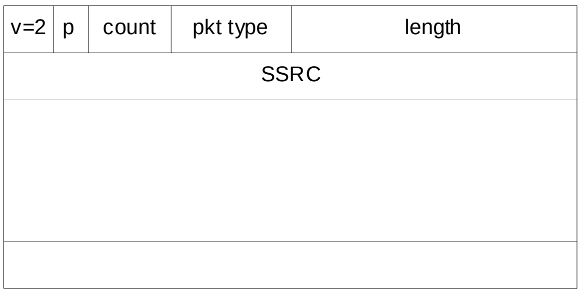


Figure 18: RTCP Packet Format

The RTCP packet type could be SR (sender reports), RR (receiver reports), SDES (source description), BYE (leaving conference), and APP (application specific function). The length varies depending on the message type, but generally around 40 bytes.

7.6.2.2.2 RTCP Encryption

RTCP messages must always be encrypted in their entirety when the negotiated encryption algorithm is a block cipher in CBC mode. RTCP messages must not be encrypted when the negotiated encryption algorithm is RTCP_ENCR_NULL. However, the encoded RTCP messages must still be formatted according to clause 7.6.2.2.2 when RTCP_ENCR_NULL is selected in conjunction with a non-NULL authentication algorithm (e.g. HMAC-SHA1-96 or HMAC-MD5-96). Security must not be applied to RTCP packets if the negotiated RTCP ciphersuite is RTCP_AUTH_NULL and RTCP_ENCR_NULL. After the message is encrypted, an additional header and MAC (Message Authentication Code) are added. The result packet has the format in figure 19.

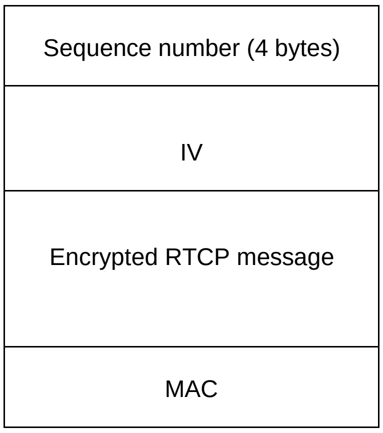


Figure 19: RTCP Encrypted Packet Format

The first 4 bytes must be the sequence number, MSB first. The initial sequence number for each direction of traffic must be 0. Afterwards, the sequence number for each direction must be incremented by 1. Generally, one RTCP message is sent every 5 seconds for each channel. Thus 32 bits for the sequence number field would be big enough for any connections without wrapping around.

The IV (Initialization Vector) must immediately follow the sequence number. The IV must be randomly generated by the sender for each RTCP message and the IV size must be the same as the block size for the selected block cipher. The Initialization Vector (IV) must not be included when RTCP_ENCR_NULL is used.

The original cleartext RTCP message encrypted in its entirety must immediately follow the IV. The MAC (Message Authentication Code) computed over the concatenation of the sequence number, IV and the encrypted message must follow the encrypted RTCP message. The size of the MAC is algorithm-dependent.

7.6.2.2.3 Sequence Numbers

The receiver of RTCP messages should keep a sliding window of the RTCP sequence numbers. The size of the sliding window W_{RTCP} depends on the reliability of the UDP transport and is locally configured at each endpoint. W_{RTCP} should be 32 or 64. The sliding window is most efficiently implemented with a bit mask and bit shift operations.

When the receiver is first ready to receive RTCP packets, the first sequence number in this window must be 0 and the last must be W_{RTCP} - 1. All sequence numbers within this window must be accepted the first time but must be rejected when they are repeated. All sequence numbers that are smaller than the "left" edge of the window must be rejected.

When an authenticated RTCP packet with a sequence number that is larger than the "right" edge of the window is received, that sequence number is accepted and the "right" edge of the window is replaced with this sequence number. The "left" edge of the window is updated in order to maintain the same window size.

When for a window ($S_{RIGHT} - W_{RTCP} + 1, S_{RIGHT}$), sequence number S_{NEW} is received and $S_{NEW} > S_{RIGHT}$, then the new window becomes:

$$(S_{NEW} - W_{RTCP} + 1, S_{NEW})$$

7.6.2.2.4 Block Termination

Residual block termination (RBT) must be used to terminate RTCP messages that end with less than a full block of data to encrypt (see clause 9.3).

7.6.2.2.5 RTCP Message Encoding

Each RTCP message must be encoded using the following procedure:

1) A random IV is generated.

2) The entire RTCP message is encrypted with the selected block cipher and the just generated IV.

3) The current sequence number, IV and the encrypted RTCP message are concatenated in that order.

4) The MAC is computed (using the selected MAC algorithm) over the result in c) and appended to the message.

7.6.2.2.6 RTCP Message Decoding

Each RTCP message must be decoded using the following procedure:

1) Regenerate the MAC code and compare to the received value. If the two do not match, the message is dropped.

2) The sequence number is verified based on the sliding window approach specified in clause 7.6.2.2.3. If the sequence number is rejected, the message is dropped. The sliding window is also updated as specified in clause 7.6.2.2.3.

3) The RTCP message is decrypted with the shared encryption key and with the IV that is specified in the message header.

7.6.2.3 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, 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.

During call setup MTA_0 (the initiating MTA) and MTA_1 (the terminating MTA) exchange randomly generated keying material, carried inside the call signalling messages. Call signalling messages are themselves protected by IPsec ESP or TLS at each hop. This keying material is then used to generate the AES-CBC keys used to protect both RTP and RTCP messages between the two MTAs.

MTA_0 generates two randomly generated values: End-End $Secret_0$ (46-bytes) and Pad_1 (46-bytes).

MTA_1 generates two randomly generated values: End-End $Secret_1$ (46-bytes) and Pad_0 (46-bytes).

MTA_0 uses End-End $Secret_1$ and Pad_1 to derive encryption and authentication keys to be applied to its outbound traffic and used by MTA_1 to decrypt and authenticate it.

MTA_1 uses End-End $Secret_0$ and Pad_0 to derive encryption and authentication keys to be applied to its outbound traffic, and used by MTA_0 to decrypt and authenticate it. As a result, both MTA_0 and MTA_1 contribute randomly generated bytes to all of the keying material for both RTP and RTCP traffic.

The distribution of the end-to-end keying material is specific to the call signalling from [2] and is described in the following clauses.

7.6.2.3.1 Key Management over NCS

Figure 20 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.

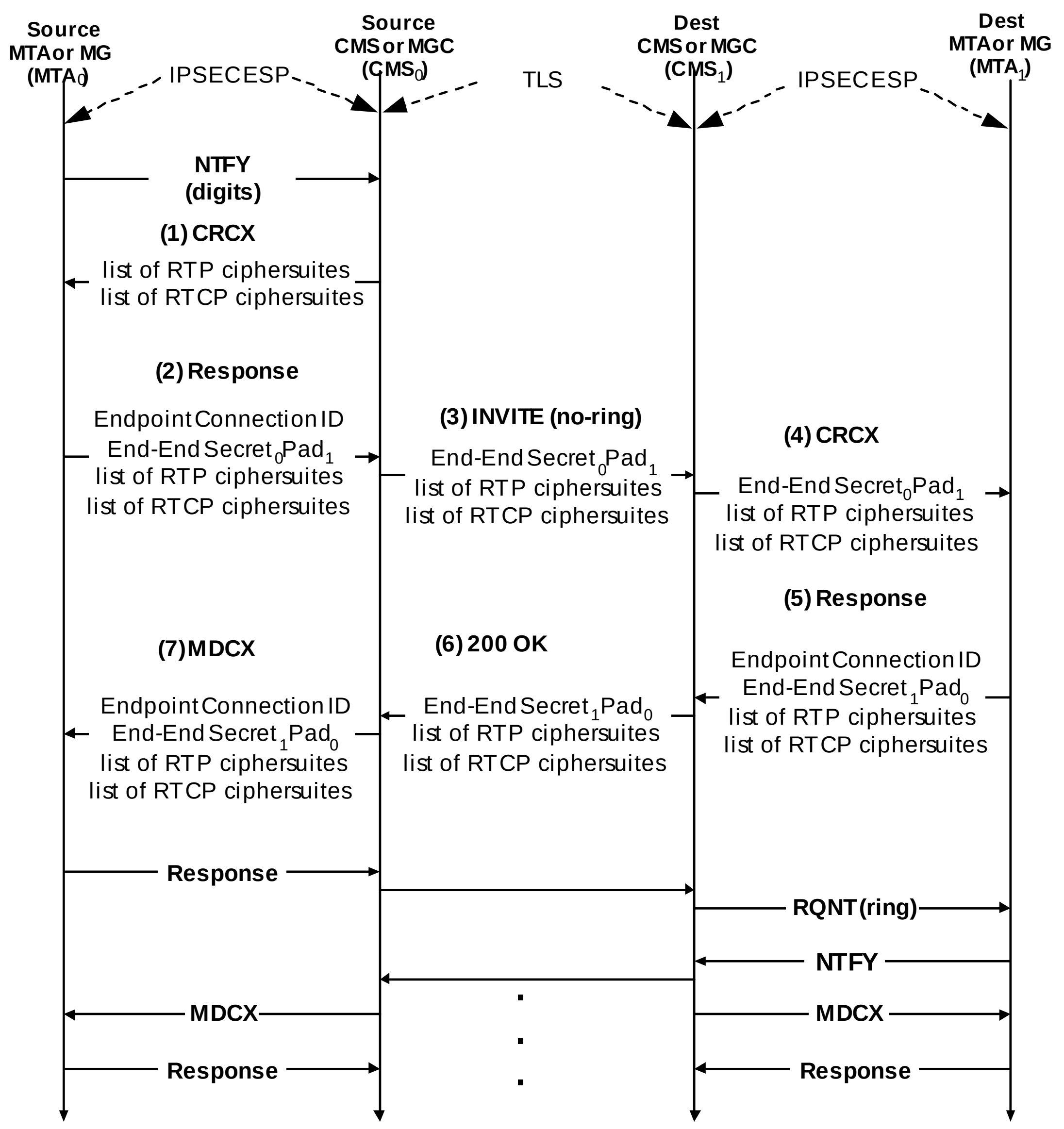


Figure 20: End-End Secret Distribution over NCS

Figure 20 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. It is also assumed that CMS-CMS signalling is secure.

This allows the End-End Secrets to be distributed securely, with privacy, integrity and anti-replay mechanisms already in place. The CMSs have access to this keying material but are trusted by the MTAs.

7.6.2.3.1.1 NULL Ciphersuite Combinations and Ordering

RTP_ENCR_NULL must only be used in conjunction with AUTH_NULL. RTP packets, with authentication but no encryption, are not allowed.

RTCP_AUTH_NULL must only be used in conjunction with RTCP_ENCR_NULL. RTCP messages with encryption and without authentication are not allowed.

Both RTP and RTCP security must be enabled or disabled together. The following five combinations must not be generated:

- RTP NULL encryption and RTP non-NULL authentication
- RTCP non-NULL encryption & RTCP NULL authentication

- RTP non-NULL encryption and RTCP NULL authentication
- RTP NULL encryption and RTCP non-NULL authentication
- RTP NULL encryption and RTCP non-NULL encryption

If the MTA receives LocalConnectionOptions parameter that meet the above combinations, the MTA must return the error code 524 (Internal inconsistency in LocalConnectionOptions). Otherwise, if the MTA receives RemoteConnectionDescriptor parameter that meet the above combinations, then the MTA must return the error code 505 (Unsupported RemoteConnectionDescriptor).

For both RTP and RTCP ciphersuite lists exchanged during ciphersuite negotiation, the combination of NULL encryption and NULL authentication algorithms must always be included last. For example, the list of RTP ciphersuites "60/50;62/51;64/51" is not allowed, while the list of RTP ciphersuites "62/51;64/51;60/50", or "60/50" is allowed. If the list of ciphersuites in LocalConnectionOptions includes the NULL authentication and NULL encryption combination (60/50 for RTP, and 80/70 for RTCP), but this combination is not last, the MTA must return error code 524 (Internal inconsistency in LocalConnectionOptions). Otherwise, if this combination is not last in a RemoteConnectionDescriptor, error code 505 (Unsupported RemoteConnectionDescriptor) must be returned.

7.6.2.3.1.2 Ciphersuite Negotiation For MTAs

The present document only defines security for RTP/RTCP media streams, therefore ciphersuite negotiation applies only to RTP/RTCP media streams. Use of security for any other type of media streams is not specified.

An MTA must perform RTP and RTCP ciphersuite negotiation when processing any of the following:

- a CreateConnection command
- a ModifyConnection command with a RemoteConnectionDescriptor parameter
- a ModifyConnection command where the LocalConnectionOptions parameter includes ciphersuite fields

An MTA must not perform ciphersuite negotiation in any other case. The steps involved in ciphersuite negotiation are the following:

1) An approved list of ciphersuites is formed by taking the intersection of the internal list of ciphersuites and ciphersuites allowed by the LocalConnectionOptions parameter, subject to the constraints specified in clause 7.6.2.3.1.1. The internal list of ciphersuites contains the ciphersuites that the MTA supports and which the present document requires. If the LocalConnectionOptions parameter was not included, or if the ciphersuite fields were not provided in the LocalConnectionOptions parameter, the approved list of ciphersuites contains the previously agreed upon approved list, or if no such list exists, the internal list of ciphersuites.

2) If the approved list of ciphersuites is empty, an error response must be generated, error code 532 (Unsupported value(s) in LocalConnectionOptions).

3) Otherwise, a negotiated list of ciphersuites is formed by taking the intersection of the approved list of ciphersuites and ciphersuites allowed by the RemoteConnectionDescriptor parameter (if present), subject to the constraints specified in clause 7.6.2.3.1.1. If a RemoteConnectionDescriptor was not provided, the negotiated list of ciphersuites thus contains the approved list of ciphersuites. If a RemoteConnectionDescriptor parameter is provided without fields containing the RTP and RTCP ciphersuite lists, then the RTP AUTH_NULL/RTP_ENCR_NULL and RTCP_AUTH_NULL/RTCP_ENCR_NULL ciphersuites are assumed for the remote endpoints, and the regular ciphersuite negotiation process continues (i.e. the negotiated list of ciphersuites is formed by taking the intersection of the approved list of ciphersuites and the RTP AUTH_NULL/RTP_ENCR_NULL and RTCP_AUTH_NULL/RTCP_ENCR_NULL ciphersuites).

4) If the negotiated list of ciphersuites is empty, a ciphersuite negotiation failure has occurred and an error response must be generated. If a RemoteConnectionDescriptor parameter was provided, two different error codes can be returned:

   a) If the endpoint does not support any of the ciphersuites allowed by the RemoteConnectionDescriptor, error code 505 (Unsupported RemotedConnectionDescriptor) must be used.

   b) If the endpoint does support at least one of the ciphersuites, but the negotiated list of ciphersuites ended up being empty, error code 506 (Unable to satisfy both LocalConnectionOptions and RemoteConnectionDescriptor) must be used.

5) Otherwise, ciphersuite negotiation has succeeded, and the negotiated list of ciphersuites is returned in the LocalConnectionDescriptor parameter. Note that both LocalConnectionOptions and the RemoteConnectionDescriptor parameters can contain a list of ciphersuites that must be ordered by preference provided by the CMS in the RemoteConnectionDescriptor parameter. When both are supplied, the MTA should adhere to the preferences provided by the CMS in the RemoteConnectionDescriptor parameter, and otherwise, the MTA should adhere to the preferences provided in the LocalConnectionOptions parameter. If the MTA receives a RemoteConnectionDescriptor parameter with AUTH_NULL/RTP_ENCR_NULL for RTP or RTCP_AUTH_NULL/RTCP_ENCR_NULL for RTCP that is not last in the list, it must return the error code 505 (Unsupported RemoteConnectionDescriptor).

The following requirements apply during ciphersuite negotiation:

- A CMS must be capable of sending the allowable lists of ciphersuites for RTP and/or RTCP in the LocalConnectionOptions parameter of a CreateConnection command (CRCX) or a ModifyConnection command (MDCX) in the order of preference specified by the operator subject to the constraints specified in clause 7.6.2.3.1.1.

- Whenever possible, a MTA should select the first supported ciphersuite for RTP and the first supported ciphersuite for RTCP in the RemoteConnectionDescriptor parameter. This allows the MTA to immediately start sending RTP and RTCP packets to the other MTA. An MTA may instead select alternate ciphersuites specified by the other MTA.

- When returning a LocalConnectionDescriptor and the negotiated list of RTP and RTCP ciphersuites is NULL, an MTA must not include an End-End Secret or Pad.

- When returning a LocalConnectionDescriptor and the negotiated list of RTP and RTCP ciphersuites contains at least one non-NULL selection each, an MTA must include an End-End Secret (for incoming RTP and RTCP packets) and may include a Pad value (for outgoing RTP and RTCP packets). The following rules apply:

  1) The MTA must generate a new End-End Secret when responding to a CreateConnection command.

  2) The MTA must generate a new End-End Secret when responding to a ModifyConnection command if the remote connection address (e.g. IP Address) or the remote transport address (e.g. port) are not identical to what was previously assigned.

  3) The MTA must use the existing End-End Secret when responding to a ModifyConnection command where there was no previous RemoteConnectionDescriptor provided.

  4) The MTA must generate a new Pad when responding to a CreateConnection command without a RemoteConnectionDescriptor.

  5) The MTA must generate a new Pad when generating a new End-End Secret in response to a ModifyConnection command without a RemoteConnectionDescriptor.

  6) If not otherwise required, the MTA may generate a new Pad when generating a new End-End Secret.

  7) The MTA must not generate a new Pad when not generating a new End-End Secret.

- If, in response to a CreateConnection command, the list of ciphersuites selected for RTP contains at least one non-NULL encryption or authentication algorithm, before sending the response message, an MTA must:

  1) Establish inbound RTP security based on the preferred (first) RTP ciphersuite, its End-End Secret (which it generated), and a Pad value (if included in the RemoteConnectionDescriptor), as described in clause 7.6.2.3.3.1 of the present document.

  2) If a RemoteConnectionDescriptor was included and it contains media security attributes, establish outbound RTP security based on the selected RTP ciphersuite, End-End Secret (generated by the other MTA), and a Pad value (which it may have generated) as described in clause 7.6.2.3.3.1 of the present document.

3) If connection mode allows, be ready to receive RTP packets, which may arrive any time after the Response message is sent.

- If, in response to a CreateConnection command, the list of ciphersuites for RTCP contains at least one non-NULL encryption algorithm, before sending the response message, an MTA must:

  1) Establish inbound RTCP security based on the preferred (first) RTCP ciphersuite, its End-End Secret (which it generated), and a Pad value (if included in the RemoteConnectionDescriptor), as described in clause 7.6.2.3.3.1 of the present document.

  2) If a RemoteConnectionDescriptor was included and it contained media security attributes, establish outbound RTCP security based on the selected RTCP ciphersuite, End-End Secret (generated by the far-end MTA), and a Pad value (which it may have generated) as described in clause 7.6.2.3.3.1 of the present document.

  3) Be ready to receive RTCP packets, which may arrive any time after the Response message is sent.

- If, in response to a ModifyConnection command that includes a RemoteConnectionDescriptor, and negotiated lists of ciphersuites for RTP and RTCP contain at least one non-NULL encryption or authentication algorithm each, before sending the response message, an MTA must:

  1) If a Pad was included in the RemoteConnectionDescriptor and it is different than a Pad that may have previously been received, remove any existing inbound RTP keys and generate new ones, based on the keys that are generated from both the End-End Secret (generated locally) and the Pad (generated by the other MTA). The MTA must re-initialize the RTP timestamp if new keys are generated. The ciphersuites used for these inbound keys are taken from the RemoteConnectionDescriptor parameter just received from the CMS.

  2) If a Pad was included in the RemoteConnectionDescriptor and it is different than a Pad that may have previously been received, remove any existing inbound RTCP keys and generate new ones, based on the keys that are generated from both the End-End Secret (generated locally) and the Pad (generated by the other MTA). The MTA must re-initialize RTCP sequence numbers if new keys are generated. The ciphersuites used for these inbound keys are taken from the RemoteConnectionDescriptor parameter just received from CMS.

  3) If the RemoteConnectionDescriptor parameter was received without a Pad, check if the first RTP ciphersuite field in the RemoteConnectionDescriptor parameter differs from the one that the MTA originally selected. Also, check to see if a Pad had been previously received. If the ciphersuites differ, or if a Pad had been previously received, perform the following steps:

     a) Remove any existing inbound RTP key.

     b) If the new RTP ciphersuite is non-NULL, generate new inbound RTP keys and RTP timestamp from the same End-End Secret (generated locally) as the last time, as specified in clause 7.6.2.3.3.1.

  4) If the RemoteConnectionDescriptor parameter was received without a Pad, check if the first RTCP ciphersuite field in the RemoteConnectionDescriptor parameter differs from the one that the MTA originally selected. Also, check to see if a Pad had been previously received. If the ciphersuites differ, or if a Pad had been previously received, perform the following steps:

     a) Remove any existing inbound RTCP key.

     b) If the new RTCP ciphersuite is non-NULL, generate new inbound RTCP keys from the same End-End Secret (generated locally) as the last time, as specified in clause 7.6.2.3.3.1, and reset the RTCP sequence number to 0.

  5) If the End-End Secret included in the RemoteConnectionDescriptor has changed or the negotiated RTP ciphersuite has changed, perform the following steps:

     a) Remove any existing outbound RTP keys.

     b) If the new list of RTP ciphersuites is non-NULL, generate new outbound RTP keys, based on the End-End Secret (generated by the other MTA) and the Pad (generated locally), and generate a new RTP timestamp.

6) If the End-End Secret included in the RemoteConnectionDescriptor has changed or the negotiated RTCP ciphersuite has changed, perform the following steps:

    a) Remove any existing outbound RTCP keys.

    b) If the new list of RTCP ciphersuites is non-NULL, generate new outbound RTCP keys, based on the End-End Secret (generated by the other MTA) and the Pad (generated locally), and reset the RTCP sequence number to 0.

7) Be ready to send RTCP messages to and receive RTCP messages from the remote MTA. If connection mode allows, be ready to send and receive RTP messages with the remote MTA. If the list of ciphersuites for RTP was sent within a ModifyConnection command, the CMS may send an inactive directive to the MTA in the same command. The MTA should be returned to active status only when the new ciphersuite negotiation is complete.

- If, in response to a ModifyConnection command that does not include a RemoteConnectionDescriptor, and negotiated lists of ciphersuites for RTP and RTCP contain at least one non-NULL encryption or authentication algorithm each, before sending the response message, an MTA must:

    1) If the first RTP ciphersuites field in the negotiated list differs from the one that the MTA previously selected, then perform the following steps:

        a) Remove any existing inbound RTP keys.

        b) Generate new inbound RTP keys from the previous End-End Secret (locally generated) and Pad (generated by the other MTA), and generate a new RTP timestamp.

    2) If the first RTCP ciphersuites field in the negotiated list differs from the one that the MTA previously selected, then perform the following steps:

        a) Remove any existing inbound RTCP keys.

        b) Generate new inbound RTCP keys from the previous End-End Secret (locally generated) and Pad (generated by the other MTA), and reset the RTCP sequence number to 0.

    3) Be ready to send RTCP messages to and receive RTCP messages from the remote MTA. If connection mode allows, be ready to send and receive RTP messages with the remote MTA. If the list of ciphersuites for RTP was sent within a ModifyConnection command, the CMS may send an inactive directive to the MTA in the same command. The MTA should be returned to active status only when the new ciphersuite negotiation is complete.

- If an MTA receives a ModifyConnection command, and the resulting intersection of ciphersuites results in NULL encryption and authentication algorithms for RTP and RTCP, then the MTA must remove any existing RTP and RTCP keys and do not perform security on the RTP and RTCP packets.

- If an MTA returns a LocalConnectionDescriptor parameter, it must return the latest negotiated list of ciphersuites.

The following message flow is informative. Each of the numbered flows in figure 20 is described below:

(1) CMS_0 -> MTA_0

CMS0 may send the allowable lists of ciphersuites for the new communication to MTA0 in the CreateConnection (CRCX) command, inside the LocalConnectionOptions parameter, if the CMS has been configured to do so. The ciphersuites are provided in the order of preference specified by the operator subject to the constraints specified in clause 7.6.2.3.1.1. There can be two lists of ciphersuites, one list for RTP security and one for RTCP security. Each of these two lists may be included to specify the list of allowable ciphersuites, however ciphersuite negotiation will take place for both RTP and RTCP irrespective of whether the lists are included or not.

If RTP and/or RTCP ciphersuites are included but do not adhere to the rules provided in clause 7.6.2.3.1.1, the MTA returns an error, e.g. 524 (Internal inconsistency in LocalConnectionOptions).

(2) MTA_0 -> CMS_0

MTA_0 performs ciphersuite negotiation according to the ciphersuite negotiation procedure described above, and returns a non-empty list of RTP ciphersuites in the response message. This list contains the list of MTA_0's list of allowed ciphersuites in the order of preference specified by CMS_0 if the LocalConnectionOptions ciphersuites parameter(s) is included in step (1), as specified above. If RTP or RTCP ciphersuite negotiation fails, MTA_0 returns an error code as specified above.

If the lists of negotiated ciphersuites for RTP and RTCP contain at least one non-NULL combination each, MTA0 generates the End-End $Secret_0$ and Pad_1 value and returns them along with the ciphersuites in the LocalConnectionDescriptor parameter. For further details on the NCS message syntax, refer to [2]. Note that the NULL authentication and NULL encryption combinations will be at the end of each ciphersuite list.

The response message also includes the ConnectionId and the EndpointId for MTA_0 as described in [2]. The pair (ConnectionId, EndpointId) uniquely identifies this connection, where the EndpointId is an NCS identifier for MTA_0.

If the list of ciphersuites for RTP contains at least one non-NULL encryption or authentication algorithm, before sending the response message, MTA_0 must:

1) Establish inbound RTP security based on its preferred (first) RTP ciphersuite and End-End $Secret_0$, as described in clause 7.6.2.3.3.1 of the present document.

2) If connection mode allows, be ready to receive RTP packets, which may arrive any time after this message is sent by the MTA_0. If the list of ciphersuites for RTP was sent within a ModifyConnection command, the CMS may send an inactive directive to the MTA in the same command. The MTA should be returned to active status only when the new ciphersuite negotiation is complete.

If the list of ciphersuites for RTCP contains at least one non-NULL encryption algorithm, before sending the response message, MTA_0 must:

1) Establish inbound RTCP security based on its preferred (first) RTCP ciphersuite and End-End $Secret_0$, as described in clause 7.6.2.3.3.1 of the present document.

2) Be ready to receive RTCP packets, which may arrive any time after this message is sent by MTA_0.

If MTA_1 decides to use an alternate ciphersuite listed by MTA_0, MTA_0 will later have to update its RTP and RTCP keys. If MTA_1 decides to send MTA_0 packets before ciphersuite negotiation had completed, processing on those packets at MTA_0 will fail (since it assumed a different ciphersuite). If media stream security is disabled (AUTH_NULL/RTP_ENCR_NULL ciphersuite list for RTP and RTCP_AUTH_NULL/RTCP_ENCR_NULL for RTCP), MTA_0 will later have to discard its keys and send and receive RTP and RTCP packets without any security.

(3) CMS_0 -> CMS_1

CMS_0 must send End-End $Secret_0$ (if included), Pad_1 (if included) and the list of RTP and RTCP ciphersuites to CMS_1 (local to MTA_1) as selected by MTA_0. CMS_1 will later forward this information to MTA_1. Note that End-End $Secret_0$ and Pad_1 will not be included if the RTP and RTCP ciphersuites lists both contain only the NULL authentication and NULL encryption combination.

(4) CMS_1 -> MTA_1

CMS_1 sends a CreateConnection to MTA_1. CMS_1 may provide lists of approved RTP and RTCP ciphersuites, if the CMS has been configured to do so. The ciphersuites are provided in the order of preference specified by the operator subject to the constraints specified in clause 7.6.2.3.1.1. The RemoteConnectionDescriptor must be included in this CRCX command. It must contain End-End $Secret_0$ (if sent in step (3))and Pad_1 (if sent in step (3)) received from MTA_0 (via CMS_0). It must also contain the ciphersuites preferred by MTA_0.

(5) MTA_1 -> CMS_1

MTA_1 has received a CRCX message that contains both LocalConnectionOptions and RemoteConnectionDescriptor parameters and must follow the ciphersuite negotiation procedure described above to negotiate RTP and RTCP ciphersuites. This list will consist of MTA_1's allowed ciphersuites in the order of preference specified by CMS_1 if the LocalConnectionOptions ciphersuites parameter is included in step (4). If RTP and RTCP ciphersuite negotiation succeeds and there is at least one RTP ciphersuite and at least one RTCP ciphersuite, then MTA_1 returns the negotiated list of ciphersuites in the subsequent response message, in the LocalConnectionDescriptor parameter, in the form of SDP attributes. Note that if media stream security is being disabled, the NULL authentication and NULL encryption combination will be the only entry in both the RTP and RTCP ciphersuites lists. If RTP or RTCP ciphersuite negotiation fails, MTA_1 must return an error code as specified above.

In the event that MTA_1 receives SDP in the RemoteConnectionDescriptor parameter without ciphersuites media attributes, MTA_1 assumes that the lists of RTP and RTCP ciphersuites supported by the remote endpoint is RTP AUTH_NULL/RTP_ENCR_NULL and RTCP_AUTH_NULL/RTCP_ENCR_NULL.

If the RTP and RTCP ciphersuites provided do not adhere to the rules provided in clause 7.6.2.3.1.1, the MTA returns an error, e.g. 524 (Internal inconsistency in LocalConnectionOptions).

Whenever possible, MTA_1 should select the first supported ciphersuite for RTP and the first supported ciphersuite for RTCP in the RemoteConnectionDescriptor parameter. This allows MTA_1 to immediately start sending RTP and RTCP packets to MTA_0. MTA_1 may instead select alternate ciphersuites specified by MTA_0. MTA_1 returns a response message, which includes lists of the selected ciphersuites inside the LocalConnectionDescriptor parameter, in the form of SDP attributes. The first ciphersuite in each list (one for RTP and one for RTCP) must be the one that was selected by MTA_1. Additional ciphersuites in each list are alternatives in a prioritized order. If at any time, MTA_0 wants to switch to one of the alternatives that were selected by MTA_1, it would have to go through a new key negotiation. The response message must also include the ConnectionId (generated by MTA_1) as specified in [2]. Thus, both End-End $Secret_0$ and End-End $Secret_1$ are now associated with a pair (EndpointId, ConnectionId).

If the lists of ciphersuites for RTP and RTCP contain at least one non-NULL selection each, then MTA_1 must generate the End-End $Secret_1$ for the incoming RTP and RTCP packets, MTA_1 must and return it along with the ciphersuite lists in the LocalConnectionDescriptor parameter. If the lists of ciphersuites for RTP and RTCP contain at least one non-NULL selection each, MTA_1 should also generate Pad_0 and return it in the same LocalConnectionDescriptor parameter.

Although the option of not generating Pad_0 is provided in order to better support early media flows from MTA_1, it results in MTA_1 using a send key that is completely dependent on a random value generated by MTA_0. In other words, privacy of the media stream generated by MTA_1 in this case depends on the strength of MTA_0's random number generator.

If the list of ciphersuites for RTP contains at least one non-NULL encryption or authentication algorithm, before sending the response message, MTA_1 must:

1) Establish inbound RTP security based on its selected RTP ciphersuite, End-End $Secret_1$ and Pad_1, as described in clause 7.6.2.3.3.1 of the present document.

2) Establish outbound RTP security based on its selected RTP ciphersuite and End-End $Secret_0$, as described in clause 7.6.2.3.3.1 of the present document. If Pad_0 was generated by MTA_1, the outbound RTP security will also be based on Pad_0.

3) If connection mode allows, be ready to receive RTP packets, which may arrive from MTA_0 any time after this message is sent.

If the list of ciphersuites for RTCP contains at least one non-NULL encryption or authentication algorithm, before sending the response message, MTA1 must:

1) Establish inbound RTCP security based on its selected RTCP ciphersuite, End-End Secret1 and Pad1 as described in clause 7.6.2.3.3.1 of the present document.

2) Establish outbound RTCP security based on its selected RTCP ciphersuite and End-End Secret0, as described in clause 7.6.2.3.3.1 of the present document. If Pad0 was generated by MTA1, the outbound RTCP security will also be based on Pad0.

3) Be ready to receive RTCP messages, which may arrive from MTA0 any time after this message is sent.

Any time after sending this response message to the CMS_1, MTA_1 may begin sending RTP and RTCP packets to MTA_0. However, in the case that MTA_1 generated Pad_0 or selected a different ciphersuite from the one preferred by MTA_0, MTA_0 will not be able to decrypt packets from MTA_1, until MTA_0 has received MTA_1's SDP.

(6) CMS_1 -> CMS_0

CMS_1 must forward the End-End $Secret_1$, (if included) Pad_0 (if included) and the selected ciphersuites sent from MTA_1 to CMS_0. Note that End-End $Secret_0$ and Pad_1 will not be included if the RTP and RTCP ciphersuites lists both contain only the NULL authentication and NULL encryption algorithm combination.

(7) CMS_0 -> MTA_0

CMS_0 may send to MTA_0 in the ModifyConnection command, inside the LocalConnectionOptions parameter, the lists of allowed RTP and RTCP ciphersuites. These ciphersuites should be what CMS_0 policy allows. (The reason that CMS_0 is not required to send the lists of ciphersuites is because it might have already sent them to MTA_0 in a CreateConnection command. CMS_0 would send the ciphersuites again for consistency

In the event that MTA_0 receives SDP in the RemoteConnectionDescriptor parameter without fields containing ciphersuites media attributes, MTA_0 assumes that the RTP and RTCP ciphersuite lists supported by the remote endpoint are AUTH_NULL/RTP_ENCR_NULL for RTP and RTCP_AUTH_NULL/RTCP_ENCR_NULL for RTCP.

In the event that CMS_0 received SDP from MTA_1, the RemoteConnectionDescriptor parameter must be included in this ModifyConnection command. If present, it must contain the RTP and RTCP ciphersuites (and alternatives) selected by MTA_1. If ciphersuites are included in the LocalConnectionOptions parameter or a RemoteConnectionDescriptor parameter is included with the ModifyConnection command, MTA_0 must perform ciphersuite negotiation as described above.

If the RemoteConnectionDescriptor is not sent in this MDCX command, MTA_0 will still be able to receive RTP and RTCP messages but will be unable to send anything to MTA_1.

After receiving this message, MTA_0 must:

1) If Pad_0 was received, remove its inbound RTP keys and replace them with new ones, based on the keys that are generated from both End-End $Secret_0$ and Pad_0. Re-initialize the RTP timestamp for the new keys. The ciphersuites used for these inbound keys are taken from the RemoteConnectionDescriptor just received from CMS_0.

2) If Pad_0 was received, remove its inbound RTCP keys and replace them with new ones, based on the keys that are generated from both End-End $Secret_0$ and Pad_0. Re-initialize RTCP sequence numbers for the new keys. The ciphersuites used for these inbound keys are taken from the RemoteConnectionDescriptor just received from CMS_0.

3) If the RemoteConnectionDescriptor was received without Pad_0, check if the first RTP ciphersuite in the RemoteConnectionDescriptor differs from the one that MTA_0 selected in step (2). If they differ, perform the following steps:

   a) Remove the inbound RTP key.

   b) If the new RTP ciphersuite is non NULL, generate new inbound RTP keys and RTP timestamp from the same End-End $Secret_0$ as the last time, as specified in clause 7.6.2.3.3.1.

4) If the RemoteConnectionDescriptor parameter was received without Pad_0, check if the first RTCP ciphersuite field in the RemoteConnectionDescriptor parameter differs from the one that MTA_0 selected in step (2). If they differ, perform the following steps:

   a) Remove the inbound RTCP key.

   b) If the new RTCP ciphersuite is non NULL, generate a new key based on the key generated from the same End-End Secret0 as the last time, but for the new authentication and/or encryption algorithms.

5) If the RemoteConnectionDescriptor parameter was received, establish outbound RTP keys, based on End-End Secret1 and Pad1.

6) If the RemoteConnectionDescriptor parameter was received, establish outbound RTCP keys, based on End-End Secret1 and Pad1.

7) Be ready to send and receive RTCP messages with MTA1. If connection mode allows, be ready to send and receive RTP messages with MTA1.

For full syntax of the NCS messages, please refer to the NCS signalling specification [2].

7.6.2.3.2 Ciphersuite Format

Each ciphersuite for both RTP security and RTCP security 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 6.6 for RTP security and 6.7 for RTCP security. For the exact syntax of how the Authentication Algorithm and the Encryption Transform ID are included in the signalling messages, refer to [2] for NCS.

7.6.2.3.3 Derivation of End-to-End Keys

7.6.2.3.3.1 Initial Key Derivation

The End-End Secrets must be 46 bytes long. The Pad parameters must be 46 bytes long.

Keys are independently derived by each MTA from either just the End-End Secret or from the End-End Secret and Pad concatenated together. The Pad may or may not be available - see the call flow details specified in clause 7.6.2.3.1.

The keys derived from one End-End Secret (and possibly a Pad) must be used to secure RTP and RTCP messages directed to only one of the MTAs. There is a separate End-End Secret and a separate Pad value for each direction, negotiated through NCS signalling. The keys must be derived as follows, in the specified order:

1) RTP (media stream security). Derive a set of the following keys with the derivation function F(S, "End-End RTP Security Association"). Here, S is concatenation of the following binary values, each in MSB-first order:

   a) End-End Secret.

   b) Pad (optional, if it was negotiated through signalling).

   The string "End-End RTP Security Association" is taken without quotes and without a terminating null character. Function F (specified in clause 9.6) is used to recursively generate enough random bytes to produce all of the keys and other parameters that are specified below, in the listed order:

   a) RTP privacy key.

   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). The length must be the same as the selected cipher's block size. This value is used to derive the IV according to clause 7.6.2.1.2.2. 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 7.6.2.1.2.1.1.

2) RTCP security. Derive a set of the following keys in the specified order with the derivation function F(S, "End-End RTP Control Protocol Security Association"). Here, S is concatenation of the following binary values:

   a) End-End Secret.

   b) Pad (optional, if it was negotiated through signalling).

Function F (specified in clause 9.6) is used to recursively generate enough random bytes to produce all of the keys that are specified below, in the listed order:

a) RTCP authentication key.

b) RTCP encryption key.

7.6.2.4 RTP-RTCP Summary Security Profile Matrix

Table 27: Security Profile Matrix - RTP and RTCP

	RTP (MTA - MTA, MTA - MG)	**RTCP (MTA - MTA, MTA - MG, MG - MG)**
authentication	optional (indirect) (see note)	optional (indirect)
access control	optional	optional
integrity	optional	optional
confidentiality	optional	optional
non-repudiation	no	no
security mechanisms	Application Layer Security via RTP IPCablecom Security Profile End-to-End Secret distributed over secured MTA-CMS links. Final keys derived from this secret. AES-128 in CBC mode encryption algorithm Optional 2-byte or 4-byte MAC based on MMH algorithm RTP encryption and authentication can be optionally turned off with the selection of NULL encryption and NULL authentication algorithms. RTP security and RTCP security are disabled together. IPCablecom requires support for ciphersuite negotiation.	RTCP messages are secured by RTCP application layer security mechanisms specified in the profile. RTCP ciphersuites are negotiated separately from the RTP ciphersuites and include both encryption and message authentication algorithms. RTCP encryption can be optionally turned off with the selection of a null encryption algorithm. Both RTCP encryption and authentication can be optionally turned off with the selection of NULL encryption and NULL authentication algorithms. RTCP security and RTP security are disabled together. Keys are derived from the end-end secret using the same mechanism as used for RTP encryption.
NOTE: MTAs do not authenticate directly. Authentication refers to the authentication of identity.		

7.7 Audio Server Services

7.7.1 Reference Architecture

Figure 21 shows the network components and the various interfaces to be discussed in this clause, see [27].

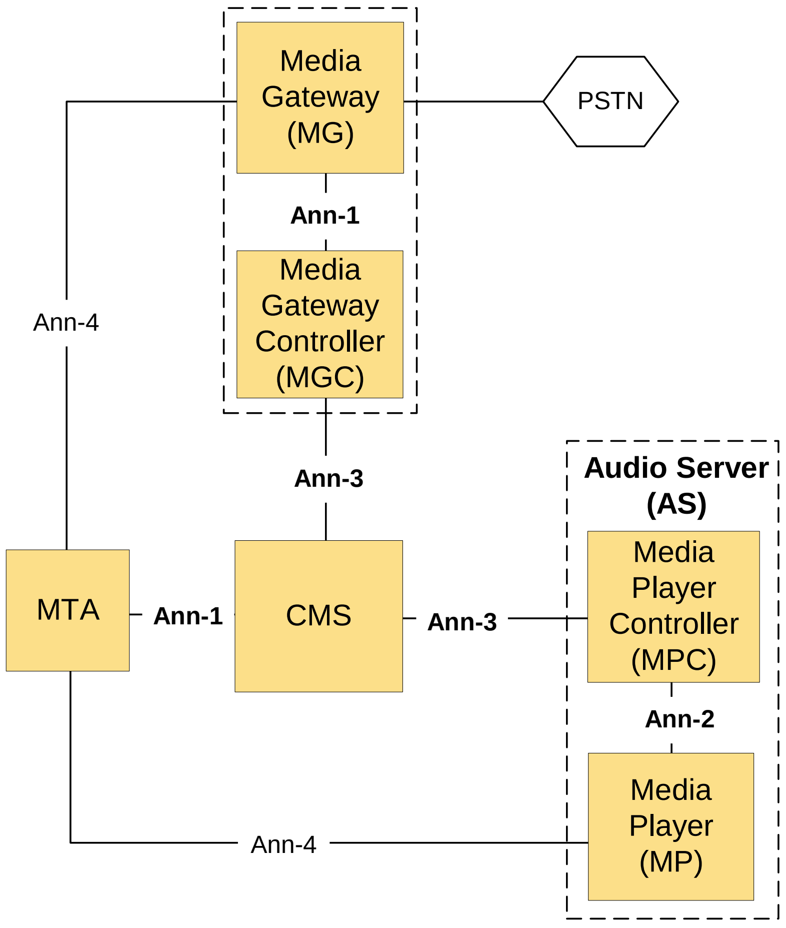


Figure 21: Audio Server Components and Interfaces

Figure 21 shows a network-based Media Player (MP). It has an optional Audio Server Protocol (ASP) interface (Ann-2) to the Media Player Controller (MPC), in the case that MPC and MP 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 and between the Media Gateway Controller (MGC) and the Media Gateway (MG). Refer to clause 7.4.1 for NCS signalling security. There is also a signalling interface (Ann-3) between the CMS and the MPC and the CMS and the MGC. 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 (Ann-4) between the MTA and the MP. This is a standard media stream interface, for which security is defined in clause 6.6 of the present document.

The Audio Server Architecture 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.

7.7.2 Security Services

7.7.2.1 MTA-CMS NCS Signalling (Ann-1)

Refer to the security services in the NCS signalling clause 7.4.1.2 of the present document.

7.7.2.2 MPC-MP Signalling (Ann-2)

Authentication: all signalling messages must be authenticated, in order to prevent a third party masquerading as either an authorized MPC or MP. A rogue MPC could configure the MP to play obscene or inappropriate messages. A rogue MP could likewise play obscene or inappropriate messages that the MPC did not intend it to play. If MP is unable to authenticate to the MPC, the MPC should not pass it the key for media packets, preventing unauthorized announcement playout.

Confidentiality: if a snooper is able to monitor ASP 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 MPC should keep a list of valid Media Players and which announcements each supports. Along with authentication, this insures that wrong announcements are not played out.

7.7.2.3 MTA-MP (Ann-4)

Security services on this media packet interface are listed in clause 7.6.1.

7.7.3 Cryptographic Mechanisms

7.7.3.1 MTA-CMS NCS Signalling (Ann-1)

Refer to the cryptographic mechanisms in the NCS signalling clause 7.4.1.3 of the present document.

7.7.3.2 MPC-MP Signalling (Ann-2)

IPsec ESP must be used to both authenticate and encrypt the messages from MPC to MP and vice versa. Refer to clause 6.1.2 for details of how IPsec ESP is used within IPCablecom and for the list of available ciphersuites.

7.7.3.3 MTA-MP (Ann-4)

Cryptographic mechanisms on this media packet interface are specified in clause 7.6.2.

7.7.4 Key Management

7.7.4.1 MTA-CMS NCS Signalling (Ann-1)

Refer to the key management in the NCS signalling clause 7.4.1.

7.7.4.2 MPC-MP Signalling (Ann-2)

The MPC and the MP negotiate a shared secret (MPC-MP Secret) using IKE or Kerberos (implementations must support IKE with pre-shared keys; they may support IKE with X.509 certificates and they may support Kerberos using symmetric keys). For more information on the IPCablecom use of IKE, refer to clause 6.2.2. For more information on the IPCablecom use of Kerberos with symmetric keys, refer to clauses 6.4.3 and 6.5.

The key management protocol must be running asynchronous to the signalling messages and will guarantee that there is always a valid, non-expired MPC-MP Secret.

7.7.4.3 MTA-MP (Ann-4)

Key Management on the media packet interface is specified in clause 7.6.2.3. 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 must be the same, except that here MG is replaced with the MP and MGC (which delivers the key to MG) is replaced with MPC (which delivers the key to MP).

7.7.5 MPC-MP Summary Security Profile Matrix

The CMS to MPC 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 28: Security Profile Matrix - Audio Server Services

	Ann-1: NCS (MTA - CMS) and (MG - MGC)	**Ann-2: ASP (MPC-MP)**	**Ann-3: unspecified (CMS-MPC) and (CMS - MGC) Interface Security Requirement**	**Ann-4: RTP (MTA-MP)**	**Ann-4: RTCP (MTA-MP)**
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 for MTA - CMS interface IKE or Kerberos for MG - MGC interface	IPsec IKE or Kerberos		Application Layer Security via RTP Packet Cable Security Profile keys distributed over secured MTA-CMS and MP-MPC links AES-128 encryption algorithm Optional 2-byte or 4-byte MAC based on MMH algorithm.	RTCP messages are secured by RTCP application layer security mechanisms specified in the profile. Keys are derived from the end-end secret using the same mechanism as used for RTP encryption.
NOTE: Although (CMS - MPC) is a proprietary interface, the following are security requirements for the CMS-MPC interface.					

7.8 Lawful Interception Interfaces

7.8.1 Reference Architecture

The IPCablecom system for Lawful Interception (see [30]) consists of the following elements and interfaces:

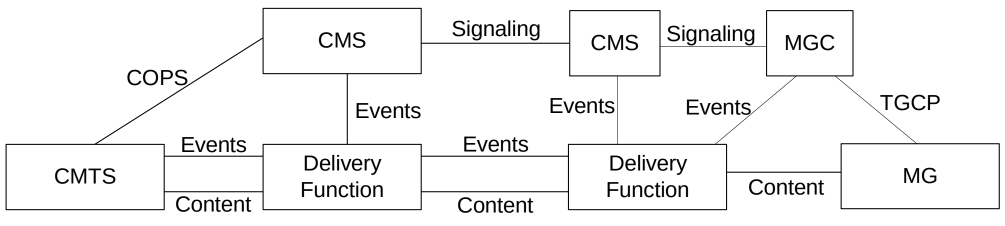


Figure 22: Lawful Interception Security Interfaces

The DF (Delivery Function) in this diagram is responsible for redirecting duplicated media packets to law enforcement, for the purpose of wiretapping.

The event interface between the CMS or the MGC and the DF provides descriptions of calls, which is 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 CMTS is used to signal the CMTS to start/stop duplicating media packets to the DF for a particular call. This is the same COPS interface that is used for (DQoS) Gate Authorization messages. For the corresponding security services, refer to clauses 7.2.1.2.2, 7.2.1.3.2 and 7.2.1.4.1.

The TGCP signalling interface between the MGC and MG is used to signal the MG to start/stop duplicating media packets to the DF for a particular call. This is the same TGCP signalling interface that is used during call setup on the PSTN Gateway side. For the corresponding security services, refer to clauses 7.8.2.1, 7.8.3.1 and 7.8.4.1.

The event interface between the CMTS 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 CMTS.

The interface between the CMTS and DF for call content is where the CMTS 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. Similarly, there is no additional security applied to the call content interface between the MG and DF: the MG simply encapsulates copies of the encrypted RTP packets inside UDP and forwards them to the DF.

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.

7.8.2 Security Services

7.8.2.1 Event Interfaces CMS-DF, MGC-DF, CMTS-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.

7.8.2.2 Call Content Interfaces CMTS-DF, MG-DF, MG-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).

7.8.3 Cryptographic Mechanisms

7.8.3.1 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 7.3.3.1.

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 16 ASCII 0s.

7.8.3.2 Interface between CMTS 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 CMTS-RKS interface specified in clause 7.3.3.2.

As with the CMTS-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 16 ASCII 0s.

7.8.3.3 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 7.3.3.1.

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 16 ASCII 0s.

7.8.3.4 Interface between MGC 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 MGC-RKS interface specified in clause 7.3.3.3.

As with the MGC-RKS interface, the MAC value normally used to authenticate RADIUS messages is not used (message integrity is provided by IPsec). The key for this RADIUS MAC must always be hardcoded to 16 ASCII 0s.

7.8.4 Key Management

7.8.4.1 Interface between CMS and DF

The CMS and the DF must negotiate a pair of IPsec Security Associations (inbound and outbound) using IKE or Kerberos (implementations must support IKE with pre-shared keys; they may support IKE with X.509 certificates and they may support Kerberos using symmetric keys). For more information on the IPCablecom use of IKE, refer to clause 6.2.2. For more information on the IPCablecom use of Kerberos with symmetric keys, refer to clauses 6.4.3 and 6.5.

The key management protocol will be running asynchronous to the event message generation, and will guarantee that there is always a valid, non-expired pair of Security Associations.

7.8.4.2 Interface between CMTS and DF

The CMTS and the DF must negotiate a pair of Security Associations (inbound and outbound) using IKE or Kerberos (implementations must support IKE with pre-shared keys; they may support IKE with X.509 certificates and they may support Kerberos using symmetric keys). For more information on the IPCablecom use of IKE, refer to clause 6.2.2. For more information on the IPCablecom use of Kerberos with symmetric keys, refer to clauses 6.4.3 and 6.5.

The key management protocol will be running asynchronous to the event message generation, and will guarantee that there is always a valid, non-expired pair of Security Associations.

7.8.4.3 Interface between DF and DF

The two DF hosts must negotiate a shared secret (DF-DF Secret) using IKE with certificates. The IPCablecom profile for IKE with certificates is specified in clause 6.2.2. 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 SA, 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 SAs 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 SAs (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 SAs 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 SAs.

It will still be possible (with very small probability) that a DF uses a 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 SAs.

7.8.4.4 INTERFACE BETWEEN MGC AND DF

MGC and the DF must negotiate a pair of IPsec SAs (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.

At the DF, MGC 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 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. One way to query this database is through SNMP, using an IPsec MIB.

7.8.5 Lawful Interception Security Profile Matrix

Table 29: Security Profile Matrix - Lawful Interception

	CMS-DF Events, MGC-DF Events and CMTS-DF Events	DF-DF Events	CMTS-DF Content and MG-DF Content	DF-DF Content
Authentication	yes	yes	yes (indirect)	yes (indirect)
Access control	yes	yes	optional	optional
Integrity	yes	yes	optional	optional
Confidentiality	yes	yes	yes	yes
Non-repudiation	no	no	no	no
Security mechanisms	IPsec with encryption and message integrity enabled Key management is IKE or Kerberos	IPsec with encryption and message integrity enabled Key management is IKE with certificates	RTP packets are already encrypted and authenticated by the source (MTA or MG)	RTP packets are already encrypted and authenticated by the source (MTA or MG)

7.9 CMS Provisioning

7.9.1 Reference Architecture

Provisioning is defined as the operations necessary to provide a specified service to a customer. IPCablecom service provisioning can be viewed as two distinct operations: MTA provisioning and CMS subscriber provisioning. CMS provisioning refers to the interface between the Provisioning Server and the CMS.

7.9.2 Security Services

Authentication: Provisioning Server needs to be authenticated to prevent a third party from masquerading as a provisioning server to enable services for unauthorized MTAs. CMS needs to be authenticated to prevent someone from impersonating the CMS to receiving provisioning messages, thereby compromising privacy and deny service to provisioned MTAs.

Access Control: required along with authentication to prevent unauthorized access to provisioning data as well as denial-of-service.

Integrity: must be assured to disallow tampering with provisioning messages, in order to prevent a class of denial-of-service attacks.

Confidentiality: Provisioning messages contains private customer information, thus confidentiality is required.

7.9.3 Cryptographic Mechanisms

IPsec ESP must be used to both authenticate and encrypt the messages from CMS to Provisioning Server and vice versa. Refer to clause 6.1.2 for details of how IPsec ESP is used within IPCablecom and for the list of available ciphersuites.

7.9.4 Key Management

Key management for the CMS-Provisioning Server interface is either IKE or Kerberos. Implementations must support IKE with pre-shared keys. Implementations may support IKE with X.509 certificates and they may support Kerberos using symmetric keys. For more information on the IPCablecom use of IKE, refer to clause 6.2.2. For more information on the IPCablecom use of Kerberos with symmetric keys, refer to clauses 6.4.3 and 6.5.

7.9.5 Provisioning Server-CMS Summary Security Profile Matrix

Table 30: Security Profile Matrix - CMS Provisioning

	CMS - Provisioning Server
Authentication	yes
Access control	yes
Integrity	yes
Confidentiality	yes
Non-repudiation	no
Security Mechanisms	IPsec IKE or Kerberos

8 IPCablecom Certificates

IPCablecom uses digital certificates, which comply with the X.509 specification [33] and the IETF PKIX specification [34].

8.1 Generic Structure

8.1.1 Version

The Version of the certificates must be V3. All certificates must comply with [34] except where the non-compliance with the RFC is explicitly stated in this clause of the present document.

8.1.2 Public Key Type

RSA Public Keys are used throughout the hierarchy. The subjectPublicKeyInfo.algorithm.algorithm Object Identifier (OID) used must be 1.2.840.113549.1.1.1 (rsaEncryption).

The public exponent for all RSA IPCablecom keys must be F4 - 65537.

8.1.3 Extensions

The following four extensions must be used as specified in the clauses below. Any other certificate extensions may also be included but must be marked as non-critical.

8.1.3.1 subjectKeyIdentifier

The subjectKeyIdentifier extension included in all IPCablecom CA certificates as required by [34] (e.g. all certificates except the device and ancillary certificates) must include the keyIdentifier value composed of the 160-bit SHA1 hash of the value of the BIT STRING subjectPublicKey (excluding the tag, length and number of unused bits from the ASN1 encoding) (see [34]).

8.1.3.2 authorityKeyIdentifier

The authorityKeyIdentifier extension must be included in all IPCablecom certificates, with the exception of root certificates, and must include a keyIdentifier value that is identical to the subjectKeyIdentifier in the CA certificate.

8.1.3.3 KeyUsage

The KeyUsage extension must be used for all IPCablecom CA certificates and must be marked as critical with a value of keyCertSign and cRLSign. A KeyUsage extension may be included in end-entity certificates and should be marked as critical if included as specified in [34].

8.1.3.4 BasicConstraints

The basicConstraints extension must be used for all IPCablecom CA certificates and must be marked as critical. The values for each certificate for basicConstraints must be marked as specified in each of the certificate description tables (tables 31, 32, 33, 34, 35, 36, 37, 38, 39 and 40).

8.1.4 Signature Algorithm

The signature mechanism used must be SHA-1 with RSA Encryption. The specific OID is 1.2.840.113549.1.1.5.

8.1.5 SubjectName and IssuerName

If a string cannot be encoded as a PrintableString it must be encoded as a UTF8String (tag [UNIVERSAL 12]).

When encoding an X.500 Name:

1) Each RelativeDistinguishedName (RDN) must contain only a single element in the set of X.500 attributes.

2) The order of the RDNs in an X.500 name must be the same as the order in which they are presented in the present document.

It should be noted that [34] and X.509 defines constraints (i.e. upper bounds) on the length of the attribute values. For example, the maximum length for common name (CN), organization name (O) and organizational unit (OU) name values is 64 characters. Where the present document mandates the inclusion of a static string in one of these values, (i.e. CN=<Company> IPCablecom System Operator CA) companies must ensure that the addition of their identifying information does not cause the total length of the value to exceed the upper bound. In the case where a company's name causes the length of the value to exceed the upper bound, the vendor must truncate or abbreviate their information to ensure the total length does not exceed the upper bound.

8.1.6 Certificate Profile Notation

The tables below use the following notation:

- Extension details are specified by [c:critical, n:non-critical; m:mandatory, o:optional].
- Optional subject naming attributes are surrounded by square brackets (e.g. [L = <city>]).
- Variable naming attribute values are surrounded by angle brackets. (e.g. CN = <Company Name> IPCablecom CA). Values not surrounded by angle brackets are static and cannot be modified.

8.2 Certificate Trust Hierarchy

There are two distinct certificate hierarchies used in IPCablecom.

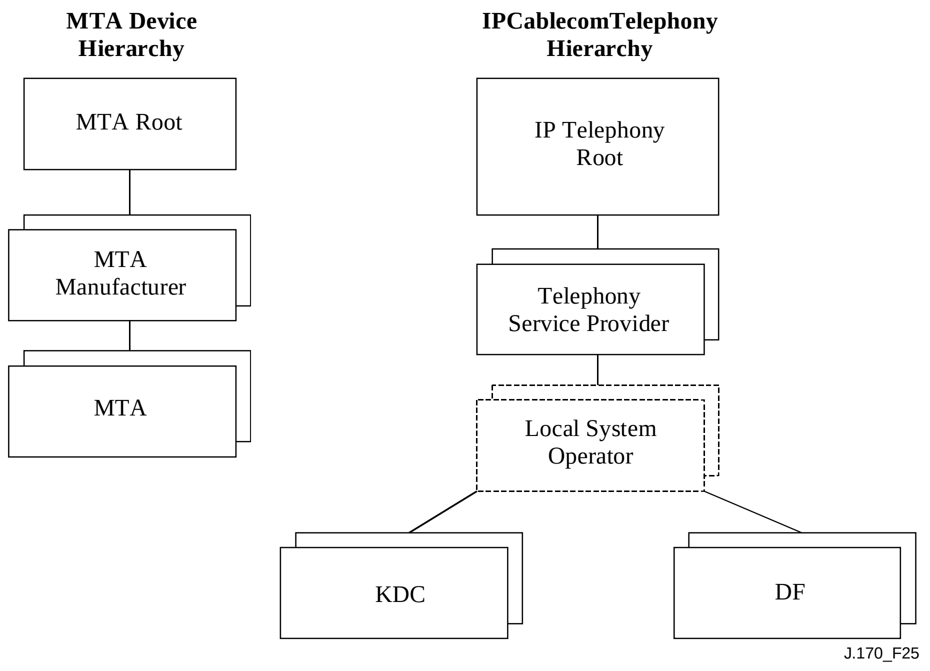


Figure 23: IPCablecom Certificate Hierarchy

8.2.1 Certificate Validation

Within IPCablecom certificate validation in general involves validation of a whole chain of certificates. As an example, when the Provisioning Server validates an MTA Device certificate, the actual chain of three certificates is validated:

MTA Root Certificate + MTA Manufacturer Certificate + MTA Device Certificate.

The signature on the MTA Manufacturer Certificate is verified with the MTA Root Certificate and the signature on the MTA Device Certificate is verified with the MTA Manufacturer Certificate. The MTA Root Certificate is self-signed and is known in advance to the Provisioning Server. The public key present in the MTA Root Certificate is used to validate the signature on this same certificate.

Usually the first certificate in the chain is not explicitly included in the certificate chain that is sent over the wire. In the cases where the first certificate is explicitly included it must already be known to the verifying party ahead of time and must NOT contain any changes to the certificate with the possible exception of the certificate serial number, validity period and the value of the signature. If changes other than the certificate serial number, validity period and the value of the signature, exist in the IP Telephony Root certificate that was passed over the wire in comparison to the known IP Telephony Root certificate, the device making the comparison must fail the certificate verification.

The exact rules for certificate chain validation must fully comply with [34], where they are referred to as "Certificate Path Validation". In general, X.509 certificates support a liberal set of rules for determining if the issuer name of a certificate matches the subject name of another. The rules are such that two name fields may be declared to match even though a binary comparison of the two name fields does not indicate a match. [34] recommends that certificate authorities restrict the encoding of name fields so that an implementation can declare a match or mismatch using simple binary comparison. An IPCablecom security follows this recommendation. Accordingly, the DER-encoded tbsCertificate.issuer field of an IPCablecom certificate must be an exact match to the DER-encoded tbsCertificate.subject field of its issuer certificate. An implementation may compare an issuer name to a subject name by performing a binary comparison of the DER-encoded tbsCertificate.issuer and tbsCertificate.subject fields.

The clauses below specify the required certificate chain, which must be used to verify each certificate that appears at the leaf node (i.e. at the bottom) in the IPCablecom certificate trust hierarchy illustrated in figure 23.

Validity period nesting is not checked and intentionally not enforced. Thus, the validity period of a certificate need not fall within the validity period of the certificate that issued it.

8.2.2 MTA Device Certificate Hierarchy

The device certificate hierarchy exactly mirrors that of the DOCSIS® 1.1/BPI+ hierarchy. It is rooted in an IPCablecom MTA Root certificate, which is used as the issuing certificate of a set of manufacturer's certificates. The manufacturer's certificates are used to sign the individual device certificates.

The information contained in the following tables contains the IPCablecom specific values for the required fields according to [34]. These IPCablecom specific values must be followed according to the table below, except that Validity Periods should be as given in the tables. If a required field is not specifically listed for IPCablecom then the guidelines in [34] must be followed.

8.2.2.1 MTA Root Certificate

This certificate must be verified as part of a certificate chain containing the MTA Root Certificate, MTA Manufacturer Certificate and the MTA Device Certificate.

Table 31: MTA Root Certificate

MTA Root Certificate	

Subject Name Form	C=US
	O=CableLabs
	OU=PacketCable
	CN=PacketCable Root Device Certificate Authority

Intended Usage	This certificate is used to sign MTA Manufacturer Certificates and is used by the KDC. This certificate is not used by the MTAs and thus does not appear in the MTA MIB
Signed By	Self-Signed
Validity Period	20+ Years. It is intended that the validity period is long enough that this certificate is never re-issued
Modulus Length	2 048
Extensions	keyUsage[c,m](keyCertSign, cRLSign) subjectKeyIdentifier[n,m] basicConstraints[c,m](cA=true, pathLenConstraint=1)

8.2.2.2 MTA Manufacturer Certificate

This certificate must be verified as part of a certificate chain containing the MTA Root Certificate, MTA Manufacturer Certificate and the MTA Device Certificate.

The state/province, city and manufacturer's facility are optional attributes. A manufacturer may have more than one manufacturer's certificate, and there may exist one or more certificates per manufacturer. All Certificates for the same manufacturer may be provided to each MTA either at manufacture time or during a field update. The MTA must select an appropriate certificate for its use by matching the issuer name in the MTA Device Certificate with the subject name in the MTA Manufacturer Certificate. If present, the authorityKeyIdentifier of the device certificate must be matched to the subjectKeyIdentifier of the manufacturer certificate as described in [34].

The <CompanyName> field that is present in O and CN may be different in the two instances.

Table 32: MTA Manufacturer Certificate

MTA Manufacturer Certificate	

Subject Name Form	C=<country>
	O=<CompanyName>
	[ST=<state/province>]
	[L=<city>]
	OU=PacketCable
	[OU=<Manufacturer's Facility>]
	CN=<CompanyName> PacketCable CA

Intended Usage	This certificate is issued to each MTA manufacturer and can be provided to each MTA as part of the secure code download as specified by the IPCablecom Security Specification (either at manufacture time, or during a field update). This certificate appears as a read-only parameter in the MTA MIB. This certificate along with the MTA Device Certificate is used to authenticate the MTA device identity (MAC address) during authentication by the KDC.
Signed By	MTA Root Certificate CA
Validity Period	20 Years
Modulus Length	2 048
Extensions	keyUsage[c,m](keyCertSign, cRLSign) subjectKeyIdentifier[n,m] authorityKeyIdentifier[n,m](keyIdentifier=<subjectKeyIdentifier value from CA certificate>) basicConstraints[c,m](cA=true, pathLenConstraint=0)

8.2.2.3 MTA Device Certificate

This certificate must be verified as part of a certificate chain containing the MTA Root Certificate, MTA Manufacturer Certificate and the MTA Device Certificate.

The state/province, city and manufacturer's facility are optional attributes. The Manufacturer's Facility OU field, (if present) may be different from the Manufacturer's Facility OU field (if present) in the MTA Manufacturer certificate.

The MAC address must be 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 MTA device certificate should not be replaced or renewed.

Table 33: MTA Device Certificate

MTA Device Certificate	

Subject Name Form	C=<country>
	O=<Company Name>
	[ST=<state/province>]
	[L=<city>]
	OU=PacketCable
	[OU=<Product Name>]
	[OU=<Manufacturer's Facility>]
	CN=<MAC Address>

Intended Usage	This certificate is issued by the MTA manufacturer and installed in the factory. The provisioning server cannot update this certificate. This certificate appears as a read-only parameter in the MTA MIB. This certificate is used to authenticate the MTA device identity (MAC address) during provisioning.
Signed By	MTA Manufacturer Certificate CA
Validity Period	At least 20 years
Modulus Length	1 024, 1 536 or 2 048
Extensions	keyUsage[c,o](digitalSignature, keyEncipherment) authorityKeyIdentifier[n,m](keyIdentifier=<subjectKeyIdentifier value from CA certificate>)

8.2.3 IPCablecom Telephony Certificate Hierarchy

The Service Provider Certificate Hierarchy is rooted in an IP Telephony Root certificate. That certificate is used as the issuing certificate of a set of service provider's certificates. The service provider's certificates are used to sign an optional local system certificate. If the local system certificate exists then that is used to sign the ancillary equipment certificates, otherwise the ancillary certificates are signed by the Service Provider's CA.

The information contained in the following table contains the IPCablecom specific values for the required fields according to [34]. These specific values must be followed according to table 34, except that Validity Periods should be as given in the certificate description tables (tables 31, 32, 33, 34, 35, 36, 37, 38, 39 and 40). If a required field is not specifically listed then the guidelines in [34] must be followed.

8.2.3.1 IP Telephony Root Certificate

Before any Kerberos key management can be performed, an MTA and a KDC need to perform mutual authentication using the PKINIT extension to the Kerberos protocol. An MTA authenticates a KDC after it receives a PKINIT Reply message containing a KDC certificate chain. In authenticating the KDC, the MTA verifies the KDC certificate chain, including KDC's Service Provider Certificate signed by the IP Telephony Root Certificate.

Table 34: IP Telephony Root Certificate

IP Telephony Root Certificate	

Subject Name Form	C=US
	O=CableLabs
	CN=CableLabs Service Provider Root CA

Intended Usage	This certificate is used to sign Service Provider CA certificates. This certificate is installed into each MTA at the time of manufacture or with a secure code download as specified by the IPCablecom Security Specification and cannot be updated by the Provisioning Server. Neither this root certificate nor the corresponding public key appears in the MTA MIB.
Signed By	Self-signed
Validity Period	20+. It is intended that the validity period is long enough that this certificate is never re-issued.
Modulus Length	2 048
Extensions	keyUsage[c,m](keyCertSign, cRLSign) subjectKeyIdentifier[n,m] basicConstraints[c,m](cA=true)

8.2.3.2 Service Provider CA Certificate

This is the certificate (see table 35) held by the telephony service provider, signed by the IP Telephony Root CA. It is verified as part of a certificate chain that includes the IP Telephony Root Certificate, Telephony Service Provider Certificate, optional Local System Certificate and an end-entity server certificate. The authenticating entities normally already possess the IP Telephony Root Certificate and it is not transmitted with the rest of the certificate chain.

The fact that a Telephony Service Provider CA Certificate is always explicitly included in the certificate chain allows a Service Provider the flexibility to change its certificate without requiring re configuration of each entity that validates this certificate chain (e.g. MTA validating a PKINIT Reply). Each time the Service Provider CA Certificate changes, its signature must be verified with the IP Telephony Root Certificate. However, a new certificate for the same Service Provider must preserve the same value of the OrganizationName attribute in the SubjectName.

The <Company> field that is present in O and CN may be different in the two instances.

Table 35: Service Provider CA Certificate

Service Provider CA Certificate	

Subject Name Form	C=<Country>
	O=<Company>
	CN=<Company> CableLabs Service Provider CA

Intended Usage	This certificate corresponds to a top-level Certification Authority within a domain of a single Service Provider. In order to make it easy to update this certificate, each network element is configured with the OrganizationName attribute of the Service Provider Certificate SubjectName. This is the only attribute in the certificate that must remain constant. In the case of an MTA, there is a read-write parameter in the MIB that identifies the OrganizationName attribute for each Kerberos realm (that may be shared among multiple MTA endpoints). The MTA does not accept Service Provider certificates that do not match this value of the OrganizationName attribute in the SubjectName. An MTA needs to perform the first PKINIT exchange with the MSO KDC right after a reboot, at which time its MIB tables are not yet configured. At that time, the MTA must accept any Service Provider OrganizationName attribute, but it must later check that the value added into the MIB for this realm is the same as the one in the initial PKINIT Reply.
Signed By	Signed by IP Telephony Root Certificate
Validity Period	20 years
Modulus Length	2 048
Extensions	keyUsage[c,m](keyCertSign, cRLSign) subjectKeyIdentifier[n,m] authorityKeyIdentifier[n,m](keyIdentifier=<subjectKeyIdentifier value from CA certificate>) basicConstraints[c,m](cA=true, pathLenConstraint=1)

8.2.3.3 Local System CA Certificate

This is the certificate (see table 36) held by the local system. The existence of this certificate is optional, as the Telephony Service Provider CA may be used to directly sign all network server end-entity certificates. A certificate chain with a Local System Certificate MUST consist of the IP Telephony Root CA Certificate, Service Provider CA Certificate, Local System CA Certificate and an end entity certificate.

The <Company> field that is present in O and CN may be different in the two instances.

Table 36: Local System CA Certificate

Local System CA Certificate	

Subject Name Form	C=<Country>
	O=<Company>
	OU=<Local System Name>
	CN=<Company> CableLabs Local System CA

Intended Usage	A Service Provider CA may delegate the issuance of certificates to a regional Certification Authority called Local System CA (with the corresponding Local System Certificate). Network servers are allowed to move freely between regional Certification Authorities of the same Service Provider. Therefore, the MTA MIB does not contain any information regarding a Local System Certificate (which might restrict an MTA to KDCs within a particular region).
Signed By	Service Provider CA Certificate
Validity Period	20 years
Modulus Length	1 024, 1 536, 2 048
Extensions	keyUsage[c,m](keyCertSign, cRLSign) subjectKeyIdentifier[n,m] authorityKeyIdentifier[n,m](keyIdentifier=<subjectKeyIdentifier value from CA certificate>) basicConstraints[c,m](cA=true, pathLenConstraint=0)

8.2.3.4 Operational Ancillary Certificates

All of these are signed by the either the Local System CA or by the Service Provider CA. Other ancillary certificates may be added to the present document at a later time.

8.2.3.4.1 Key Distribution Centre Certificate

This certificate must be verified as part of a certificate chain containing the IP Telephony Provider Root Certificate, Service Provider CA Certificate and the Ancillary Device Certificates.

The PKINIT specification in annex C requires the KDC certificate to include the subjectAltName v.3 certificate extension, the value of which must be the Kerberos principal name of the KDC.

Table 37: Key Distribution Centre Certificate

Key Distribution Centre Certificate	

Subject Name Form	C=<Country>
	O=<Company>
	[OU=<Local System Name>]
	OU= CableLabs Key Distribution Centre
	CN=<DNS Name>

Intended Usage	To authenticate the identity of the KDC server to the MTA during PKINIT exchanges. This certificate is passed to the MTA inside the PKINIT replies and is therefore not included in the MTA MIB and cannot be updated or queried by the Provisioning Server.
Signed By	Service Provider CA Certificate or Local System Certificate
Validity Period	20 years.
Modulus Length	1 024, 1 536 or 2 048
Extensions	keyUsage[c,o](digitalSignature) authorityKeyIdentifier[n,m](keyIdentifier=<subjectKeyIdentifier value from CA certificate>) subjectAltName[n,m] (See annex C)

8.2.3.4.2 Delivery Function (DF)

This certificate must be verified as part of a certificate chain containing the IP Telephony Root Certificate, Service Provider CA Certificate and the Ancillary Device Certificates.

This certificate is used to sign phase 1 IKE intra-domain exchanges between DFs (which are used in Lawful Interception, referred to in table 38 as Electronic Surveillance). Although Local System Name is optional, it is required when the Local System CA signs this certificate. The IP address must be specified in standard dotted-quad notation, e.g. 245.120.75.22.

Table 38: DF Certificate

DF Certificate	

Subject Name Form	C=<Country>
	O=<Company>
	[OU=<Local System Name>]
	OU=PacketCable Electronic Surveillance
	CN=<IP address>

Intended Usage	To authenticate IKE key management, used to establish IPsec Security Associations between pairs of DFs. These Security Associations are used when a subject that is being legally wiretapped forwards the call and event messages containing call info have to be forwarded to a new wiretap server (DF).
Signed By	Service Provider CA Certificate or Local System CA Certificate
Validity Period	20 years
Modulus Length	2 048
Extensions	keyUsage[c,o](digitalSignature) authorityKeyIdentifier[n,m](keyIdentifier=<subjectKeyIdentifier value from CA certificate>) subjectAltName[n,m](dNSName=<DNSName>)

8.2.3.4.3 IPCablecom Server Certificates

These certificates must be verified as part of a certificate chain containing the Service Provider Root Certificate, Service Provider Certificate, Local System Operator Certificate (if used) and the Ancillary Device Certificates.

These certificates are used to identify various servers in the IPCablecom system. For example, they may be used to sign phase 1 IKE exchanges or to authenticate a PKINIT exchange. Although the Local System Name is optional, it is required when the Local System CA signs this certificate. 2IP address values must be specified in standard dotted decimal notation: e.g. 245.120.75.22. DNS Name values must be specified as a fully qualified domain name (FQDN): e.g. device.packetcable.com.

Table 39: IPCablecom Server Certificates

IPCablecom Server Certificates	
Subject Name Form	C=<Country> O=<Company> OU=PacketCable OU=[<Local System Name>] OU=<Sub-System Name>[&<Sub-System Name>] CN=[<Server Identifier>] Or, CN=[<Element ID>][&<Element ID>] The CN will contain either a <Server Identifier> or one or more <Element ID>s. If the CN contains a <Server Identifier>, the value of <Server Identifier> must be the server's FQDN or its IP address, optionally followed by a colon (:) and an Element ID with no white space either before or after the colon. <Element ID> is the identifier that appears in billing event messages and it must be included in a certificate of every server that is capable of generating event messages. This includes a CMS, CMTS and MGC. There may be multiple <Element ID> fields, each separated by the character "&". [6] defines the Element ID as an 5-octet right-justified, space-padded ASCII-encoded numerical string. When converting the Element ID for use in a certificate, any spaces must be converted to ASCII zeroes (0x30). For example, a CMTS that has the Element ID " 311" will have a common name "00311". The value of <Sub-System Name> must be one of the following: • For Border Proxy: bp • For Cable Modem Termination System: cmts • For Call Management Server: cms • For Media Gateway: mg • For Media Gateway Controller: mgc • For Media Player: mp • For Media Player Controller: mpc • For Provisioning Server: ps • For Record Keeping Server: rks • For Signalling Gateway: sg Components that contain combined elements (such as a CMS with an integrated MGC) must indicate this in the Subject Name by including all Sub-System Names, joined with the character "&", in the OU field. In the case of combined elements, a single Element ID or multiple Element IDs may be used. If multiple Element IDs are used, all Element IDs must be included in the CN, and the order of these Element IDs must correspond to the order of the Sub-System Name fields in the OU. The following is an example OU and CN for a combined CMS and MGC. The CMS with Element ID " 311" and a MGC with Element ID " 312". OU=cms&mgc CN=00311&00312 The following is an example OU and CN for a combined CMS and MGC. In this case, the CMS and MGC share a single Element ID of " 311". OU=cms&mgc CN=00311&00311
Intended Usage	These certificates are used to identify various servers in the IPCablecom system. For example they may be used to sign phase 1 IKE exchanges or to authenticate a device in a PKINIT exchange
Signed By	Telephony Service Provider Certificate or Local System Certificate
Validity Period	Set by cable operator policy
Modulus Length	2 048
Extensions	keyUsage[c,o](digitalSignature, keyEncipherment) authorityKeyIdentifier[n,m](keyIdentifier=<subjectKeyIdentifier value from CA cert>) subjectAltName[n,o](dNSName=<DNSName> | iPAddress=<IP Address Name>) The keyUsage tag is optional. When it is used it must be marked as critical. The subjectAltName extension must be included for all servers that are capable of generating event messages. For all other servers, the subjectAltName extension may be included. If the subjectAltName extension is included, it must include the corresponding name value as specified in the CN field of the subject.

8.2.3.4.4 TLS Certificates

These certificates must be verified as part of a certificate chain containing the Service Provider Root Certificate, Service Provider Certificate, Local System Operator Certificate (if used) and the Ancillary Device Certificates.

These certificates are used to authenticate TLS handshake exchanges (and encrypt when using RSA key exchange). Although the Local System Name is optional, it is required when the Local System CA signs this certificate. DNS Name values must be specified as a fully qualified domain name (FQDN): e.g. device.packetcable.com.

Table 40: IPCablecom TLS Certificates

IPCablecom Server Certificates	

Subject Name Form	C=<Country>
	O=<Company>
	OU=[<Local System Name>]
	OU=PacketCable
	CN=[<Server Identifier>]
	
	The value of <Server Identifier> must be the server's FQDN. Note that only a single FQDN can be included in the CN field.

Intended Usage	These certificates are used to authenticate TLS handshake exchanges (and encrypt when using RSA key exchange).
Signed By	Telephony Service Provider Certificate or Local System Certificate
Validity Period	Set by cable operator policy
Modulus Length	1 024, 1 536, 2 048
Extensions	KeyUsage[c,m](digitalSignature, keyEncipherment) extendedKeyUsage[n,m] (id-kp-serverAuth, id-kp-clientAuth) authorityKeyIdentifier[n,m](keyIdentifier=<subjectKeyIdentifier value from CA cert>)

8.2.4 Certificate Revocation

Out of scope for IPCablecom at this time.

9 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.

9.1 AES

AES-128 is a 128-bit block cipher that must be implemented according to the AES (Advanced Encryption Standard) proposed submission specified in [35]. AES-128 is used in CBC mode with a 128-bit block size in IPCablecom. AES-128 requires 10 rounds of cryptographic operations in encryption or decryption. The Initialization Vector for CBC mode is specified for each use of AES in IPCablecom.

In 1997, the National Institute of Standards and Technology (NIST) initiated a process to select a symmetric-key encryption algorithm to be used to protect sensitive (unclassified) Federal information in furtherance of NIST's statutory responsibilities. In 1998, NIST announced the acceptance of fifteen candidate algorithms and requested the assistance of the cryptographic research community in analysing the candidates. This analysis included an initial examination of the security and efficiency characteristics for each algorithm. NIST reviewed the results of this preliminary research and selected MARS, RC6(tm), Rijndael, Serpent and Twofish as finalists. Having reviewed further public analysis of the finalists, NIST has decided to propose Rijndael as the Advanced Encryption Standard.

9.2 DES

The Data Encryption Standard (DES) is specified in [31] For Media Stream encryption, IPCablecom does not require error checking on the DES key, and the full 64-bits of key provided to the DES algorithm will be generated according to clause 7.6.2.3.3.1.

9.2.1 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 [45]. The CBC mode of DESX-XEX is shown a figure below, where DESX-XEX is executed within the block called "block cipher." Inside the block, DESX-XEX is performed as shown in a figure below 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.

9.2.2 DES-CBC-PAD

This variant of DES is also based on the analysis of DESX presented in [45]. 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 [31], 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 9.3), the left-most-bits of the IV are used in the final XOR padding operation. This mode of DES-CBC is shown a figure below, where DES is executed in the block called "block cipher." A 64-bit key value is used.

9.2.3 3DES-EDE

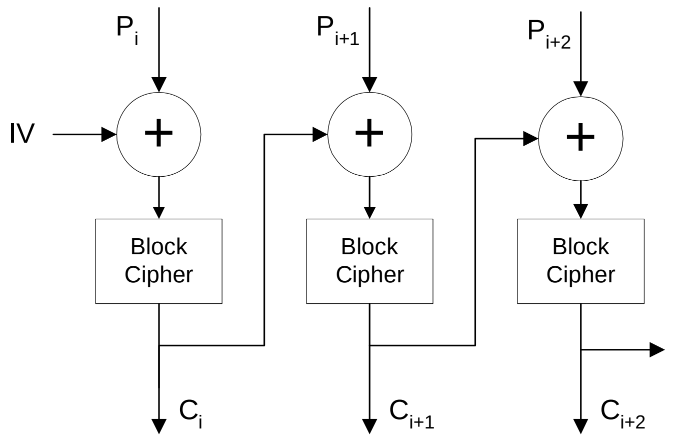
Another option for the encryption of RTP packets for IPCablecom, is 3DES-EDE-CBC. The CBC mode of 3DES-EDE is shown in a figure below, where 3DES-EDE is executed within the block called "block cipher." Inside the block, 3DES-EDE is performed as shown in a figure below using a 128-bit key. K1 is the first 8-bytes of the key, and K2 represents the second 8-bytes of key; and K3=K1.

9.3 Block Termination

If block ciphers are supported, a short block (n bits < block size depending on the cipher algorithms) must be terminated by residual block termination as shown in the figure below. Residual block termination (RBT) is executed as follows:

Given a final block having n bits, where n is less than block size, the n bits are padded up to a block by appending (block size - n) bites of arbitrary value to the right of the n-bits. The resulting block is encrypted using B-bit CFB mode, with the next-to-last ciphertext block serving as the initialization vector for the CFB operation (see [43], B. Schneier's Applied Cryptography). Here, B stands for the cipher-specific block size. 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 the cipher block size, the procedure is the same as for a short final block, with the provided initialization vector serving as the initialization vector for the operation. Residual block termination is illustrated in the figure below for both encryption and decryption operations.

CBC Encryption Architecture



CBC Decryption Architecture

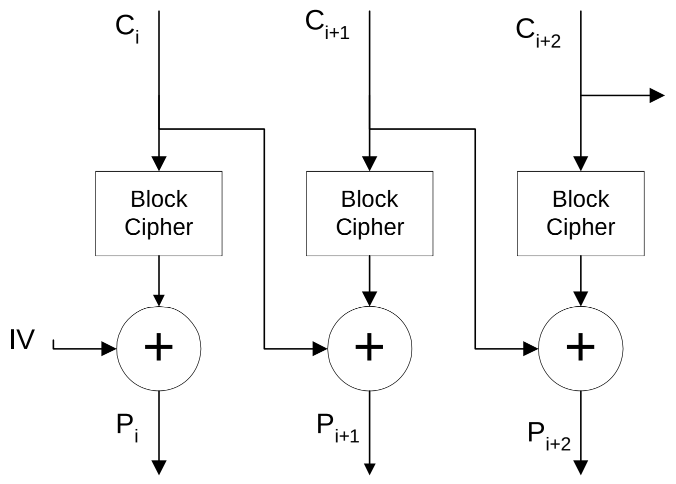


Figure 24: CBC Mode

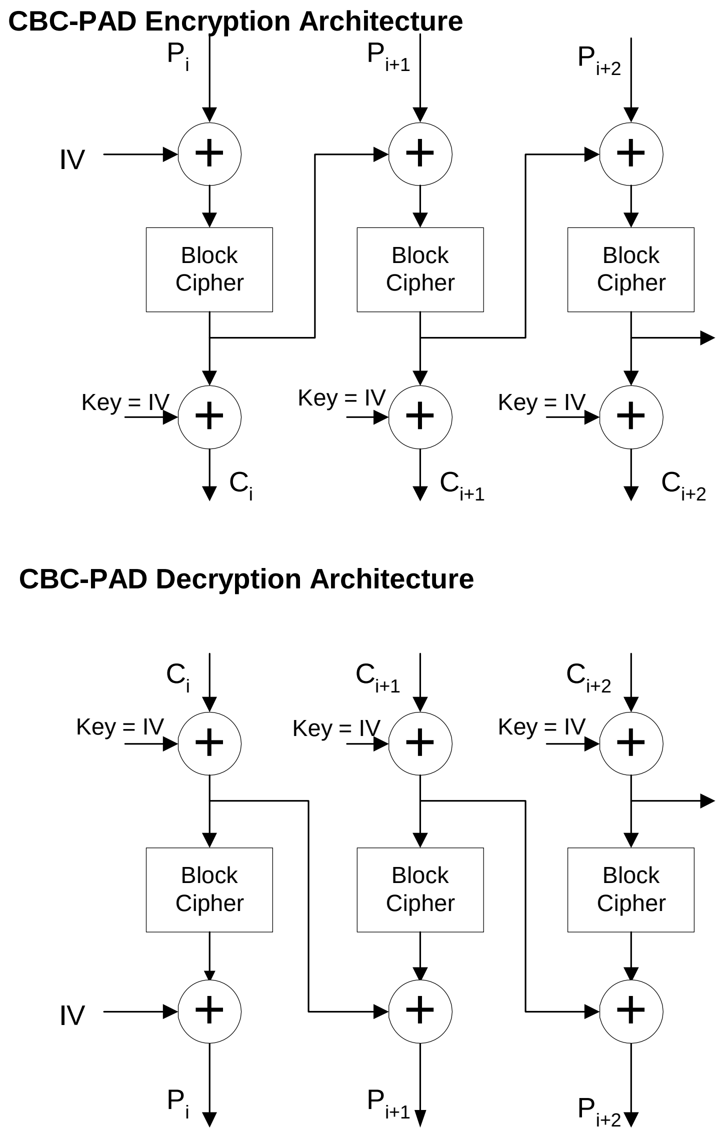


Figure 25: CBC Pad Mode

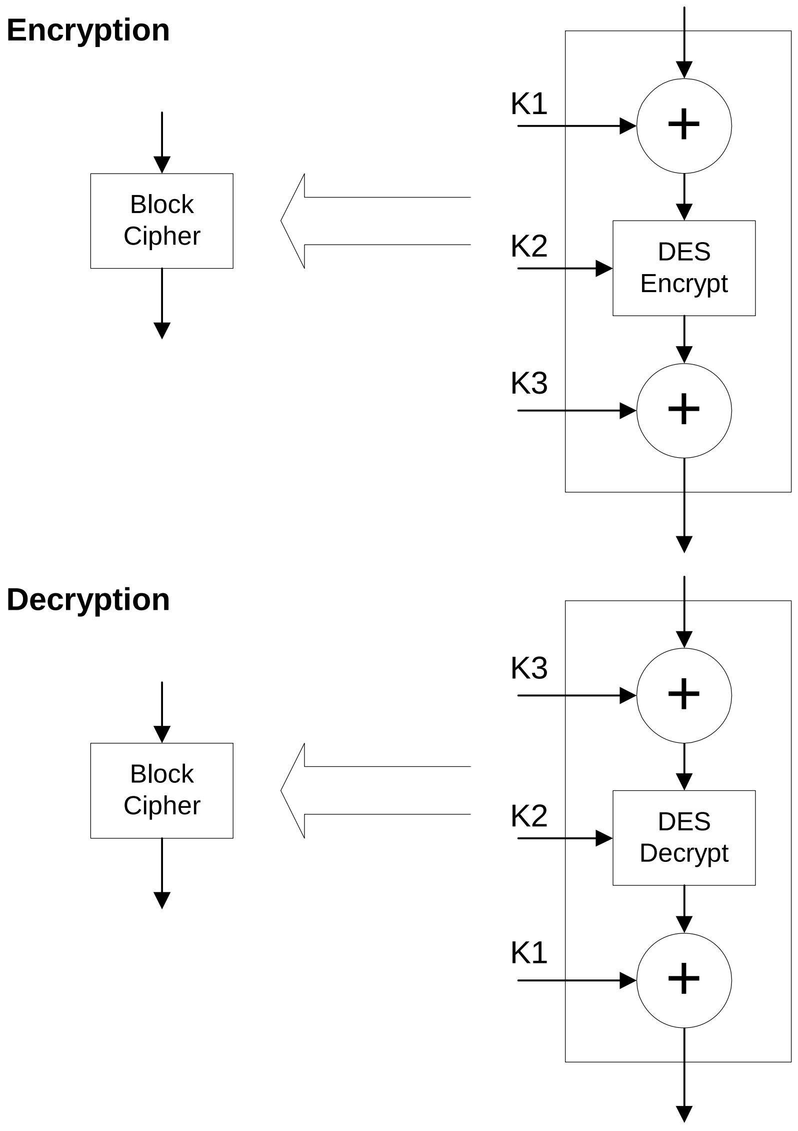


Figure 26: DESX-XEX as Block Cipher

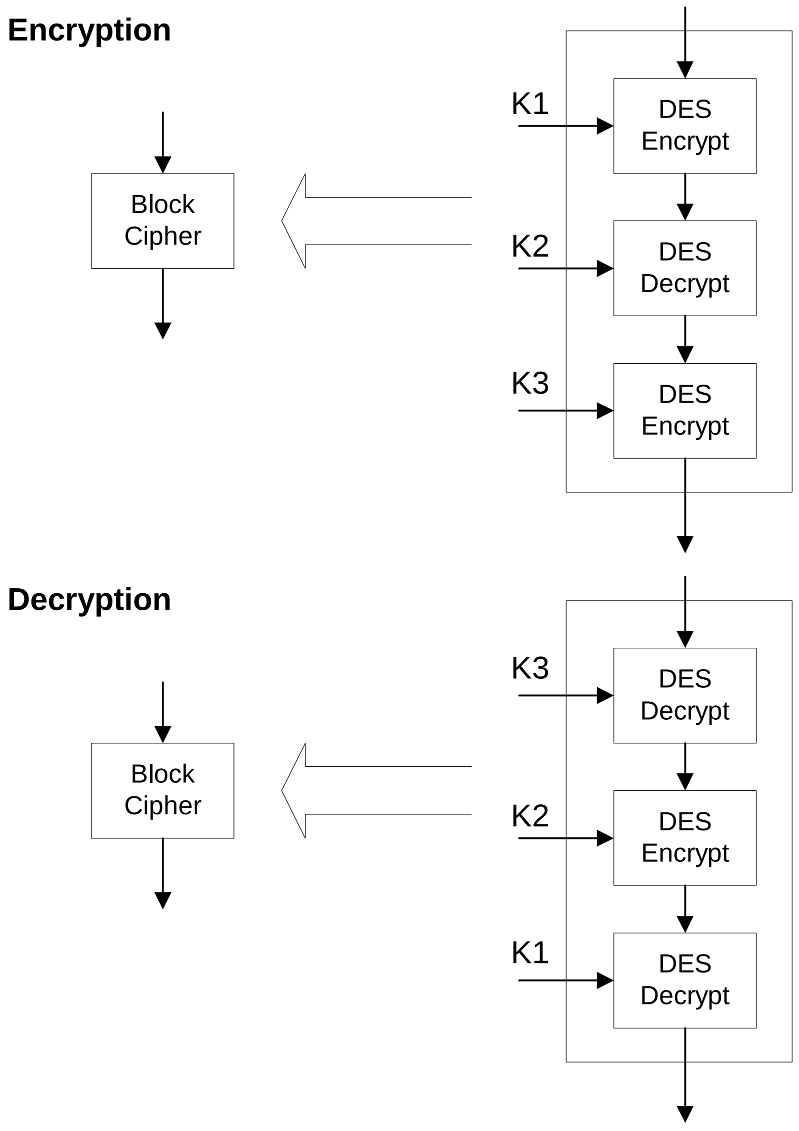


Figure 27: 3DES-EDE as Block Cipher

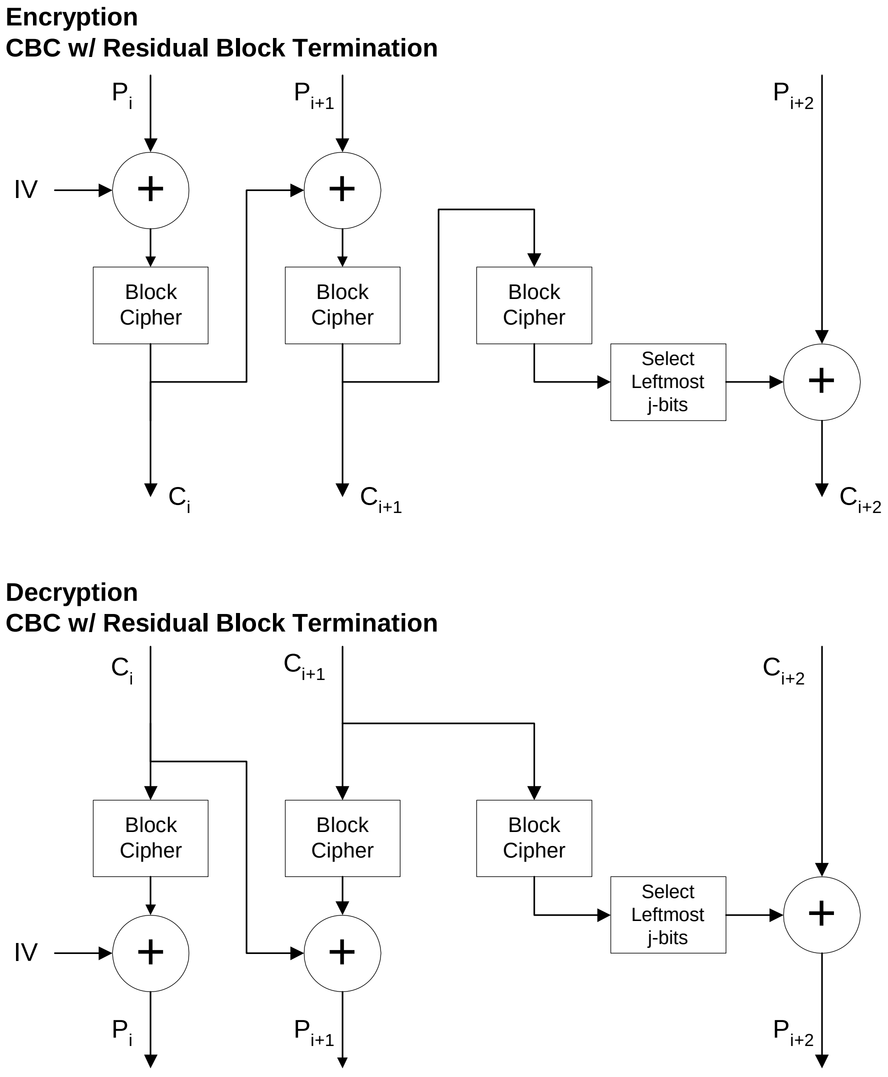


Figure 28: CBC with Residual Block Termination

9.4 RSA Signature

All public key signatures for IPCablecom must be generated and verified using the RSA signature algorithm described in [16]. The format for all IPCablecom signatures must be compliant with the Cryptographic Message Syntax [12].

9.5 HMAC-SHA1

The keyed hash employed by the HMAC-Digest Attribute must use the HMAC message authentication method [11] with the SHA-1 hash algorithm [15].

9.6 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 [17]:

```
A() is defined as: A(0) = seed
                   A(i) = HMAC_SHA-1(S, A(i-1))
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.
```

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.

9.7 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 7.6.2, 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.

9.7.1 The MMH Function

The Multilinear Modular Hash (MMH) Function described below is a variant of the MMH Function described in [18]. Some of the computations described below use signed arithmetic whereas the computations in [18] 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 [18] 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[ω,*s*,*t*] specifies the hash function with word size ω, *s* input words and *t* output words. For IPCablecom the word size is fixed to 16 bits: $\omega = 16$. 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.

9.7.1.1 MMH[16,s,1]

For the remainder of this clause 9.7, 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 9.7.2. The key is denoted by k and the ith word of the key by ki: k = k1, k2,…, ks. Likewise the input message is denoted by m and the ith word of the input message by mi: m = m1, m2,…, ms.

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,…,0,…,(*n*/2)-1}. For example, $S_{2^{16}} = \{-2^{15},\dots,0,\dots,2^{15}-1\}$ is the set of signed 16 bit integers.

For any integer *z*, *z* smod *n* is the unique element ω of S_n such that $z \equiv \omega \pmod n$. 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, …, *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 $H_1(k,m) = \sum_{i=1}^{s} k_i \cdot m_i \text{ smod } 2^{32}$.

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.

Step1. $H_1(k,m)$ 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 $S_{2^{32}}$.

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 $(x + y) \text{ smod } 2^{32} = (x \text{ smod } 2^{32} + y \text{ smod } 2^{32}) \text{ smod } 2^{32}$.

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.

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 $b \in \{0,\ldots,2^{16} - 1\}$. 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 range of the quantity $b - a + 2^{16}$ is given by:

$$1 \le b - a + 2^{16} \le 2^{17} - 2^{15} - 1 \le 2p - 1$$

Therefore, if $b - a + 2^{16} < p$ then $x \bmod p = b - a + 2^{16}$. If $b - a + 2^{16} \ge p$ then $x \bmod p = b - a + 2^{16} - p$.

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.

9.7.1.2 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, $H'(k,m) = H(k^{(1)},m) \circ H(k^{(2)},m)$.

9.7.2 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: $\omega = 16$. 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}.

9.7.2.1 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 7.6.2.1.2.2.3.

9.7.2.2 Handling Variable-Size Data

In order to handle data of all possible sizes up to a maximum value, the following rules must be followed for computing an MMH function:

- If the data is not a multiple of the word size, pad the data up to a multiple of the word size (16-bits) with zero-bytes. In other words, if the length of message *m* is not a multiple of word size w, but rather of length b octets, $b = n \times w + r$ with $n \geq 0$ and $0 < r < w$, then pad message m at the end with w-r zero-bytes before passing it as the input to *M*.
- It the key is larger than what is needed for a particular message, truncate the key. In other words, if a message *m* is not of length *s* words, but rather of length $v < s$ words, then truncate the value of the key k to $v + t - 1$ words before it is used to calculate the MMH hash. (For MMH hash with 1 word output, t = 1 and k is truncated to *v* words. For 2 word output, t = 2 and k is truncated to $v + 1$ words.)

9.8 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 [44] gives some suggestions; other possibilities include use of a per-MTA secret installed at manufacture time and used in the random number generation process.

10 Physical Security

10.1 Protection for MTA Key Storage

An MTA must 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 140-2, Security Requirements for Cryptographic Modules, standard (see [46]). An MTA should, at a minimum meet FIPS PUBS 140-2 Security Level 1 requirements [46].

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 networks.

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 MTA's, 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) An 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 DOCSIS® 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 CMTS 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 140-2 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 DOCSIS® 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 DOCSIS® 1.1 / BPI+ multicast implementation, where passive clones would be restricted to a single downstream CMTS segment.) The snooping of the point-to-point data is limited to the DOCSIS® 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.

10.2 MTA Key Encapsulation

As stated in the previous clause, FIPS PUB 140-2 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.

One of the (many) requirements of FIPS PUB 140-2 Security Level 3 is that "the entry or output of plaintext Critical Security Parameters (CSPs) be performed using ports that are physically separated from other ports, or interfaces that are logically separated using a trusted path from other interfaces. Plaintext CSPs may be entered into or output from the cryptographic module in encrypted form (in which case they may travel through enclosing or intervening systems)". As also mentioned in the previous clause, the IPCablecom security specification is not requiring compliance with any of the FIPS PUB 140-2 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 Cable Modem (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.

11 Secure Software Download

IPCablecom 1.5 includes only Embedded MTAs. E-MTAs are embedded with DOCSIS® 1.1 cable modems (including BPI+). E-MTAs must have their software upgraded by the Cable Modem according to the DOCSIS® 1.1 requirements as specified in [8] and [9].

Annex A (informative): IPCablecom Admin 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.

A.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 be a provisioned parameter that can be used in one the following ways:

In the case of manual key changes, an administrator is prompted or reminded to manually change a CMS service key.

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 90 days (but at least the maximum ticket lifetime). This may further reduce the risk of key compromise.

Annex B (normative): Kerberos Network Authentication Service

See RFC 4120 [14].

Annex C (informative): PKINIT Specification

See RFC 4556 [49].

Annex D (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 PacketCable 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 E (normative): Oakley Groups

PKINIT states that DH parameters should be taken from the first or second Oakley groups as defined in [26]. Additionally, the present document requires that DH groups are used exactly as defined in [26].

[26] defines several so-called "Oakley groups." Only the first two are relevant to the present document. [26] requires implementations to support the first group, and recommends that they support the second. This annex is included because [26] does not give values of q (the p-1 factor) for the groups, and these are necessary in order to encode the dhpublicnumber type used in the subjectPublicKeyInfo data structure in PKINIT.

The first two Oakley groups are defined as follows:

First Oakley Group:

Prime (p):

```
FFFFFFFF FFFFFFFF C90FDAA2 2168C234 C4C6628B 80DC1CD1
29024E08 8A67CC74 020BBEA6 3B139B22 514A0879 8E3404DD
EF9519B3 CD3A431B 302B0A6D F25F1437 4FE1356D 6D51C245
E485B576 625E7EC6 F44C42E9 A63A3620 FFFFFFFF FFFFFFFF
```

Generator (g or b):

2.

Factor (q):

```
7FFFFFFF FFFFFFFF E487ED51 10B4611A 62633145 C06E0E68
94812704 4533E63A 0105DF53 1D89CD91 28A5043C C71A026E
F7CA8CD9 E69D218D 98158536 F92F8A1B A7F09AB6 B6A8E122
F242DABB 312F3F63 7A262174 D31D1B10 7FFFFFFF FFFFFFFF
```

Second Oakley Group:

Prime (p):

```
FFFFFFFF FFFFFFFF C90FDAA2 2168C234 C4C6628B 80DC1CD1
29024E08 8A67CC74 020BBEA6 3B139B22 514A0879 8E3404DD
EF9519B3 CD3A431B 302B0A6D F25F1437 4FE1356D 6D51C245
E485B576 625E7EC6 F44C42E9 A637ED6B 0BFF5CB6 F406B7ED
EE386BFB 5A899FA5 AE9F2411 7C4B1FE6 49286651 ECE65381
FFFFFFFF FFFFFFFF
```

Generator (g or b):

2.

Factor (q):

```
7FFFFFFF FFFFFFFF E487ED51 10B4611A 62633145 C06E0E68
94812704 4533E63A 0105DF53 1D89CD91 28A5043C C71A026E
F7CA8CD9 E69D218D 98158536 F92F8A1B A7F09AB6 B6A8E122
F242DABB 312F3F63 7A262174 D31BF6B5 85FFAE5B 7A035BF6
F71C35FD AD44CFD2 D74F9208 BE258FF3 24943328 F67329C0
FFFFFFFF FFFFFFFF
```

Annex F (informative): Bibliography

IETF RFC 2327: "SDP: Session Description Protocol", April 1998.

History

Document history		
V1.1.1	October 2011	Publication