Data-Over-Cable Service Interface Specifications

Baseline Privacy Plus Interface Specification

SP-BPI+-I09-020830

ISSUED
SPECIFICATION

Document Status Sheet

Document Control Number:	SP-BPI+-I09-020830			
Reference:	Baseline Privacy Plus Interface Specification			
Revision History:	I01 - First Issued Release, March 16, 1999 I02 - Second Issued Release, July 31, 1999 I03 - Third Issued Release, November 5, 1999 I04 - Fourth Issued Release, April 7, 2000 I05 - Fifth Issued Release, July 14, 2000 I06 - Sixth Issued Release, December 15, 2000 I07 - Seventh Issued Release, August 29, 2001 I08 - Eighth Issued Release, March 1, 2002 I09 - Ninth Issued Release, August 30, 2002			
Date:	August 30, 2002			
Status Code:	~~Work in Process~~	~~Draft~~	Issued	~~Closed~~
Distribution Restrictions:	~~CableLabs Only~~	~~CableLabs Reviewers~~	~~CableLabs Vendors~~	Public

Key to Document Status Codes

Work in Process An incomplete document, designed to guide discussion and generate feedback, that may include several alternative requirements for consideration.

Draft A document in specification format considered largely complete, but lacking review by cable industry and vendors. Drafts are susceptible to substantial change during the review process.

Issued A stable document, which has undergone rigorous member and vendor review and is suitable for product design and development, cross-vendor interoperability, and for certification testing.

Closed A static document, reviewed, tested, validated, and closed to further engineering change requests to the specification through CableLabs.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

This page intentionally left blank.

1 Scope and Purpose

1.1 Scope

The content of this BPI+ specification is to describe MAC layer security services for DOCSIS CMTS - CM communications. Its security enhancements are to afford:

- protection of cable modem users via data privacy across the network
- provide MSOs with service protection; i.e., prevent unauthorized users from gaining access to the network's RF MAC services

BPI+ provides a level of data privacy across the shared medium cable network equal to or better than that provided by dedicated line network access services (analog modems or digital subscriber lines).

The protected RF MAC data communications services fall into three categories:

- best-effort, high-speed, IP data services
- QoS (e.g., constant bit rate) data services
- IP multicast group services

The earlier BPI specification [DOCSIS2] had "weak" service protection because the underlying key management protocol did not authenticate CMs. BPI+ strengthens this service protection by adding digital-certificate based CM authentication to its key exchange protocol.

1.2 Requirements

Throughout this document, the words that are used to define the significance of particular requirements are capitalized. These words are:

"MUST"	This word or the adjective "REQUIRED" means that the item is an absolute requirement of this specification.
"MUST NOT"	This phrase means that the item is an absolute prohibition of this specification.
"SHOULD"	This word or the adjective "RECOMMENDED" means that there may exist valid reasons in particular circumstances to ignore this item, but the full implications should be understood and the case carefully weighed before choosing a different course.
"SHOULD NOT"	This phrase means that there may exist valid reasons in particular circumstances when the listed behavior is acceptable or even useful, but the full implications should be understood and the case carefully weighed before implementing any behavior described with this label.
"MAY"	This word or the adjective "OPTIONAL" means that this item is truly optional. One vendor may choose to include the item because a particular marketplace requires it or because it enhances the product, for example; another vendor may omit the same item.

Other text is descriptive or explanatory.

1.3 Background

1.3.1 Service Goals

Cable operators are interested in deploying high-speed packet-based communications systems on cable television systems that are capable of supporting a wide variety of services. Services under consideration by cable operators include high-speed Internet access, packet telephony service, video conferencing service, T1/frame relay equivalent service and many others. To this end, CableLabs' member companies have decided to prepare a series of interface specifications that will permit the early definition, design, development and deployment of data-over-cable systems on a uniform, consistent, open, non-proprietary, multi-vendor interoperable basis.

The intended service will allow transparent bi-directional transfer of Internet Protocol (IP) traffic, between the cable system headend and customer locations, over an all-coaxial or hybrid fiber/coax (HFC) cable television network. This is shown in simplified form in Figure 1-1.

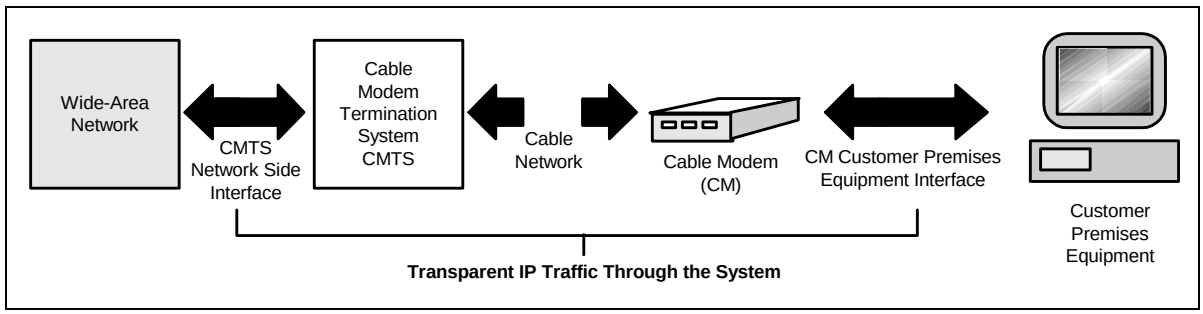


Figure 1-1. Transparent IP Traffic Through the Data-Over-Cable System

The transmission path over the cable system is realized at the headend by a CMTS, and at each customer location by a CM. At the headend (or hub), the interface to the data-over-cable system is called the Cable Modem Termination System - Network-Side Interface (CMTS-NSI)) and is specified in [DOCSIS3]. At the customer locations, the interface is called the cable-modem-to-customer-premise-equipment interface (CMCI) and is specified in [DOCSIS4]. The intent is for the cable operators to transparently transfer IP traffic between these interfaces, including but not limited to datagrams, DHCP, ICMP, and IP Group addressing (broadcast and multicast).

1.3.2 Reference Architecture

The reference architecture for the data-over-cable services and interfaces is shown in Figure 1-2.

1.3.3 Categories of Interface Specification[1]

The basic reference architecture of Figure 1-2 identifies a number of interfaces.

1. References updated per BPI-N-02098 by RKV on 8/26/02.

Data Interfaces - These are the CMCI [DOCSIS4] and CMTS-NSI [DOCSIS3], corresponding respectively to the cable modem to customer-premises-equipment (CPE) interface (for example between the customer's computer and the cable modem), and the cable modem termination system network side interface between the cable modem termination system and the data network.

Operations Support System Interfaces - OSSI - These are network element management layer interfaces between the network elements and higher level OSSs (operations support systems) which support the basic business processes and are documented in [DOCSIS5] or [DOCSIS10].

Telephony Return Interface - CMTRI - This is the interface between the cable modem and a telephone return path, for use in cases where the return path is not provided or not available via the cable network and is documented in [DOCSIS6].

RF Interfaces -

- Between the cable modem and the cable network, [DOCSIS1] or [DOCSIS9].
- Between the CMTS and the cable network, in the downstream direction (traffic toward the customer) [DOCSIS1] or [DOCSIS9].
- Between the CMTS and the cable network, in the upstream direction (traffic from the customer) [DOCSIS1] or [DOCSIS9].

Security requirements -

- The Data-Over-Cable Baseline Privacy interface specification is this document.

1.3.3.1 Data-Over-Cable Interface Documents

A list of the documents in the Data-Over-Cable Interface Specifications family is provided below. For update, please refer to URL http://www.cablemodem.com.

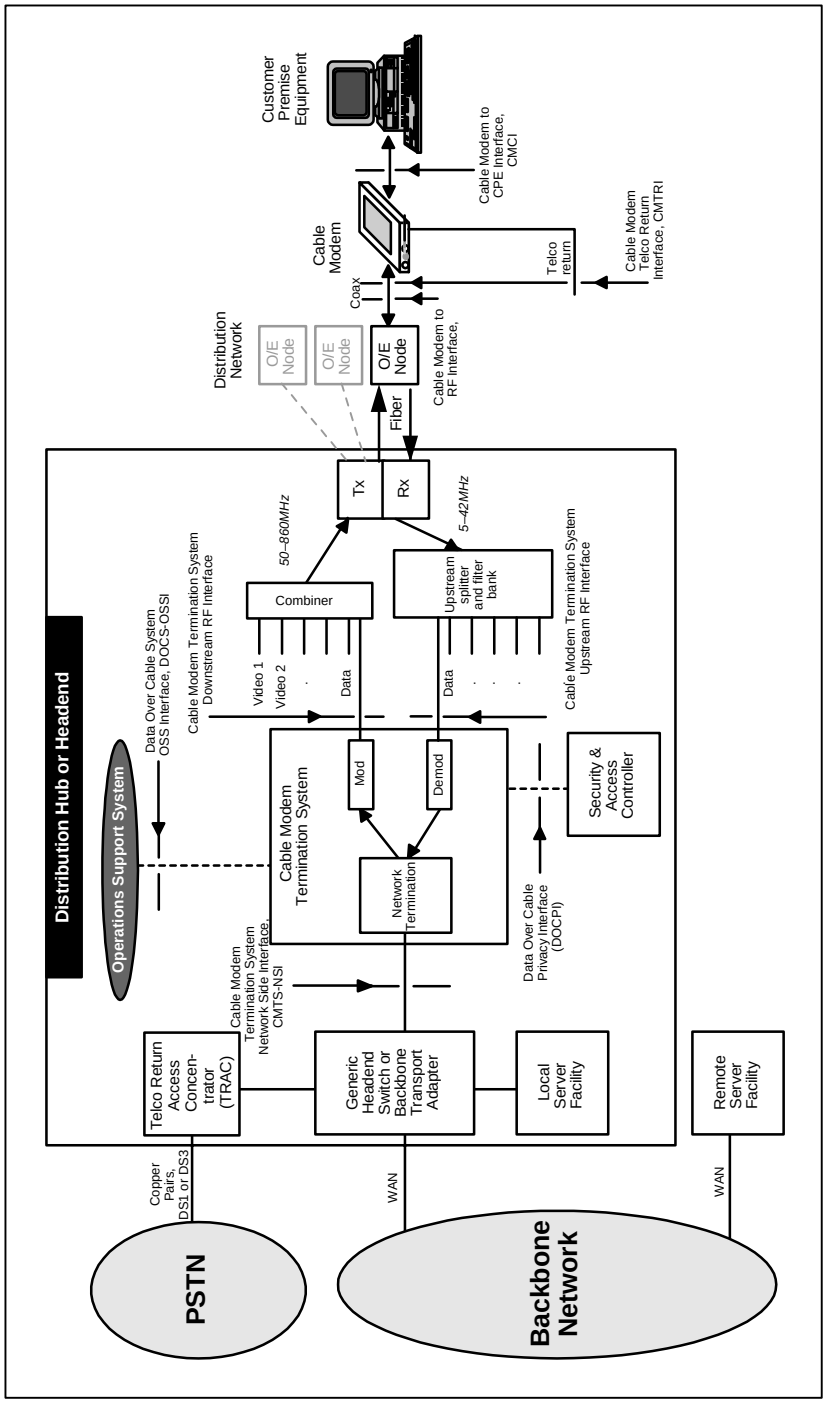


Figure 1-2. Data-over-Cable Reference Architecture

Designation	Title
SP-BPI	Baseline Privacy Interface Specification
SP-CMCI	Cable Modem-to-Customer Premises Equipment Interface Specification
SP-CMTRI	Cable Modem Telephony Return Interface Specification
SP-CMTS-NSI	Cable Modem Termination System Network Side Interface Specification
SP-OSSI	Operations Support System Interface Specification
SP-OSSI-RF	Operations Support System Interface Radio Frequency MIB
SP-OSSI-BPI	Operations Support System Interface Baseline Privacy System MIB
SP-OSSI-TR	Operations Support Systems Interface Telephony Return MIB
SP-RFI	Radio Frequency Interface Specification
TR-OSSF	Operations Support System Framework

Key to Designation:

SP Specification
TR Technical Report (provides a context for understanding and applying the specification.)

1.3.4 Statement of Compatibility

This document specifies an interface, commonly referred to as BPI+, which is an extension of the interface specified in [DOCSIS2], commonly referred to as BPI. These extensions are entirely backwards and forwards compatible with the previous specification. BPI+ devices will interoperate in networks with CMs or CMTSs implementing BPI, and vice versa.

Refer to Appendix C for further compatibility information.

This page intentionally left blank.

2 Baseline Privacy Plus Overview

Baseline Privacy Plus (BPI+) provides cable modem users with data privacy across the cable network. It does this by encrypting traffic flows between CM and CMTS.

In addition, BPI+ provides cable operators with strong protection from theft of service. The protected DOCSIS MAC data communications services fall into three categories:

- best-effort, high-speed, IP data services
- QoS (e.g., constant bit rate) data services
- IP multicast group services

Under BPI+, the CMTS protects against unauthorized access to these data transport services by enforcing encryption of the associated traffic flows across the cable network. BPI+ employs an authenticated client/server key management protocol in which the CMTS, the server, controls distribution of keying material to client CMs.

The original BPI specification had “weak” service protection because the underlying key management protocol did not authenticate CMs. BPI+ strengthens this service protection by adding digital-certificate based CM authentication to its key management protocol.

2.1 Architectural Overview

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”) providing 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.

2.1.1 Packet Data Encryption

BPI+ encryption services are defined as a set of extended services within the DOCSIS MAC sublayer. Packet Header information specific to BPI+ is placed in a Baseline Privacy Extended Header element within the MAC Extended Header.

At the time of this specification’s release, BPI+ supports a single packet date encryption algorithm: the Cipher Block Chaining (CBC) mode of the US Data Encryption Standard (DES) algorithm [FIPS-46-1] [FIPS-81]. BPI+ does not pair DES CBC with any packet data authentication algorithm. Additional data encryption algorithms may be supported in future enhancements to the BPI+ protocol specification, and these algorithms may be paired with data authentication algorithms.

BPI+ encrypts a DOCSIS MAC Frame's packet data; the DOCSIS MAC Frame's Header is not encrypted. DOCSIS MAC management messages MUST be sent in the clear to facilitate registration, ranging, and normal operation of the DOCSIS MAC sublayer.[2]

Section 3 specifies the format of DOCSIS MAC Frames carrying encrypted packet data payloads.

2.1.2 Key Management Protocol

CMs use the Baseline Privacy Key Management protocol to obtain authorization and traffic keying material from the CMTS, and to support periodic reauthorization and key refresh. The key management protocol uses X.509 digital certificates [ITU1], RSA [RSA, RSA1, RSA3] (a public-key encryption algorithm) and two-key triple DES to secure key exchanges between CM and CMTS.

The Baseline Privacy Key Management protocol adheres to a client/server model, where the CM, a BPKM "client", requests keying material, and the CMTS, a BPKM "server", responds to those requests, ensuring individual CM clients only receive keying material they are authorized for. The BPKM protocol uses DOCSIS MAC management messaging.

BPI+ uses public-key cryptography to establish a shared secret (i.e., an Authorization Key) between CM and CMTS. The shared secret is then used to secure subsequent BPKM exchanges of traffic encryption keys. This two-tiered mechanism for key distribution permits refreshing of traffic encryption keys without incurring the overhead of computation-intensive public-key operations.

A CMTS authenticates a client CM during the initial authorization exchange. Each CM carries a unique X.509 digital certificate issued by the CM's manufacturer. The digital certificate contains the CM's Public Key along with other identifying information; i.e., CM MAC address, manufacturer ID and serial number. When requesting an Authorization Key, a CM presents it's digital certificate to a CMTS. The CMTS verifies the digital certificate, and then uses the verified Public Key to encrypt an Authorization Key, which the CMTS then sends back to the requesting CM.

The CMTS associates a cable modem's authenticated identity to a paying subscriber, and hence to the data services that subscriber is authorized to access. Thus, with the Authorization Key exchange, the CMTS establishes an authenticated identity of a client CM, and the services (i.e., specific traffic encryption keys) the CM is authorized to access.

Since the CMTS authenticates CMs, it can protect against an attacker employing a *cloned* modem, masquerading as a legitimate subscriber's modem. The use of the X.509 certificates prevents cloned modems from passing fake credentials onto a CMTS.

CMs MUST have factory-installed RSA private/public key pairs or provide an internal algorithm to generate such key pairs dynamically. If a CM relies on an internal algorithm to generate its RSA key pair, the CM MUST generate the key pair prior to its first Baseline Privacy initialization, described in Section 2.2.1. CMs with factory-installed RSA key pairs MUST also have factory-installed X.509 certificates. Cable modems that rely on internal algorithms to generate an RSA key pair MUST support a mechanism for installing a manufacturer-issued X.509 certificate following key generation.

---

2. The DOCSIS MAC headers of Packet Data PDUs and non-BPI+ DOCSIS MAC management messages MAY be encrypted when part of a fragmented concatenated packet [DOCSIS1] or [DOCSIS9].

The BPKM protocol is defined in detail in Section 4.

2.1.3 BPI+ Security Associations

A BPI+ *Security Association* (SA) is the set of security information a CMTS and one or more of its client CMs share in order to support secure communications across the cable network. BPI+ defines three types of Security Associations: *Primary*, *Static*, and *Dynamic*. A Primary Security Association is tied to a single CM, and is established when that CM completes DOCSIS MAC registration. Static Security Associations are provisioned within the CMTS. Dynamic Security Associations are established and eliminated, on the fly, in response to the initiation and termination of specific (downstream) traffic flows. Both Static and Dynamic SAs can by shared by multiple CMs.

A Security Association's shared information includes traffic encryption keys and CBC initialization vectors. In order to support, in future BPI+ enhancements, alternative data encryption and data authentication algorithms, BPI+ Security Association parameters include a cryptographic suite identifier, indicating a the particular pairing of packet data encryption and packet data authentication algorithms employed by the security association. At the time of release of this specification, 56-bit DES and 40-bit DES are the only packet data encryption algorithms supported, and neither are paired with a packet data authentication algorithm.[3]

BPI+ identifies Security Associations with a 14-bit *Security Association Identifier (SAID)*.

Each (BPI+ enabled) CM establishes an exclusive Primary Security Association with its CMTS. All of a CM's upstream traffic MUST be encrypted under the CM's exclusive, Primary Security Association. The SAID corresponding to a CM's Primary SA MUST be equal to the CM's Primary DOCSIS 1.1 Service ID (SID) [DOSCIS1]. On the other hand, while typically all downstream unicast traffic directed at CPE device(s) behind the CM, are encrypted under the CM's exclusive Primary Security Association, selected downstream unicast traffic flows can be encrypted under Static or Dynamic SAs. That is, downstream traffic MAY be encrypted under any of the three types of SAs. A downstream IP multicast data packet, however, is typically intended for multiple CMs and hence is more likely to be encrypted under Static or Dynamic SAs, which multiple CMs can access, as opposed to a Primary SA, which is restricted to a single CM.

Using the BPKM protocol, a CM requests from its CMTS a SA's keying material. The CMTS ensures that each client CM only has access to the Security Associations it is authorized to access.

A SA's keying material (e.g., DES key and CBC Initialization Vector) has a limited lifetime. When the CMTS delivers SA keying material to a CM, it also provides the CM with that material's remaining lifetime. It is the responsibility of the CM to request new keying material from the CMTS before the set of keying material that the CM currently holds expires at the CMTS. The BPKM protocol specifies how CM and CMTS maintain key synchronization.

3. BPI+ encrypts a Packet PDU's Ethernet/802.3 CRC. While this provides some degree of data authentication, it does not provide cryptographically secure data authentication.

2.1.4 QoS SIDs and BPI+ SAIDs

The BPI+ Extended Header Element in downstream DOCSIS MAC frames contains the BPI+ SAID under which the downstream frame is encrypted. If the downstream frame is a unicast packet addressed to a CPE device behind a particular CM, the frame will typically be encrypted under the CM's Primary SA, in which case the SAID will be equal to the target CM's Primary SID. If the downstream frame is a multicast packet intended for receipt by multiple CMs, the extended header element will contain the Static or Dynamic SAID mapped to that multicast group. The SAID (Primary, Static or Dynamic), in combination with other data fields in the downstream extended header element, identifies to a receiving modem the particular set of keying material required to decrypt the DOCSIS MAC frame's encrypted Packet Data field.

Since all of a CM's upstream traffic is encrypted under its unique Primary SA, upstream DOCSIS MAC Frames, unlike downstream DOCSIS MAC Frames, need not carry a BPI+ SAID in their extended headers; instead, the Baseline Privacy EH element MAY contain any valid QoS SID assigned to the CM.

The Baseline Privacy extended header element serves multiple purposes in upstream DOCSIS Packet Data PDU MAC Frames. In addition to identifying the particular set of keying material used to encrypt a Frame's packet data, it also provides a mechanism for issuing piggybacked bandwidth requests, and it can carry fragmentation control data. These later two functions are tied to a particular QoS SID; for this reason, upstream Baseline Privacy Extended Header Elements contain a QoS SID rather than a BPI+ Primary SAID, which can be inferred from the QoS SID.

2.2 Operational Overview

2.2.1 Cable Modem Initialization

[DOCSIS1] divides cable modem initialization into the following sequence of tasks:

- scan for downstream channel and establish synchronization with the CMTS
- obtain transmit parameters
- perform ranging
- establish IP connectivity (DHCP)
- establish time of day
- transfer operational parameters (download parameter file via TFTP)
- CMTS Registration

Baseline Privacy establishment follows CMTS registration.

If a CM is to run Baseline Privacy, the Privacy Enable setting (type 29) in the DOCSIS 1.1 or 2.0 style configuration file MUST be explicitly/implicitly set to enable, regardless of the presence of the Baseline Privacy Configuration Settings (type 17). In other words, Baseline Privacy Configuration Settings do not need to be present in the configuration file in order to run Baseline Privacy. These additional configuration settings are defined in Appendix A.[4]

4. modified per bpi-n-02119, 07/30/02, ab

Upon completing CMTS registration, the CMTS will have assigned one or more static Service IDs (SIDs) to the registering CM that match the CM's static class-of-service provisioning. The first static SID assigned during the registration process is the Primary SID, and this SID will also serve as the CM's BPI+ Primary SAID. If a CM is configured to run Baseline Privacy, CMTS registration is immediately followed by initialization of the CM's Baseline Privacy security functions.

Baseline Privacy initialization begins with the CM sending the CMTS an Authorization Request, containing:

- data identifying the CM (e.g., MAC address)
- the CM's RSA public key
- an X.509 certificate verifying the binding between the CM's identifying data and the CM's public key
- a list of the CM's security capabilities (i.e., the particular pairings of encryption and authentication algorithms the CM supports)
- the CM's Primary SAID (i.e., the Primary SID)

If the CMTS determines the requesting CM is authorized for the Authorization Request's Primary SAID, the CMTS responds with an Authorization Reply containing an Authorization Key, from which CM and CMTS derive the keys needed to secure a CM's subsequent requests for traffic encryption keys and the CMTS's responses to these requests. The CMTS encrypts the Authorization Key with the receiving cable modem's public key.

The Authorization Reply also contains a list of security association descriptors, identifying the primary and static SAs the requesting CM is authorized to access. Each SA descriptor consists of a collection of SA parameters, including the SA's SAID, type and cryptographic. The list contains at least one entry: a descriptor describing the CM's primary security association. Additional entries are optional, and would describe any static SAs the CM was provisioned to access.

After successfully completing authentication and authorization with the CMTS, the cable modem sends key requests to the CMTS, requesting traffic encryption keys to use with each of its SAIDs. A CM's traffic key requests are authenticated using a keyed hash (the HMAC algorithm [RFC2104]); the Message Authentication Key is derived from the Authorization Key obtained during the earlier authorization exchange. The CMTS responds with key replies, containing the Traffic Encryption Keys (TEKs); TEKs are triple DES encrypted with a key encryption key derived from the Authorization Key. Like the Key Requests, Key Replies are authenticated with a keyed hash, where the Message Authentication Key is derived from the Authorization Key.

2.2.2 Cable Modem Key Update Mechanism

The traffic encryption keys which the CMTS provides to client CMs have a limited lifetime. The CMTS delivers a key's remaining lifetime, along with the key value, in the key replies it sends to its client CMs. The CMTS controls which keys are current by flushing expired keys and generating new keys. It is the responsibility of individual cable modems to insure the keys they are using match those the CMTS is using. Cable modems do this by tracking when a particular SAID's key is scheduled to expire and issuing a new key request for the latest key prior to that expiration time.

In addition, cable modems are required to periodically reauthorize with the CMTS; as is the case with Traffic Encryption Keys, an Authorization Key has a finite lifetime which the CMTS provides the CM along with the key value. It is the responsibility of each cable modem to reauthorize and obtain a fresh Authorization Key (and an up-to-date list of SA descriptors) before the CMTS expires the CM's current Authorization Key.

Baseline Privacy initialization and key update is implemented within the Baseline Privacy Key Management protocol, defined in detail in Section 4.

3 DOCSIS MAC Frame Formats

When operating with BPI+ enabled, CM and CMTS encrypt the Data PDU regions of particular DOCSIS MAC Frames they transmit onto the cable network. BPI+ encryption applies to two specific types of DOCSIS MAC frames:

- variable-length packet data PDU MAC frames
- fragmentation MAC frames

In each of the two cases, a Baseline Privacy Extended Header Element in the DOCSIS MAC Header identifies the Security Association and accompanying keying material used to encrypt the Data PDU.

3.1 Variable-Length Packet Data PDU MAC Frame Format

Figure 3-1 depicts the format of a DOCSIS variable-length Packet Data PDU with a Privacy Extended Header (EH) Element and encrypted Packet PDU payload.

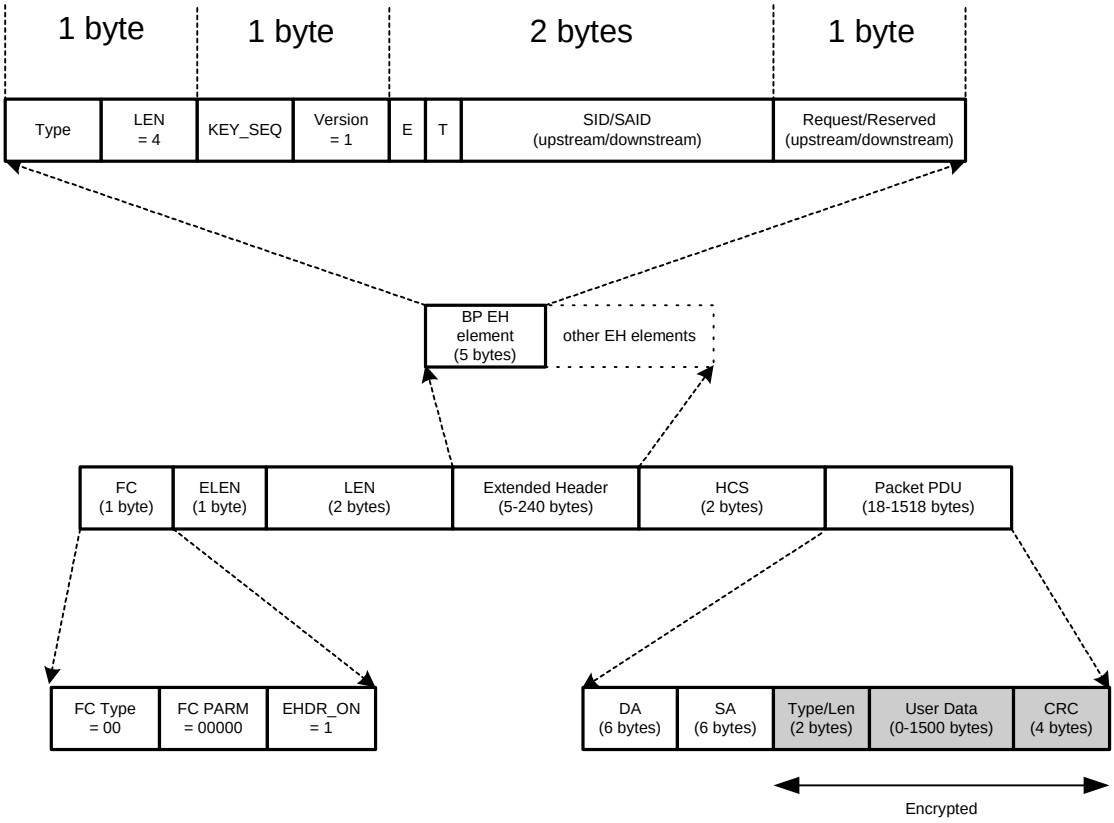


Figure 3-1. Format of DOCSIS Variable-length Packet Data PDU with Privacy EH Element

The first 12 octets of the Packet PDU, containing the Ethernet/802.3 destination and source addresses (DA/SA), are not encrypted. Transmitting a frame's destination and source addressing in the clear provides vendors with greater flexibility in how they integrate encryption/decryption with DOCSIS

MAC functionality; e.g., vendors have freedom to choose between filtering on DA/SA or SID first. The Packet PDU's Ethernet/802.3 CRC is encrypted.

The CMTS includes the Baseline Privacy EH element in all downstream Packet Data PDUs it encrypts under Baseline Privacy Plus. Similarly, a CM includes the Baseline Privacy EH element in all upstream Packet Data PDUs it encrypts under Baseline Privacy Plus. If there are multiple Extended Header elements present in the DOCSIS MAC Header, the Baseline Privacy Extended Header element MUST be the first.

The Privacy Extended Header element employs two EH element type values, BPI_UP and BPI_DOWN, for use with upstream and downstream Packet Data PDUs, respectively. [DOCSIS1] defines the specific EH element type values assigned to BPI_UP and BPI_DOWN.

The high-order 4 bits of a BPI+ Extended Header element's value field contains a key sequence number, KEY_SEQ. Recall that the keying material associated with a BPI+ SAID has a limited lifetime, and the CMTS periodically refreshes a SAID's keying material. The CMTS manages a 4-bit key sequence number independently for each SAID and distributes this key sequence number along with the SAID's keying material to client CMs. The CMTS increments the key sequence number with each new generation of keying material. The Privacy EH element includes this sequence number, along with the SAID, to identify the specific generation of that SAID's keying material being used to encrypt the attached Packet Data PDU. Being a 4-bit quantity, the sequence number wraps around to 0 when it reaches 15.

Comparing a received frame's key sequence number with what it believes to be the "current" key sequence number, a CM or CMTS can easily recognize a loss of key synchronization with its peer. A CM MUST maintain the two most recent generations of keying material for each BPI+ SAID. Keeping on-hand the two most recent key generations is necessary for maintaining uninterrupted service during a SAID's key transition.

The 4 bits following KEY_SEQ contain a protocol version number. This protocol version number is set to 1 in DOCSIS Variable-length Packet Data PDU MAC headers.

The next two bytes contain the 2 bits of encryption status and the 14-bit SID/SAID (SID for upstream frames, SAID for downstream frames). The ENABLE encryption status bit indicates whether encryption is enabled or disabled for that PDU. If the ENABLE bit is 0, the Packet Data PDU is not encrypted and the Baseline Privacy EH element MUST be ignored (with the exception of the optional piggybacked bandwidth request - see below). The TOGGLE bit MUST match the state of the Least Significant Bit (LSB) of KEY_SEQ, the Key Sequence Number.

The DOCSIS MAC protocol [DOCSIS1] defines a Request EH element for piggybacking a bandwidth request on a data transmission. Baseline Privacy defines an additional mechanism for piggybacking bandwidth requests: the last byte of the Baseline Privacy upstream EH element (EH element type = BPI_UP) carries an optional piggybacked bandwidth allocation request. If there is a piggybacked request, the byte represents the number of requested mini-slots. The 14-bit SID within the upstream Baseline Privacy EH element identifies the Service ID the bandwidth request applies to. If there is no piggybacked request within the Baseline Privacy EH element, the request byte is set to zero. A piggybacked request within the Baseline Privacy EH element MUST be processed regardless of the status of the ENABLE bit.

In downstream packets (extender header element type = BPI_DOWN) the fourth and final byte is reserved and set to zero.

Table 3-1. Summary of the contents of the two Baseline Privacy EH Elements.

EH_TYPE	EH_LEN	EH_VALUE
BPI__UP See [DOCSIS1]	4	KEY_SEQ (4 bits), Version (4 bits), SID (2 bytes), Request [piggyback] (1 byte) [CM --> CMTS] KEY_SEQ field (4 bits): Key sequence number Version field (4 bits) is defined as: 0x1 SID field is defined as: Bit[15]: ENABLE: 1..Encryption enabled; 0..Encryption Disabled Bit[14]: TOGGLE: 1..Odd Key; 0..Even Key Bit[13:0]: Service ID. Request field contains the number of mini-slots requested for upstream bandwidth.
BPI__DOWN See [DOCSIS1]	4	KEY_SEQ (4 bits), Version (4 bits), SID (2 bytes), Reserved (1 byte) [CMTS --> CM] KEY_SEQ field (4 bits): Key sequence number Version field (4 bits) is defined as: 0x1 SAID field is defined as: Bit[15]: ENABLE: 1..Encryption enabled; 0..Encryption Disabled Bit[14]: TOGGLE: 1..Odd Key; 0..Even Key Bit[13:0]: Security Association ID. Reserved field is set to 0.

In the case of encrypted Packet Data PDUs transmitted in an upstream data contention interval, the SID in the Baseline Privacy EH element MUST identify the QoS SID; it MUST NOT be set to the Request/Data contention interval's Multicast Service ID.

3.2 Fragmentation MAC Frame Format

In order to support fragmentation of upstream DOCSIS MAC frames, DOCSIS 1.1 has recast the Baseline Privacy EH element to carry both encryption and fragmentation control fields [DOCSIS1]. When functioning in this dual role, the upstream Baseline Privacy EH element (EH element type BPI_UP) is extended by one byte, the final byte serving as the fragmentation control field. Figure 3-2 depicts the format of a DOCSIS Fragmentation MAC Frame with an encrypted fragmentation payload.

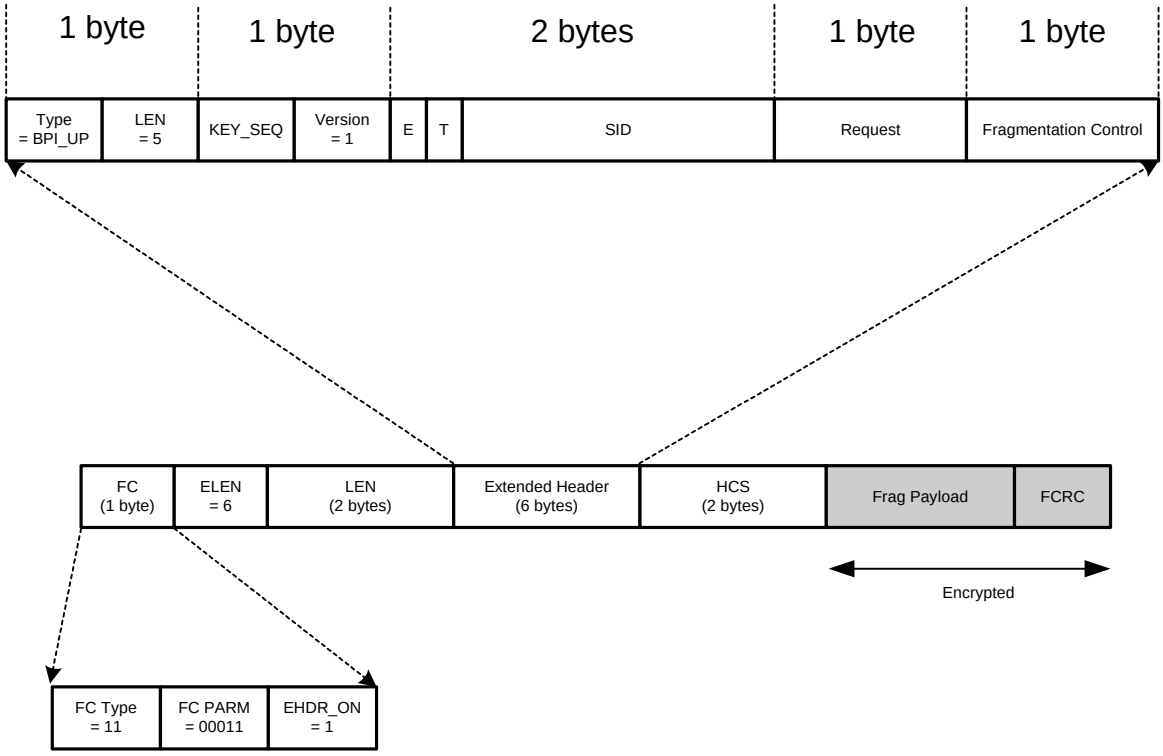


Figure 3-2. Format of a DOCSIS MAC Fragmentation Frame with an Encrypted Payload

An FC Type = 11 and FC PARM = 00011 identify a DOCSIS MAC frame as a Fragmentation frame. Unlike Packet Data PDU MAC frames, Fragmentation MAC frames have a fixed-size (six-byte) MAC Extended Header containing the "stretched" Baseline Privacy EH element.

The Fragmentation MAC header is followed by a Fragment Payload and a Fragment CRC. When Baseline Privacy encryption is applied to a Fragmentation MAC frame, the *entire* Fragment Payload is encrypted along with the Fragment CRC. In other words, unlike Baseline Privacy's encryption of Packet Data PDUs, there is no 12-byte offset into the payload before beginning encryption.[5]

5. For non-fragmented frames, the first 12-bytes are left in the clear to allow pre-decryption DA/SA filtering. For fragmented frames, DA/SA filtering cannot occur before packet reassembly; hence, there is no value in supporting the 12-byte encryption offset in DOCSIS Fragmentation MAC frames.

The LEN field of the Baseline Privacy EH element in Fragmentation MAC Frames is 5 rather than 4, accounting for the additional 1-byte fragmentation control field.[6] The KEY_SEQ field, VERSION field, ENABLE and TOGGLE flags, and SID field are what they would be for an upstream Packet Data PDU MAC Frame.

Table 3-2. Summary of the contents of a DOCSIS Fragmentation MAC Frame's Baseline Privacy EH Element.

EH_TYPE	EH_LEN	EH_VALUE
BPI__UP See [DOCSIS1]	5	KEY_SEQ (4 bits), Version (4 bits), SID (2 bytes), Request [piggyback] (1 byte), Fragmentation Control (1 byte) [CM --> CMTS] KEY_SEQ field (4 bits): Key sequence number Version field (4 bits) is defined as: 0x1 SID field is defined as: Bit[15]: ENABLE: 1..Encryption enabled; 0..Encryption Disabled Bit[14]: TOGGLE: 1..Odd Key; 0..Even Key Bit[13:0]: Service ID. Request field contains the number of mini-slots requested for upstream bandwidth. Fragmentation Control field contains fragmentation-specific control information; see [DOCSIS1] for details.

The fragmentation operation overrides BPI+ in the sense that the CM must first determine whether or not a packet will be fragmented based on grant size (the number of mini-slots a CMTS grants to a CM in an Upstream Bandwidth Allocation MAP [DOCSIS 1.1]). If the packet is to be fragmented, the BPI+ encryption MUST occur on a fragment by fragment basis, and not over the PDU as a whole; each fragment will have its own fragmentation header and be encrypted separately. If the packet is not to be fragmented, then it MUST be encrypted as a single unit, with a single privacy header.

3.3 Requirements on Usage of BP Extended Header Element in MAC Header

If BPI+ is not enabled on a particular downstream traffic flow (e.g., a CM's unicast traffic or a particular IP multicast group), the BP Extended Header element SHOULD NOT be used.

If BPI+ is not enabled for a CM's unicast traffic, fragmented upstream frames MUST still use the BP Extended Header element, but with the Encryption ENABLE bit turned off (0). The fragmented frame's piggybacked bandwidth requests MUST be carried within this BP Extended Header element.

6. edited 07/12/01 per bpi-n-01009, ab

If BPI+ is not enabled for a CM's unicast traffic, fragmented upstream frames MUST still use the BP Extended Header element, but with the Encryption ENABLE bit turned off (0). This way the BP Extended Header can still be used for piggybacked bandwidth requests according to fragmentation rules described in [DOCSIS1] or [DOCSIS9].

For MAC frames consisting of only a MAC header and optional EHDR, Baseline privacy MUST be disabled. A Baseline Privacy EHDR MAY be present on these frames, but the enable bit MUST be cleared to disable privacy.

4 Baseline Privacy Key Management (BPKM) Protocol

4.1 State Models

4.1.1 Introduction

The BPKM protocol is specified by two separate, but interdependent, state models: an authorization state model (the Authorization state machine) and an operational service key state model (the Traffic Encryption Key, or *TEK* state machine). This section defines these two state models. The state models are for explanatory purposes only, and should not be construed as constraining an actual implementation.

Cable modem authorization, controlled by the Authorization state machine, is the process of:

- the CMTS authenticating a client CM's identity
- the CMTS providing the authenticated CM with an Authorization Key, from which a Key Encryption Key (KEK) and message authentication keys are derived
- the CMTS providing the authenticated CM with the identities (i.e., the SAIDs) and properties of primary and static security associations the CM is authorized to obtain keying information for

The KEK is a two-key triple DES encryption key that the CMTS uses to encrypt the Traffic Encryption Keys (TEKs) it sends to the modem. Traffic encryption keys are used for encrypting user data traffic. CM and CMTS use message authentication keys to authenticate, via a keyed message digest, the key requests and responses they exchange.

After achieving initial authorization, a cable modem periodically seeks re-authorization with the CMTS; reauthorization is also managed by the CM's Authorization state machine. A CM MUST maintain its authorization status with the CMTS in order to be able to refresh aging Traffic Encryption Keys. TEK state machines manage the refreshing of Traffic Encryption Keys.

A cable modem begins authorization by sending an Authentication Information message to its CMTS. The Authentication Information message contains the cable modem manufacturer's X.509 certificate, issued by DOCSIS. The Authentication Information message is strictly informative, i.e., the CMTS may choose to ignore it; however it does provide a mechanism for a CMTS to learn the manufacturer certificates of its client CMs.

The cable modem sends an Authorization Request message to its CMTS immediately after sending the Authentication Information message. This is a request for an Authorization Key, as well as for the SAIDs identifying any Static Security Associations the CM is authorized to participate in. The Authorization Request includes:

- the cable modem's manufacturer ID and serial number
- the cable modem's MAC address
- the cable modem's public key
- a manufacturer-issued X.509 certificate binding the cable modem's public key to its other identifying information

- a description of the cryptographic algorithms the requesting cable modem supports; a CM's cryptographic capabilities is presented to the CMTS as a list of cryptographic suite identifiers, each indicating a particular pairing of packet data encryption and packet data authentication algorithms the CM supports
- the cable modem's Primary SAID, *which is equal to the CM's Primary SID* (the Primary SID is the first static SID the CMTS assigns to a CM during RF MAC registration)

In response to an Authorization Request message, a CMTS validates the requesting CM's identity, determines the encryption algorithm and protocol support it shares with the CM, activates an Authorization Key for the CM, encrypts it with the cable modem's public key, and sends it back to the CM in an Authorization Reply message. The authorization reply includes:

- an Authorization Key encrypted with the CM's public key
- a 4-bit key sequence number, used to distinguish between successive generations of Authorization Keys
- a key lifetime
- the identities (i.e., the SAIDs) and properties of the single primary and zero or more static security associations the CM is authorized to obtain keying information for

While the Authorization Reply MAY identify Static SAs in addition to the Primary SA whose SAID matches the requesting CM's best-effort SID, the Authorization Reply MUST NOT identify any Dynamic SAs.

The CMTS, in responding to a CM's Authorization Request, will determine whether the re-questing cable modem, whose identity can be verified via the X.509 digital certificate, is authorized for basic unicast services, and what additional statically provisioned services (i.e., Static SAIDs) the cable modem's user has subscribed for. Note that the protected services a CMTS makes available to a client CM can depend upon the particular cryptographic suites CM and CMTS share support for.

Upon achieving authorization, a CM starts a separate TEK state machine for each of the SAIDs identified in the Authorization Reply message. Each TEK state machine operating within the CM is responsible for managing the keying material associated with its respective SAID. TEK state machines periodically send Key Request messages to the CMTS, requesting a refresh of keying material for their respective SAIDs. A Key Request includes:

- identifying information unique to the cable modem, consisting of the manufacturer ID, serial number, MAC address and RSA Public Key
- the SAID whose keying material is being requested
- an HMAC keyed message digest, authenticating the Key Request

The CMTS responds to a Key Request with a Key Reply message, containing the CMTS's active keying material for a specific SAID. This keying material includes:

- the triple-DES-encrypted traffic encryption key
- CBC initialization vector
- a key sequence number
- a key's remaining lifetime
- an HMAC keyed message, authenticating the Key Reply

The traffic encryption key (TEK) in the Key Reply is triple DES (encrypt-decrypt-encrypt or EDE mode) encrypted, using a two-key, triple DES key encryption key (KEK) derived from the Authorization Key.

Note that at all times the CMTS maintains two active sets of keying material per SAID. The lifetimes of the two generations overlap such that each generation becomes active halfway through the life of it predecessor and expires halfway through the life of its successor. A CMTS includes in its Key Replies *both* of a SAID's active generations of keying material.

The Key Reply provides the requesting CM, in addition to the TEK and CBC initialization vector, the remaining lifetime of each of the two sets of keying material. The receiving CM uses these remaining lifetimes to estimate when the CMTS will invalidate a particular TEK, and therefore when to schedule future Key Requests such that the CM requests and receives new keying material before the CMTS expires the keying material the CM currently holds.

The operation of the TEK state machine's Key Request scheduling algorithm, combined with the CMTS's regimen for updating and using a SAID's keying material (see Section 6), insures that the CM will be able to continually exchange encrypted traffic with the CMTS.

A CM MUST periodically refresh its Authorization Key by re-issuing an Authorization Request to the CMTS. Reauthorization is identical to authorization with the exception that the CM does not send Authentication Information messages during reauthorization cycles. Section 4.1.2's description of the authorization state machine clearly indicates when Authentication Information messages are sent.

To avoid service interruptions during reauthorization, successive generations of the CM's Authorization Keys have overlapping lifetimes. Both CM and CMTS MUST be able to support up to two simultaneously active Authorization Keys during these transition periods. The operation of the Authorization state machine's Authorization Request scheduling algorithm, combined with the CMTS's regimen for updating and using a client CM's Authorization Keys (see Section 6), insures that CMs will be able to refresh TEK keying information without interruption over the course of the CM's reauthorization periods.

A TEK state machine remains active as long as:

- the CM is authorized to operate in the CMTS's security domain; i.e., it has a valid Authorization Key *and*
- the CM is authorized to participate in that particular Security Association; i.e. CMTS continues to provide fresh keying material during re-key cycles.

The parent Authorization state machine stops *all* of its child TEK state machines when the CM receives from the CMTS an Authorization Reject during a reauthorization cycle. Individual TEK state machines can be started or stopped during a reauthorization cycle if a CM's Static SAID authorizations changed between successive re-authorizations.

Communication between Authorization and TEK state machines occurs through the passing of events and protocol messaging. The Authorization state machine generates events (i.e., Stop, Authorized, Authorization Pending, and Authorization Complete events) that are targeted at its child TEK state machines. TEK state machines do not target events at their parent Authorization state machine. The TEK state machine affects the Authorization state machine indirectly through the messaging a CMTS sends in response to a modem's requests: a CMTS MAY respond to a TEK machine's Key Requests

with a failure response (i.e., Authorization Invalid message) that will be handled by the Authorization state machine.

4.1.1.1 Preliminary Comment on Dynamic Security Associations and Dynamic SA Mapping

Section 2 introduced Dynamic SAs and mentioned how a CMTS can establish or eliminate a Dynamic SA in response to the initiation or termination of downstream traffic flows (e.g., a particular IP multicast group's traffic). In order for a CM to run a TEK state machine to obtain a Dynamic Security Association's keying material, the CM needs to know the corresponding SAID value. The CMTS, however, does not volunteer to client CMs the existence of Dynamic SAs; instead, it is the responsibility of CMs to request of the CMTS the mappings of traffic flow identifiers (e.g., an IP multicast address) to dynamic SAIDs.

BPI+ defines a messaging exchange by which a CM learns the mapping of a downstream traffic flow to a Dynamic SA (all upstream traffic is encrypted under a CM's Primary SA). A SA Mapping state machine specifies how cable modems manage the transmission of these mapping request messages. Currently only DOCSIS's IP multicast management services utilize this mechanism. In the future, additional services may employ BPI+ Dynamic SAs.

The Authorization state machine controls the establishment and termination of TEK state machines associated with the Primary and any Static SAs; it does not, however, control the establishment and termination of TEK state machines associated with Dynamic SAs. CMs MUST implement the necessary logic to establish and terminate a Dynamic SA's TEK state machine. This interface specification, however, does not specify how CMs should manage their Dynamic SA's TEK state machines.

A full description of the SA Mapping state model is deferred to Section 5.

4.1.1.2 Security Capabilities Selection

As part of their BPI+ authorization exchange, the CM provides the CMTS with a list of all the cryptographic suites (pairing of data encryption and data authentication algorithms) the CM supports. The CMTS selects from this list a single cryptographic suite to employ with the requesting CM's primary SA. The Authorization Reply the CMTS sends back to the CM includes a primary SA descriptor which, among other things, identifies the cryptographic suite the CMTS selected to use for the CM's primary SA. A CMTS MUST reject the authorization request if it determines that none of the offered cryptographic suites are satisfactory.

The Authorization Reply also contains an optional list of static SA descriptors; each static SA descriptor identifies the cryptographic suite employed within the SA. The selection of a static SA's cryptographic suite is typically made independent of the requesting CM's cryptographic capabilities. A CMTS MAY include in its Authorization Reply static SA descriptors identifying cryptographic suites the requesting CM does not support; if this is the case, the CM MUST NOT start TEK state machines for static SAs whose cryptographic suites the CM does not support.

The above selection framework was incorporated into BPI+ in order to support future enhancements to DOCSIS hardware and to the BPI+ protocol. At the time of release of this specification, 56-bit DES

and 40-bit DES are the only packet data encryption algorithms supported, and neither are paired with a packet data authentication algorithm.

4.1.2 Authorization State Machine

The Authorization state machine consists of six states and eight distinct events (including receipt of messages) that can trigger state transitions. The Authorization finite state machine (FSM) is presented below in a graphical format, as a state flow model (Figure 4-1), and in a tabular format, as a state transition matrix (Table 4-1).

The state flow diagram depicts the protocol messages transmitted and internal events generated for each of the model's state transitions; however, the diagram does not indicate additional internal actions, such as the clearing or starting of timers, that accompany the specific state transitions. Accompanying the state transition matrix is a detailed description of the specific actions accompanying each state transition; the state transition matrix MUST be used as the definitive specification of protocol actions associated with each state transition.

The following legend applies to the Authorization State Machine flow diagram in depicted in Figure 4-1.

- ovals are states
- events are in *italics*
- messages are in normal font
- State transitions (i.e. the lines between states) are labeled with <what causes the transition>/<messages and events triggered by the transition>. So "*timeout*/Auth Request" means that the state received a "timeout" event and sent an Authorization Request ("Auth Request") message. If there are multiple events or messages before the slash "/" separated by a comma, *any* of them can cause a transition. If there are multiple events or messages listed after the slash, *all* of the specified actions must accompany the transition.

The Authorization state transition matrix presented in Table 4-1 lists the six Authorization machine states in the top-most row and the eight Authorization machine events (includes message receipts) in the left-most column. Any cell within the matrix represents a specific combination of state and event, with the next state (the state transitioned to) displayed within the cell. For example, cell 4-B represents the receipt of an Authorization Reply (Auth Reply) message when in the Authorize Wait (Auth Wait) state. Within cell 4-B is the name of the next state, "Authorized." Thus, when a CM's Authorization state machine is in the Authorize Wait state and an Authorization Reply message is received, the Authorization state machine will transition to the Authorized state. In conjunction with this state transition, several protocol actions must be taken; these are described in the listing of protocol actions, under the heading 4-B, in Section 4.1.2.5.

A shaded cell within the state transition matrix implies that either the specific event cannot or should not occur within that state, and if the event does occur, the state machine MUST ignore it. For example, if an Authorization Reply message arrives when in the Authorized state, that message should be ignored (cell 4-C). The CM MAY, however, in response to an improper event, log its occurrence, generate an SNMP event, or take some other vendor-defined action. These actions, however, are not specified within the context of the Authorization state machine, which simply ignores improper events.

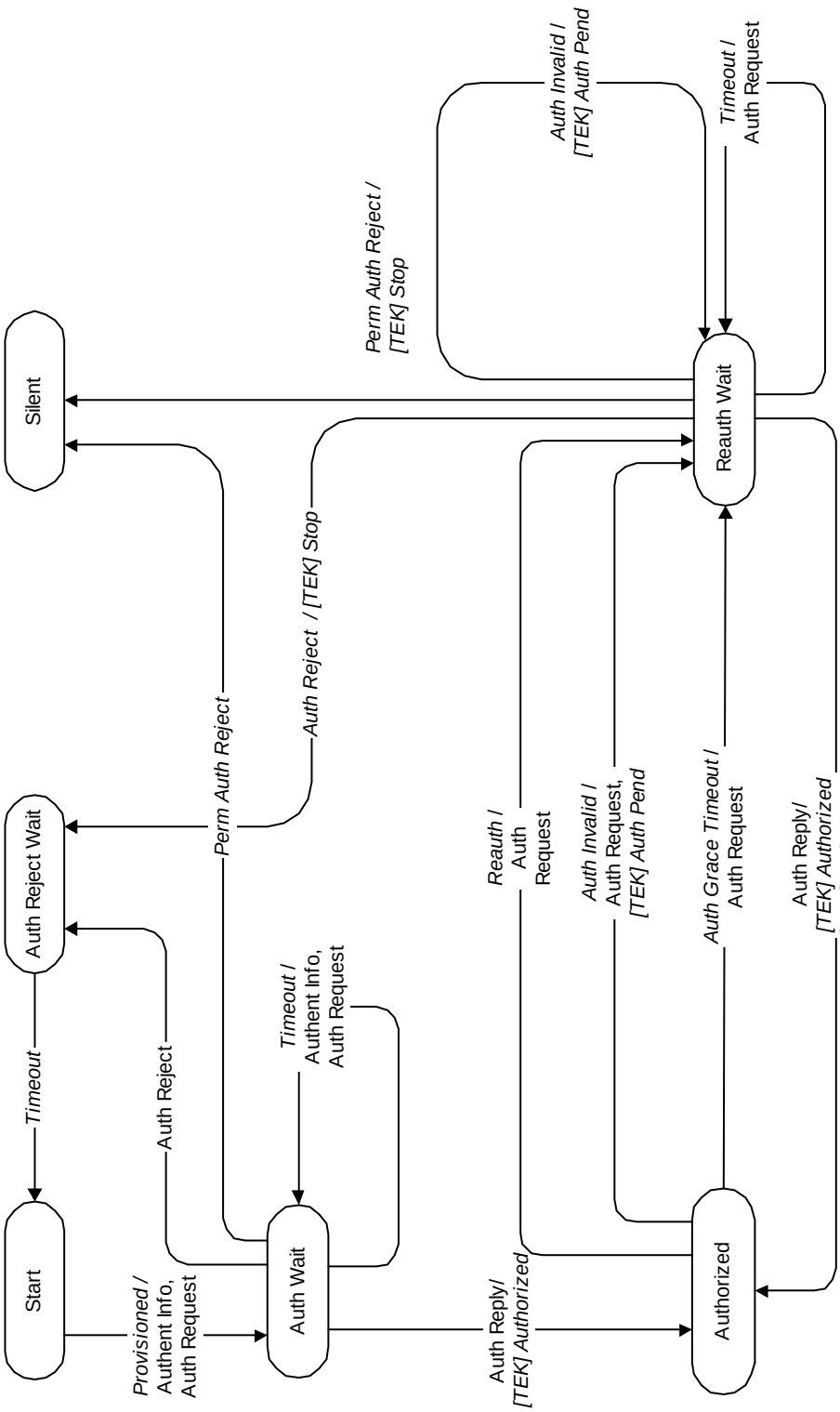


Figure 4-1. Authorization State Machine Flow Diagram

Table 4-1. Authorization FSM State Transition Matrix

State *Event or Rcvd Message*	(A) Start	(B) Auth Wait	(C) Authorized	(D) Reauth Wait	(E) Auth Reject Wait	(F) Silent
(1) *Provisioned*	Auth Wait					
(2) *Auth Reject*		Auth Reject Wait		Auth Reject Wait		
(3) *Perm Auth Reject*		Silent		Silent		
(4) *Auth Reply*		Authorized		Authorized		
(5) *Timeout*		Auth Wait		Reauth Wait	Start	
(6) *Auth Grace Timeout*			Reauth Wait			
(7) *Auth Invalid*			Reauth Wait	Reauth Wait		
(8) *Reauth*			Reauth Wait			

4.1.2.1 States

4.1.2.1.1 Start

This is the initial state of the FSM. No resources are assigned to or used by the FSM in this state—e.g., all timers are off, and no processing is scheduled.

4.1.2.1.2 Authorize Wait (Auth Wait)

The CM has received the “Provisioned” event indicating that it has completed RF MAC registration with the CMTS. In response to receiving the event, the CM has sent both an Authentication Information and an Authorize Request message to the CMTS and is waiting for the reply.

4.1.2.1.3 Authorized

The CM has received an Authorization Reply message which contains a list of valid SAIDs for this CM. At this point, the modem has a valid Authorization Key and SAID list. Transition into this state triggers the creation of one TEK FSM for each of the CM’s privacy-enabled SAIDs.

4.1.2.1.4 Reauthorize Wait (Reauth Wait)

The CM has an outstanding re-authorization request. The CM was either about to time out its current authorization or received an indication (an Authorization Invalid message from the CMTS) that its authorization was no longer valid. The CM sent an Authorization Request message to the CMTS and is waiting for a response.

4.1.2.1.5 Authorize Reject Wait (Auth Reject Wait)

The CM received an Authorization Reject message in response to its last Authorization Request. The Authorization Reject's error code indicated the error was not of a permanent nature. In response to receiving this reject message, the CM set a timer and transitioned to the Authorize Reject Wait state. The CM remains in this state until the timer expires.

4.1.2.1.6 Silent

The CM received an Authorization Reject message in response to its last Authorization Request. The Authorization Reject's error code indicated the error was of a permanent nature. This triggers a transition to the Silent state. In the Silent state, the CM MUST NOT pass CPE traffic, but MUST be able to respond to SNMP management requests arriving from across the cable network. CMTS MAY forward any IP traffic without encryption.[7]

4.1.2.2 Messages

Note that the message formats are defined in detail in Section 4.2.

4.1.2.2.1 Authorization Request (Auth Request)

Request an Authorization Key and list of authorized SAIDs. Sent from CM to CMTS.

4.1.2.2.2 Authorization Reply (Auth Reply)

Receive an Authorization Key and list of authorized, static SAIDs. Sent from CMTS to CM. The Authorization Key is encrypted with the CM's public key.

4.1.2.2.3 Authorization Reject (Auth Reject)

Attempt to authorize was rejected. Sent from the CMTS to the CM.

4.1.2.2.4 Authorization Invalid (Auth Invalid)

The CMTS can send an Authorization Invalid message to a client CM as:

- an unsolicited indication *or*

7. modified per bpi-n-02108, 07/30/02, ab

- a response to a message received from that CM.

In either case, the Authorization Invalid message instructs the receiving CM to re-authorize with its CMTS.

The CMTS responds to a Key Request with an Authorization Invalid message if (1) the CMTS does not recognize the CM as being authorized (i.e., no valid Authorization Key associated with cable modem) or (2) verification of the Key Request's keyed message digest (in HMAC-Digest Attribute) failed. Note that the Authorization Invalid *event*, referenced in both the state flow diagram and the state transition matrix, signifies either the receipt of a Authorization Invalid message or an internally generated event.

4.1.2.2.5 Authentication Information (Authent Info)

The Authentication Information message contains the cable modem manufacturer's X.509 Certificate, issued by DOCSIS. The Authent Info message is strictly an informative message the CM sends to the CMTS; with it, a CMTS MAY dynamically learn the manufacturer certificate of client CMs. Alternatively, a CMTS MAY require out-of-band configuration of its list of manufacturer certificates.

4.1.2.3 Events

4.1.2.3.1 Provisioned

The Authorization state machine generates this event upon entering the Start state if the RF MAC has completed initialization, i.e., CMTS registration. If the RF MAC initialization is not complete, the CM sends a Provisioned event to the Authorization FSM upon completing CMTS registration. The Provisioned event triggers the CM to begin the process of getting its Authorization Key and TEKs.

4.1.2.3.2 Timeout

A retransmission or wait timer timed out. Generally a request is resent.

4.1.2.3.3 Authorization Grace Timeout (Auth Grace Timeout)

The Authorization Grace timer timed out. This timer fires a configurable amount of time (the Authorization Grace Time) before the current authorization is supposed to expire, signalling the CM to re-authorize before its authorization actually expires. The Authorization Grace Time is specified in a configuration setting within the TFTP-downloaded parameter file.

4.1.2.3.4 Reauthorize (Reauth)

CM's set of authorized static SAIDs may have changed. Event generated in response to an SNMP set, [DOCSIS8], meant to trigger a reauthorization cycle.

4.1.2.3.5 Authorization Invalid (Auth Invalid)

This event can be internally generated by the CM when there is a failure authenticating a Key Reply, Key Reject, or TEK Invalid message, or externally generated by the receipt of an Authorization Invalid message, sent from the CMTS to the CM. A CMTS responds to a Key Request with an Authorization Invalid if verification of the request's message authentication code fails. Both cases indicate CMTS and CM have lost Authorization Key synchronization.

A CMTS MAY also send a CM an unsolicited Authorization Invalid message to a CM, forcing an Authorization Invalid event.

4.1.2.3.6 Permanent Authorization Reject (Perm Auth Reject)

The CM receives an Authorization Reject in response to an Authorization Request. The error code in the Authorization Reject indicates the error is of a permanent nature. What is interpreted as a permanent error is subject to administrative control within the CMTS. Authorization Request processing errors that can be interpreted as permanent error conditions include:

- unknown manufacturer (do not have CA certificate of the issuer of the CM Certificate)
- invalid signature on CM certificate
- ASN.1 parsing failure
- inconsistencies between data in the certificate and data in accompanying BPKM data Attributes
- incompatible security capabilities

BPI+'s associated OSS document [DOCSIS8] provides a description of the particular CMTS MIB objects which control the actions a CMTS takes in the event any of the above error conditions occur.

When a CM receives an Authorization Reject indicating a permanent failure condition, the Authorization State machine moves into a Silent state. CMs MUST issue an SNMP Trap upon entering the Silent state.[8]

4.1.2.3.7 Authorization Reject (Auth Reject)

The CM receives an Authorization Reject in response to an Authorization Request. The error code in the Authorization Reject does not indicate the failure was due to a permanent error condition. As a result, the CM's Authorization state machine will set a wait timer and transition into the Authorization Reject Wait State. The CM remains in this state until the timer expires, at which time it will re-attempt authorization.

Note: The following events are sent by an Authorization state machine to the TEK state machine.

4.1.2.3.8 [TEK] Stop

Sent by the Authorization FSM to an active (non -START state) TEK FSM to terminate the FSM and remove the corresponding SAID's keying material from the CM's key table.

8. modified per bpi-n-02108, 07/30/02, ab

4.1.2.3.9 [TEK] Authorized

Sent by the Authorization FSM to a non-active (START state), but valid TEK FSM.

4.1.2.3.10 [TEK] Authorization Pending (Auth Pend)

Sent by the Authorization FSM to a specific TEK FSM to place that TEK FSM in a wait state until the Authorization FSM can complete its re-authorization operation.

4.1.2.3.11 [TEK] Authorization Complete (Auth Comp)

Sent by the Authorization FSM to a TEK FSM in the Operational Reauthorize Wait (Op Reauth Wait) or Rekey Reauthorize Wait (Rekey Reauth Wait) states to clear the wait state begun by a TEK FSM Authorization Pending event.

4.1.2.4 Parameters

All configuration parameter values are specified in the TFTP-downloaded parameter file (see Appendix A: TFTP Configuration File Extensions).

4.1.2.4.1 Authorize Wait Timeout (Auth Wait Timeout)

Timeout period between sending Authorization Request messages from Authorize Wait state. See A.1.1.1.1.

4.1.2.4.2 Reauthorize Wait Timeout (Reauth Wait Timeout)

Timeout period between sending Authorization Request messages from Reauthorize Wait state. See A.1.1.1.2.

4.1.2.4.3 Authorization Grace Time (Auth Grace Timeout)

Amount of time before authorization is scheduled to expire that the CM starts re-authorization. See A.1.1.1.3.

4.1.2.4.4 Authorize Reject Wait Timeout (Auth Reject Wait Timeout)

Amount of time a CM's Authorization FSM remains in the Authorize Reject Wait state before transitioning to the Start state. See A.1.1.1.7.

4.1.2.5 Actions

Actions taken in association with state transitions are listed by <event/rcvd message> - <state> below:

1-A Start (*Provisioned*) → Auth Wait

- send Authentication Information message to CMTS
- send Authorization Request message to CMTS
- set Authorization Request retry timer to Authorize Wait Timeout

2-B Auth Wait (Auth Reject) → Auth Reject Wait

- clear Authorization Request retry timer
- set a wait timer to Authorize Reject Wait Timeout

2-D Reauth Wait (Auth Reject) → Auth Reject Wait

- clear Authorization Request retry timer
- generate TEK FSM Stop events for all active TEK state machines
- set a wait timer to Authorize Reject Wait Timeout

3-B Auth Wait (Perm Auth Reject) → Silent

- clear Authorization Request retry timer
- disable all forwarding of CPE traffic

3-D Reauth Wait (Perm Auth Reject) → Silent

- clear Authorization Request retry timer
- generate TEK FSM Stop events for all active TEK state machines
- disable all forwarding of CPE traffic

4-B Auth Wait (Auth Reply) → Authorized

- clear Authorization Request retry timer
- decrypt and record Authorization Key delivered with Authorization Reply
- start TEK FSMs for all SAIDs listed in Authorization Reply (provided the CM supports the cryptographic suite that is associated with a SAID) and issue a TEK FSM Authorized event for each of the new TEK FSMs
- set the Authorization Grace timer to go off "Authorization Grace Time" seconds prior to the supplied Authorization Key's scheduled expiration

4-D Reauth Wait (Auth Reply) → Authorized

- clear Authorization Request retry timer
- decrypt and record Authorization Key delivered with Authorization Reply
- start TEK FSMs for any newly authorized SAIDs listed in Authorization Reply (provided the CM supports the cryptographic suite that is associated with the new SAID) and issue TEK FSM Authorized event for each of the new TEK FSMs
- generate TEK FSM Authorization Complete events for any currently active TEK FSMs whose corresponding SAIDs were listed in Authorization Reply
- generate TEK FSM Stop events for any currently active TEK FSMs whose corresponding SAIDs were not listed in Authorization Reply

- set the Authorization Grace timer to go off "Authorization Grace Time" seconds prior to the supplied Authorization Key's scheduled expiration

5-B Auth Wait (*Timeout*) → Auth Wait

- send Authentication Information message to CMTS
- send Authorization Request message to CMTS
- set Authorization Request retry timer to Authorize Wait Timeout

5-D Reauth Wait (*Timeout*) → Reauth Wait

- send Authorization Request message to CMTS
- set Authorization Request retry timer to Reauthorize Wait Timeout

5-E Auth Reject Wait (*Timeout*) → Start

- no protocol actions associated with state transition

6-C Authorized (*Auth Grace Timeout*) → Reauth Wait

- send Authorization Request message to CMTS
- set Authorization Request retry timer to Reauthorize Wait Timeout

7-C Authorized (*Auth Invalid*) → Reauth Wait

- clear Authorization Grace timer
- send Authorization Request message to CMTS
- set Authorization Request retry timer to Reauthorize Wait Timeout
- if the Authorization Invalid event is associated with a particular TEK FSM, generate a TEK FSM Authorization Pending event for the TEK state machine responsible for the Authorization Invalid event (i.e., the TEK FSM that either generated the event, or sent the Key Request message the CMTS responded to with an Authorization Invalid message)

7-D Reauth Wait (*Auth Invalid*) → Reauth Wait

- if the Authorization Invalid event is associated with a particular TEK FSM, generate a TEK FSM Authorization Pending event for the TEK state machine responsible for the Authorization Invalid event (i.e., the TEK FSM that either generated the event, or sent the Key Request message the CMTS responded to with an Authorization Invalid message)

8-C Authorized (*Reauth*) → Reauth Wait

- clear Authorization grace timer
- send Authorization Request message to CMTS
- set Authorization Request retry timer to Reauthorize Wait Timeout

4.1.3 TEK State Machine

The TEK state machine consists of six states and nine events (including receipt of messages) that can trigger state transitions. Like the Authorization state machine, the TEK state machine is presented in both a state flow diagram and a state transition matrix. And as was the case for the Authorization state machine, the state transition matrix MUST be used as the definitive specification of protocol actions associated with each state transition.

Shaded states in Figure 4-2 (Operational, Rekey Wait, and Rekey Reauthorize Wait) have valid keying material and encrypted traffic can be passed.

The Authorization state machine starts an independent TEK state machine for each of its authorized SAIDs.

As mentioned previously in Section 4.1.1, the CMTS maintains two active TEKs per SAID. The CMTS includes in its Key Replies both of these TEKs, along with their remaining lifetimes. The CMTS encrypts downstream traffic with the older of its two TEKs and decrypts upstream traffic with either the older or newer TEK, depending upon which of the two keys the CM was using at the time. The CM encrypts upstream traffic with the newer of its two TEKs and decrypts downstream traffic with either the older or newer TEK, depending upon which of the two keys the CMTS was using at the time. See Section 6 for details on CM and CMTS key usage requirements.

Through operation of a TEK state machine, the CM attempts to keep its copies of a SAID's TEKs synchronized with those of its CMTS. A TEK state machine issues Key Requests to refresh copies of its SAID's keying material after the scheduled expiration time of the older of its two TEKs and before the expiration of its newer TEK. To accommodate for CM/CMTS clock skew and other system processing and transmission delays, the CM schedules its Key Requests a configurable number of seconds (*i.e.*, "TEK Grace Time") before the newer TEK's estimated expiration in the CMTS. With the receipt of the Key Reply, the CM MUST always update its records with the TEK Parameters from both TEKs contained in the Key Reply Message. Figure 6-2 illustrates the CM's scheduling of its key refreshes in conjunction with its management of a BPI+ SA's active TEKs.

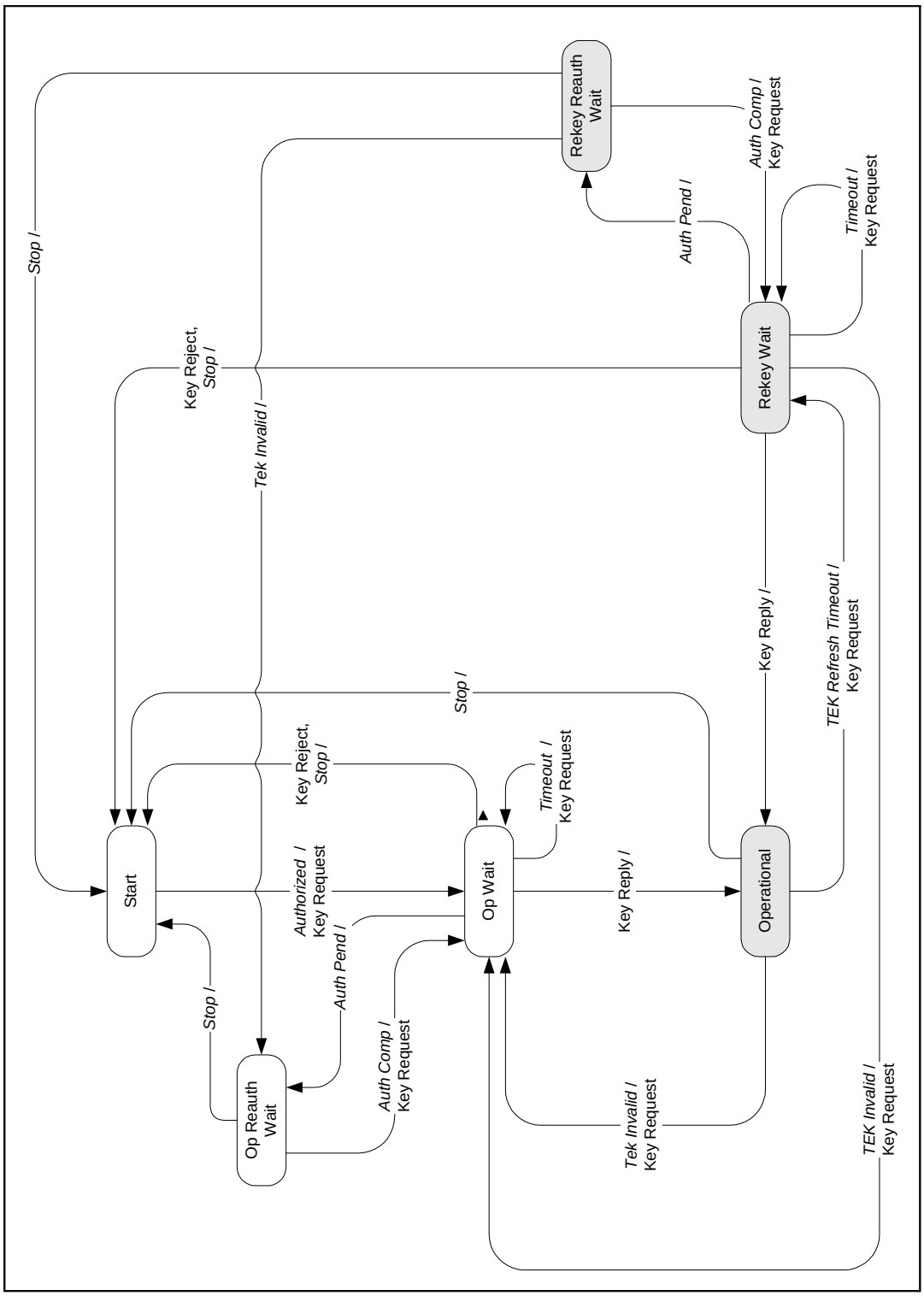


Figure 4-2. TEK State Machine Flow Diagram

Table 4-2. TEK FSM State Transition Matrix

State *Event or Rcvd Message*	(A) Start	(B) Op Wait	(C) Op Reauth Wait	(D) Op	(E) Rekey Wait	(F) Rekey Reauth Wait
(1) *Stop*		Start	Start	Start	Start	Start
(2) *Authorized*	Op Wait					
(3) *Auth Pend*		Op Reauth Wait			Rekey Re-auth Wait	
(4) *Auth Comp*			Op Wait			Rekey Wait
(5) *TEK Invalid*				Op Wait	Op Wait	Op Reauth Wait
(6) *Timeout*		Op Wait			Rekey Wait	
(7) *TEK Refresh Tim-eout*				Rekey Wait		
(8) Key Reply		Operational			Operational	
(9) Key Reject		Start			Start	

4.1.3.1 States

4.1.3.1.1 Start

This is the initial state of the FSM. No resources are assigned to or used by the FSM in this state—e.g., all timers are off, and no processing is scheduled.

4.1.3.1.2 Operational Wait (Op Wait)

The TEK state machine has sent its initial request (Key Request) for its SAID's keying material (traffic encryption key and CBC initialization vector), and is waiting for a reply from the CMTS.

4.1.3.1.3 Operational Reauthorize Wait (Op Reauth Wait)

The wait state the TEK state machine is placed in if it does not have valid keying material while the Authorization state machine is in the in the middle of a reauthorization cycle.

4.1.3.1.4 Operational

The CM has valid keying material for the associated SAID.

4.1.3.1.5 Rekey Wait

The TEK Refresh Timer has expired and the CM has requested a key update for this SAID. Note that the newer of its two TEKs has not expired and can still be used for both encrypting and decrypting data traffic.

4.1.3.1.6 Rekey Reauthorize Wait (Rekey Reauth Wait)

The wait state the TEK state machine is placed in if the TEK state machine has valid traffic keying material, has an outstanding request for the latest keying material, and the Authorization state machine initiates a reauthorization cycle.

4.1.3.2 Messages

Note that the message formats are defined in detail in Section 4.2.

4.1.3.2.1 Key Request

Request a TEK for this SAID. Sent by the CM to the CMTS and authenticated with keyed message digest. The message authentication key is derived from the Authorization Key.

4.1.3.2.2 Key Reply

Response from the CMTS carrying the two active sets of traffic keying material for this SAID. Sent by the CMTS to the CM, it includes the SAID's traffic encryption keys, triple DES encrypted with a key encryption key derived from the Authorization Key. The Key Reply message is authenticated with a keyed message digest; the authentication key is derived from the Authorization Key.

4.1.3.2.3 Key Reject

Response from the CMTS to the CM to indicate this SAID is no longer valid and no key will be sent. The Key Reject message is authenticated with a keyed message digest; the authentication key is derived from the Authorization Key

4.1.3.2.4 TEK Invalid

The CMTS sends a CM this message if it determines that the CM encrypted an upstream Packet Data PDU with an invalid TEK; i.e., a SAID's TEK key sequence number, contained within the received packet's Baseline Privacy Extended Header element, is out of the CMTS's range of known, valid sequence numbers for that SAID.

4.1.3.3 Events

4.1.3.3.1 Stop

Sent by the Authorization FSM to an active (non-START state) TEK FSM to terminate TEK FSM and remove the corresponding SAID's keying material from the CM's key table. See Section 4.1.2.3.8.

4.1.3.3.2 Authorized

Sent by the Authorization FSM to a non-active (START state) TEK FSM to notify TEK FSM of successful authorization. See Section 4.1.2.3.9.

4.1.3.3.3 Authorization Pending (Auth Pend)

Sent by the Authorization FSM to TEK FSM to place TEK FSM in a wait state while Authorization FSM completes re-authorization. See Section 4.1.2.3.10.

4.1.3.3.4 Authorization Complete (Auth Comp)

Sent by the Authorization FSM to a TEK FSM in the Operational Reauthorize Wait or Rekey Reauthorize Wait states to clear the wait state begun by the prior Authorization Pending event. See Section 4.1.2.3.11.

4.1.3.3.5 TEK Invalid

This event can be triggered by either a CM's data packet decryption logic, or by the receipt of a TEK Invalid message from the CMTS.

A CM's data packet decryption logic triggers a TEK Invalid event if it recognizes a loss of TEK key synchronization between itself and the encrypting CMTS; i.e., a SAID's TEK key sequence number, contained within the received, downstream packet's Baseline Privacy Extended Header element, is out of the CM's range of known sequence numbers for that SAID.

A CMTS sends a CM a TEK Invalid message, triggering a TEK Invalid event within the CM, if the CMTS's decryption logic recognizes a loss of TEK key synchronization between itself and the CM.

4.1.3.3.6 Timeout

A retry timer timeout. Generally, the particular request is retransmitted.

4.1.3.3.7 TEK Refresh Timeout

The TEK refresh timer timed out. This timer event signals the TEK state machine to issue a new Key Request in order to refresh its keying material. The refresh timer is set to fire a configurable length of time (*TEK Grace Time*) before the expiration of the newer TEK the CM currently holds. This is configured via the CMTS to occur after the scheduled expiration of the older of the two TEKs.

4.1.3.4 Parameters

All configuration parameter values are specified in TFTP downloaded parameter file (see Appendix A: TFTP Configuration File Extensions).

4.1.3.4.1 Operational Wait Timeout

Timeout period between sending of Key Request messages from the Op Wait state. See A.1.1.1.4.

4.1.3.4.2 Rekey Wait Timeout

Timeout period between sending of Key Request messages from the Rekey Wait state. See A.1.1.1.5.

4.1.3.4.3 TEK Grace Time

Time interval, in seconds, before the estimated expiration of a TEK that the CM starts rekeying for a new TEK.

TEK Grace Time is specified in a configuration setting within the TFTP-downloaded parameter file, and is the same across all SAIDs. See Section A.1.1.1.6.

4.1.3.5 Actions

1-B Op Wait (*Stop*) → Start

- clear Key Request retry timer
- terminate TEK FSM

1-C Op Reauth Wait (*Stop*) → Start

- terminate TEK FSM

1-D Operational (*Stop*) → Start

- clear TEK refresh timer, which is timer set to go off *"Tek Grace Time"* seconds prior to the TEK's scheduled expiration time
- terminate TEK FSM
- remove SAID keying material from key table

1-E Rekey Wait(*Stop*) → Start

- clear Key Request retry timer
- terminate TEK FSM
- remove SAID keying material from key table

1-F Rekey Reauth Wait(*Stop*) → Start

- terminate TEK FSM
- remove SAID keying material from key table

2-A Start (*Authorized*) → Op Wait

- send Key Request Message to CMTS
- set Key Request retry timer to Operational Wait Timeout

3-B Op Wait (*Auth Pend*) → Op Reauth Wait

- clear Key Request retry timer

3-E Rekey Wait (*Auth Pend*) → Rekey Reauth Wait

- clear Key Request retry timer

4-C Op Reauth Wait (*Auth Comp*) → Op Wait

- send Key Request message to CMTS
- set Key Request retry timer to Operational Wait Timeout

4-F Rekey Reauth Wait (*Auth Comp*) → Rekey Wait

- send Key Request message to CMTS
- set Key Request retry timer to Rekey Wait Timeout

5-D Operational (*TEK Invalid*) → Op Wait

- clear TEK refresh timer
- send Key Request message to CMTS
- set Key Request retry timer to Operational Wait Timeout
- remove SAID keying material from key table

5-E Rekey Wait (*TEK Invalid*) → Op Wait

- clear Key Request retry timer
- send Key Request message to CMTS
- set Key Request retry timer to Operational Wait Timeout
- remove SAID keying material from key table

5-F Rekey Reauth Wait (*TEK Invalid*) → Op Reauth Wait

- remove SAID keying material from key table

6-B Op Wait (*Timeout*) → Op Wait

- send Key Request message to CMTS
- set Key Request retry timer to Operational Wait Timeout

6-E Rekey Wait (*Timeout*) → Rekey Wait

- send Key Request message to CMTS
- set Key Request retry timer to Rekey Wait Timeout

7-D Operational (*TEK Grace Timeout*) → Rekey Wait

- send Key Request message to CMTS
- set Key Request retry timer to Rekey Wait Timeout

8-B Op Wait (Key Reply) → Operational

Note: Key Reply passed message authentication.

- clear Key Request retry timer
- process contents of Key Reply message and incorporate new keying material into key database
- set the TEK refresh timer to go off "TEK Grace Time" seconds prior to the key's scheduled expiration

8-E Rekey Wait (Key Reply) → Operational

Note: Key Reply passed message authentication.

- clear Key Request retry timer
- process contents of Key Reply message and incorporate new keying material into key database
- set the TEK refresh timer to go off "TEK Grace Time" seconds prior to the key's scheduled expiration

9-B Op Wait (Key Reject) → Start

Note: Key Reject passed message authentication.

- clear Key Request retry timer
- terminate TEK FSM

9-E Rekey Wait (Key Reject) → Start

- clear Key Request retry timer
- terminate TEK FSM
- remove SAID keying material from key table

4.2 Key Management Message Formats [9]

Baseline Privacy Key Management employs two MAC message types: BPKM-REQ and BPKM-RSP. [DOCSIS1] defines the specific type values assigned to them.

Table 4-3. Baseline Privacy Key Management MAC Messages

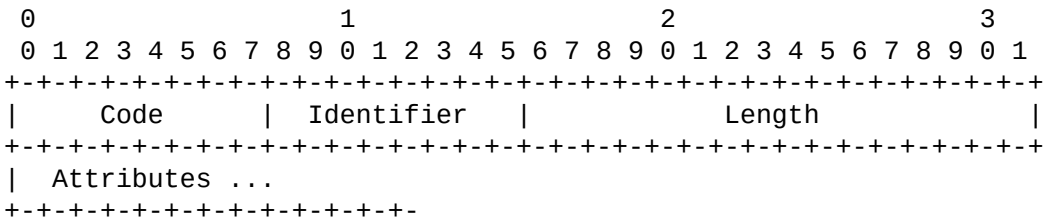
Type Value	Message Name	Message Description
See [DOCSIS1]	BPKM-REQ	Privacy Key Management Request [CM -> CMTS]
See [DOCSIS1]	BPKM-RSP	Privacy Key Management Response [CMTS -> CM]

While these two MAC management message types distinguish between BPKM requests (CM to CMTS) and responses (CMTS to CM), more detailed information about message contents is encoded in the BPKM messages themselves. This maintains a clean separation between privacy management functions and RF MAC upstream bandwidth allocation, timing and synchronization (RF MAC management's principal responsibilities).

4.2.1 Packet Formats

Exactly one BPKM message is encapsulated in the Management Message Payload field of a MAC management message.

A summary of the BPKM message format is shown below. The fields are transmitted from left to right.



Code

The Code field is one octet, and identifies the type of BPKM packet. When a packet is received with an invalid Code field, it SHOULD be silently discarded.

BPKM Codes (decimal) are assigned as follows:

9. Message formats for the Baseline Privacy Key Management protocol are modeled after those of the Remote Authentication Dial In User Service (RADIUS) protocol, defined in RFC 2058, and an Internet standards track protocol. BPKM, like RADIUS, adheres to a client/server model. Unlike RADIUS, BPKM will not run over UDP/IP. BPKM messages are encapsulated within RF MAC management messages.

Table 4-4. Baseline Privacy Key Management Message Codes

Code	BPKM Message Type	MAC Management Message Name
0-3	Reserved	-
4	Auth Request	BPKM-REQ
5	Auth Reply	BPKM-RSP
6	Auth Reject	BPKM-RSP
7	Key Request	BPKM-REQ
8	Key Reply	BPKM-RSP
9	Key Reject	BPKM-RSP
10	Auth Invalid	BPKM-RSP
11	TEK Invalid	BPKM-RSP
12	Authent Info	BPKM-REQ
13	Map Request	BPKM-REQ
14	Map Reply	BPKM-RSP
15	Map Reject	BPKM-RSP
16-255	Reserved	-

Identifier

The Identifier field is one octet. A CM uses the identifier to match a CMTS's responses to the CM's requests.

The CM MUST change (e.g., increment, wrapping around to 0 after reaching 255) the Identifier field whenever it issues a new BPKM message. A “new” message is an Authorization Request, Key Request or SA Map Request that is not a retransmission being sent in response to a Timeout event. For retransmissions, the Identifier field MUST remain unchanged.

The Identifier field in Authentication Information messages, which are informative and do not effect any response messaging, MAY be set to zero.

The Identifier field in a CMTS’s BPKM response message MUST match the Identifier field of the BPKM Request message the CMTS is responding to. The Identifier field in TEK Invalid messages, which are not sent in response to BPKM requests, MUST be set to zero. The Identifier field in unsolicited Authorization Invalid messages MUST be set to zero.

On reception of a BPKM response message, the CM associates the message with a particular state machine (the Authorization state machine in the case of Authorization Replies, Authorization Rejects, and Authorization Invalids; a particular TEK state machine in the case of Key Replies, Key Rejects and TEK Invalids; a particular SA Mapping state machine in the case of SA Map Replies and SA Map Rejects).

A CM MAY keep track of the Identifier of its latest, pending Authorization Request. The CM MAY silently discard Authorization Replies and Authorization Rejects whose Identifier fields do not match those of the pending requests.

A CM MAY keep track of the Identifier of its latest, pending Key Request. The CM MAY silently discard Key Replies and Key Rejects whose Identifier fields do not match those of the pending requests.

A CM MAY keep track of the Identifier of its latest, pending SA Map Request. The CM MAY silently discard SA Map Replies and SA Map Rejects whose Identifier fields do not match those of the pending requests.

Length

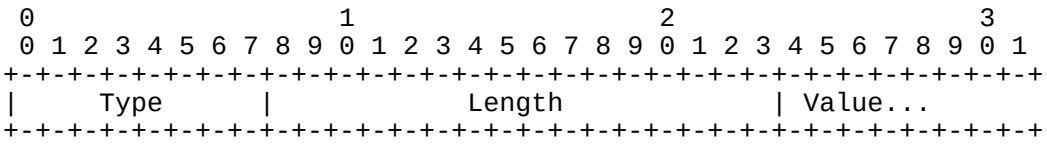
The Length field is two octets. It indicates the length of the Attribute fields in octets. The length field does not include the Code, Identifier and Length fields. Octets outside the range of the Length field MUST be treated as padding and ignored on reception. If the packet is shorter than the Length field indicates, it SHOULD be silently discarded. The minimum length is 0 and maximum length is 1490.

Attributes

BPKM Attributes carry the specific authentication, authorization and key management data exchanged between client and server. Each BPKM packet type has its own set of required and optional Attributes. Unless explicitly stated, there are no requirements on the ordering of attributes within a BPKM message.

The end of the list of Attributes is indicated by the Length of the BPKM packet.

Attributes are type/length/value (TLV) encoded, as shown below. The fields are transmitted from left to right.



Packet formats for each of the BPKM messages are described below. The descriptions list the BPKM attributes contained within each BPKM message type. The Attributes themselves are described in Section 4.2.2. Unknown attributes MUST be ignored on receipt, and skipped over while scanning for recognized attributes.

The CMTS MUST silently discard all requests that do not contain ALL required attributes. The CM MUST silently discard all responses that do not contain ALL required attributes.

4.2.1.1 Authorization Request (Auth Request)

Code: 4

Attributes:

Table 4-5. Authorization Request Attributes

Attribute	Contents
CM-Identification	contains information used to identify cable modem to CMTS
CM-Certificate	contains the CM's X.509 user certificate
Security-Capabilities	describes requesting CM's security capabilities
SAID	CM's primary SAID equal to the Primary SID

The CM-Identification attribute contains a set of data that identifies the requesting cable modem to the CMTS. Note that the CMTS is in all likelihood using only a single item in the CM-Identification attribute (e.g., CM MAC address) as a CM handle. While a specific item could be selected for inclusion in the Authorization Request message, including the entire CM-Identification attribute for client identification provides vendors with greater flexibility in the headend's system design.

The CM-Certificate attribute contains an X.509 CM certificate issued by the CM's manufacturer. The CM's X.509 certificate is a public-key certificate which binds the CM's identifying information to it's RSA public key in a verifiable manner. The X.509 certificate is digitally signed by the CM's manufacturer, and that signature can be verified by a CMTS that knows the manufacturer's public key. The manufacturer's public key is placed in an X.509 certification authority (CA) certificate, which in turn is signed by a higher level certification authority, i.e., DOCSIS.

The Security-Capabilities attribute is a compound attribute describing the requesting cable modem's security capabilities. This includes the packet data encryption algorithm(s) a CM supports and the packet data authentication algorithm(s) supported (of which there are currently none) and the version of the Baseline Privacy Protocol supported (of which there is currently one: version 1 for BPI+).

A SAID attribute contains a Baseline Privacy security association identifier, or SAID. In this case, the provided SAID is the CM's BPI+ primary SAID, which is equal to the Primary SID assigned to the cable modem during RF MAC registration.[10]

4.2.1.2 Authorization Reply (Auth Reply)

Sent by the CMTS to a client CM in response to an Authorization Request, the Authorization Reply message contains an Authorization Key, the key's lifetime, the key's sequence number, and a list of SA-Descriptors identifying the Primary and Static Security Associations the requesting cable modem is authorized to access and their particular properties (e.g., type, cryptographic suite). The Authorization Key MUST be encrypted with the CM's public key. The SA-Descriptor list MUST include a descriptor for the primary BPI+ SAID reported to the CMTS in the corresponding Authorization Request. The SA-Descriptor list MAY include descriptors of Static SAIDs the CM is authorized to access.

Code field: 5

Attributes:

Table 4-6. Authorization Reply Attributes

Attribute	Contents
AUTH-Key	Authorization (AUTH) Key, encrypted with the target client CM's public key
Key-Lifetime	Authorization key lifetime
Key-Sequence-Number	Authorization key sequence number
(one or more) SA-Descriptor	Each SA-Descriptor compound Attribute specifies a SAID and additional properties of the SA.

4.2.1.3 Authorization Reject (Auth Reject)

CMTS responds to a CM's authorization request with an Authorization Reject message if the CMTS rejects the CM's authorization request.

Code field: 6

10. typo corrected per bpi-n-02011, 07/25/02, ab

Attributes:

Table 4-7. Auth Rej Attributes

Attribute	Contents
Error-Code	Error code identifying reason for rejection of authorization request
Display-String (optional)	Display String providing reason for rejection of authorization request

The Error-Code and Display-String attributes describe to the requesting CM the reason for the authorization failure.

4.2.1.4 Key Request

Code: 7

Attributes:

Table 4-8. Key Request Attributes

Attribute	Contents
CM-Identification	Contains information used to identify cable modem to CMTS
Key-Sequence-Number	Authorization key sequence number
SAID	Security Association ID
HMAC-Digest	Keyed SHA message digest

The HMAC-Digest Attribute is a keyed message digest. The HMAC-Digest Attribute MUST be the final Attribute in the Key Request's Attribute list. The message digest is performed over the packet header and all of the Key Request's Attributes, other than the HMAC-Digest, in the order in which they appear within the packet.

Inclusion of the keyed digest allows the CMTS to authenticate the Key Request message. The HMAC-Digest's authentication key is derived from the Authorization Key. See Section 7, Cryptographic Methods, for details.

4.2.1.5 Key Reply

Code: 8

Attributes:

Table 4-9. Key Reply Attributes

Attribute	Contents
Key-Sequence-Number	Authorization key sequence number
SAID	Security Association ID
TEK-Parameters	“Older” generation of key parameters relevant to SAID
TEK-Parameters	“Newer” generation of key parameters relevant to SAID
HMAC-Digest	Keyed SHA message digest

The TEK-Parameters Attribute is a compound attribute containing all of the keying material corresponding to a particular generation of a SAID's TEK. This would include the TEK, the TEK’s remaining key lifetime, its key sequence number, and the CBC initialization vector. The TEK is encrypted. See Section 4.2.2.13 for details.

At all times the CMTS maintains two sets of active generations of keying material per SAID. (A set of keying material includes the a TEK and its corresponding CBC initialization vector.) One set corresponds to the “older” generation of keying material, the second set corresponds to the “newer” generation of keying material. The newer generation has a key sequence number one greater than (modulo 16) that of the older generation. Section 6.1 specifies CMTS requirements for maintaining and using a SAID’s two active generations of keying material.

The CMTS distributes to a client CM both generations of active keying material. Thus, the Key Reply message contains two TEK-Parameters Attributes, each containing the keying material for one of the SAIDs two active sets of keying material.

The HMAC-Digest Attribute is a keyed message digest. The HMAC-Digest Attribute MUST be the final Attribute in the Key Reply’s Attribute list. The message digest is performed over the BPKM message header (starting with the BPKM Code field) and all of the Key Reply’s Attributes, other than the HMAC-Digest, in the order in which they appear within the packet.

Inclusion of the keyed digest allows the receiving client to authenticate the Key Reply message and ensure CM and CMTS have synchronized Authorization Keys. The HMAC-Digest’s authentication key is derived from the Authorization Key. See Section 7, Cryptographic Methods, for details.

4.2.1.6 Key Reject

Receipt of a Key Reject indicates the receiving client CM is no longer authorized for a particular SAID.

Code: 9

Attributes:

Table 4-10. Key Reject Attributes

Attribute	Contents
Key-Sequence-Number	Authorization key sequence number
SAID	Security Association ID
Error-Code	Error code identifying reason for rejection of Key Request
Display-String (optional)	Display string containing reason for Key Reject
HMAC-Digest	Keyed SHA message digest

The HMAC-Digest Attribute is a keyed message digest. The HMAC-Digest Attribute MUST be the final Attribute in the Key Reject's Attribute list. The message digest is performed over the BPKM message header (starting with the BPKM Code field) and all of the Key Reject's Attributes, other than the HMAC-Digest, in the order in which they appear within the packet.

Inclusion of the keyed digest allows the receiving client to authenticate the Key Reject message and ensure CM and CMTS have synchronized Authorization Keys. The HMAC-Digest's authentication key is derived from the Authorization Key. See Section 7, Cryptographic Methods, for details.

4.2.1.7 Authorization Invalid

The CMTS can send an Authorization Invalid message to a client CM as:

- an unsolicited indication, or
- a response to a message received from that CM

In either case, the Authorization Invalid message instructs the receiving CM to re-authorize with its CMTS.

The CMTS sends an Authorization Invalid in response to a Key Request if (1) the CMTS does not recognize the CM as being authorized (i.e., no valid Authorization Key associated with the requesting cable modem) or (2) verification of the Key Request's keyed message digest (in HMAC-Digest Attribute) failed, indicating a loss of Authorization Key synchronization between CM and CMTS.

Code: 10

Attributes:

Table 4-11. Authorization Invalid Attributes

Attribute	Contents
Error-Code	Error code identifying reason for Authorization Invalid
Display-String (optional)	Display String describing failure condition

4.2.1.8 TEK Invalid

The CMTS sends a TEK Invalid message to a client CM if the CMTS determines that the CM encrypted an upstream Packet Data PDU with an invalid TEK; i.e., a SAID's TEK key sequence number, contained within the received packet's Baseline Privacy Extended Header element, is out of the CMTS's range of known, valid sequence numbers for that SAID.

Code: 11

Attributes:

Table 4-12. TEK Invalid Attributes

Attribute	Contents
Key-Sequence-Number	Authorization key sequence number
SAID	Security Association ID
Error-Code	Error code identifying reason for TEK Invalid message
Display-String (optional)	Display string containing vendor-defined in-formation
HMAC-Digest	Keyed SHA message digest

The HMAC-Digest Attribute is a keyed message digest. The HMAC-Digest Attribute MUST be the final Attribute in the TEK Invalid's Attribute list. The message digest is performed over the BPKM message header (starting with the BPKM Code field) and all of the TEK Invalid's Attributes, other than the HMAC-Digest, in the order in which they appear within the packet.

Inclusion of the keyed digest allows the receiving client to authenticate the TEK Invalid message and ensure CM and CMTS have synchronized Authorization Keys. The HMAC-Digest's authentication key is derived from the Authorization Key. See Section 7, Cryptographic Methods, for details.

4.2.1.9 Authentication Information (Authent Info)

The Authentication Info message contains a single CA-Certificate Attribute, containing an X.509 CA certificate for the manufacturer of the CM. The CM's X.509 user certificate MUST have been issued by the certification authority identified by the X.509 CA certificate. All X.509 CA certificates MUST be issued by a DOCSIS root certification authority.

Authentication Information messages are strictly informative: while the CM MUST transmit Authent Info messages as indicated by the Authentication state model (Section 4.1.2), the CMTS MAY ignore them.

Code: 12

Attributes:

Table 4-13. Authentication Information Attributes

Attribute	Contents
CA-Certificate	certificate of manufacturer CA that issued CM certificate

The CA-certificate attribute contains an X.509 CA certificate for the CA that issued the CM's X.509 user certificate. The DOCSIS certification authority issues these CA-certificates to DOCSIS-certified CM manufacturers.

4.2.1.10 SA Map Request (MAP Request)

A CM modem sends SA Map Requests to its CMTS to request the mapping of a particular downstream traffic flow to a BPI+ SA. Section 5 describes the SA Mapping state model which uses the message.

Code: 13

Attributes:

Table 4-14. SA Map Request Attributes

Attribute	Contents
CM-Identification	Contains information used to identify cable modem to CMTS
SA-Query	Contains addressing information identifying the downstream traffic flow CM is requesting an SA mapping for

4.2.1.11 SA Map Reply (Map Reply)

A CMTS sends an SA Map Reply as a positive response to a client CM's SA Map Request. The SA Map Reply informs the CM of a mapping between a queried address and a BPI+ SA. Section 5 describes the SA Mapping state model which uses the message.

Code: 14

Attributes:

Table 4-15. SA Map Reply Attributes

Attribute	Contents
SA-Query	Contains addressing information identifying the downstream traffic flow CM is requested an SA mapping for
SA-Descriptor	SA-Descriptor compound Attribute specifies the mapped SA's SAID and other properties.

4.2.1.12 SAID Map Reject (Map Reject)

A CMTS sends SA Map Reject as a negative response to a client CM's SA Map Request. The SA Map Reject informs the CM that either (1) downstream traffic flow identified in the SA-Query Attribute is not being encrypted or (2) the requesting CM is not authorized to receive that traffic. The contents of an error code attribute distinguishes between the two cases. Section 5 describes the SA Mapping state model which uses the message.

Code: 15

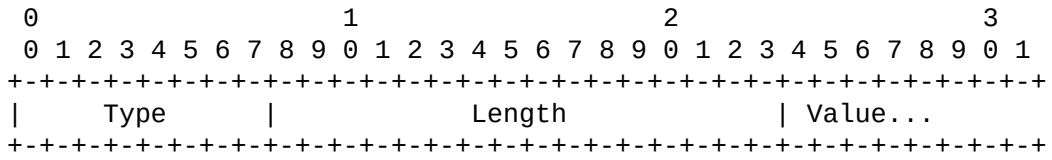
Attributes:

Table 4-16. SA MAP Reject Attributes

Attribute	Contents
SA-Query	Contains addressing information identifying the downstream traffic flow CM requested an SA mapping for
Error-Code	Error code identifying reason for rejection of SA Map Request
Display-String (optional)	Display string containing reason for Map Reject

4.2.2 BPKM Attributes

A summary of the Attribute format is shown below. The fields are transmitted from left to right.



Type

The Type field is one octet. Values of the BPKM Type field are specified below. Note that Type values between 0 and 127 are defined within the Baseline Privacy Specification, values between 128 and 255 are vendor-assigned Attribute Types.

A BPKM server MUST ignore Attributes with an unknown Type.

A BPKM client MUST ignore Attributes with an unknown Type.

BPKM client and server (i.e., CM and CMTS) MAY log receipt of unknown attribute types.

Table 4-17. BPKM Attribute Types

Type	BPKM Attribute
0	Reserved
1	Serial-Number
2	Manufacturer-ID
3	MAC-Address
4	RSA-Public-Key
5	CM-Identification
6	Display-String
7	AUTH-KEY
8	TEK
9	Key-Lifetime
10	Key-Sequence-Number
11	HMAC-Digest
12	SAID
13	TEK-Parameters
14	~~SA-Flag~~ OBSOLETED
15	CBC-IV
16	Error-Code
17	CA-Certificate
18	CM-Certificate
19	Security-Capabilities
20	Cryptographic-Suite
21	Cryptographic-Suite-List
22	BPI-Version
23	SA-Descriptor
24	SA-Type
25	SA-Query
26	SA-Query-Type
27	IP-Address
28-126	Reserved
127	Vendor-Defined
128-255	Vendor-assigned attribute types

Length

The Length field is 2 octets, and indicates the length of this Attribute's Value field, in octets. The length field *does not include* the Type and Length fields.[11] The minimum Attribute Length is 0, the maximum Length is 1487.

Packets containing attributes with invalid lengths SHOULD be silently discarded.

Value

The Value field is zero or more octets and contains information specific to the Attribute. The format and length of the Value field is determined by the Type and Length fields. All multi-octet integer quantities are in network-byte order, i.e., the octet containing the most-significant bits is the first transmitted on the wire.

Note that a "string" does not require termination by an ASCII NULL because the Attribute already has a length field.

11. Note that this is consistent with both the TLV encoding employed in the RF MAC's Extended Header Elements, and the TLV encoding employed for configuration settings in the CM Configuration File [DOCSIS1]. BPKM's TLV encoding differs from that employed by the RADIUS protocol, on which BPKM's basic message structure is based: the Length field of RADIUS attributes includes the Type and Length fields, as well as an attribute's Value field.

The format of the value field is one of five data types.

Table 4-18. Attribute Value Data Types

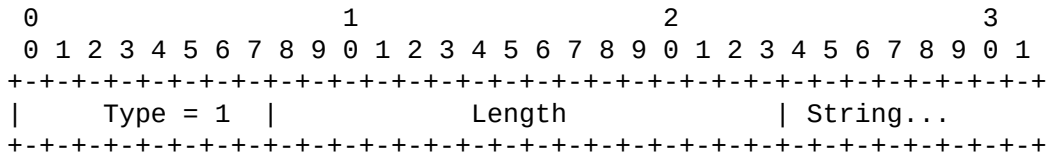
string	0 – 1487 octets
uint8	8-bit unsigned integer
uint16	16-bit unsigned integer
uint32	32-bit unsigned integer
compound	collection of Attributes

4.2.2.1 Serial-Number

Description

This Attribute indicates the serial number assigned by the manufacturer to a cable modem device.

A summary of the Serial-Number Attribute format is shown below. The fields are transmitted from left to right.



Type

1 for Serial-Number

Length

>= 0 and =<255

String

The String field is zero or more octets and contains a manufacturer-assigned serial number.

The manufacturer-assigned serial number MUST be encoded in the ISO 8859-1 character encoding. The characters employed MUST be restricted to the following:

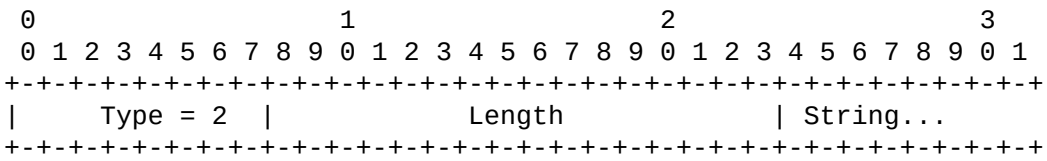
- A-Z (0x41-0x5A)
- a-z (0x61-0x7A)
- 0-9 (0x30-0x39)
- “-” (0xD2)

4.2.2.2 Manufacturer-ID

Description

This Attribute identifies the manufacturer. The identifier is 3 octets long and contains the 3-octet Organizationally Unique Identifier (OUI) assigned to applying organizations by the IEEE [IEEE1]. The first two bits of the 3-octet string are set to zero.

A summary of the Manufacturer-ID Attribute format is shown below. The fields are transmitted from left to right.



Type

2 for Manufacturer-ID

Length

3

String

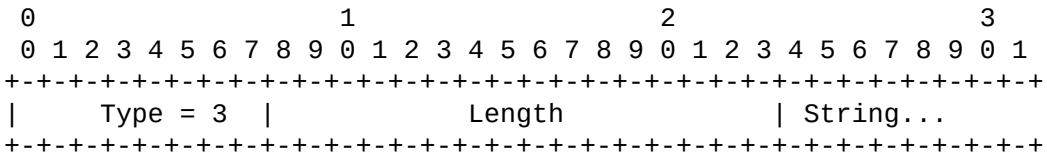
The String field is three octets and contains an IEEE OUI.

4.2.2.3 MAC-Address

Description

This Attribute identifies the IEEE MAC address assigned to the CM. Guaranteed to be unique, it is likely to be used as a cable modem handle/index at the CMTS.

A summary of the MAC-Address Attribute format is shown below. The fields are transmitted from left to right.



Type

3 for MAC-Address

Length

6

String

The String field contains a 6-octet MAC address.

4.2.2.4 RSA-Public-Key

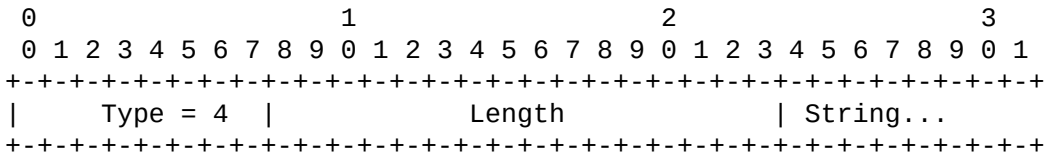
Description

This Attribute is a string attribute containing a DER-encoded RSAPublicKey ASN.1 type, as defined in the RSA Encryption Standard PKCS #1 v2.0 [RSA3].

PKCS #1 v2.0 specifies that an RSA public key consists of both an RSA public modulus and an RSA public exponent; the RSAPublicKey type includes both of these as DER-encoded INTEGER types.

PKCS #1 v2.0 states that the RSA public exponent may be standardized in specific applications, and the document suggests values of 3 or 65537 (F4). Baseline Privacy Plus standardizes on F4 for a public exponent and employs a 1024-bit modulus (Baseline Privacy employed a 768-bit modulus). In order to enable software upgrades of DOCSIS 1.0 hardware to BPI+, the BPI+ implementations MUST support a 768-bit modulus.

A summary of the Public-Key Attribute format is shown below. The fields are transmitted from left to right.



Type

4 for RSA-Public-Key

Length

106, 140, or 270 (length of DER-encoding, using F4 as the public exponent, and a 768-bit, 1024-bit, or 2048-bit public modulus, respectively)

String

DER-encoded RSAPublicKey ASN.1 type

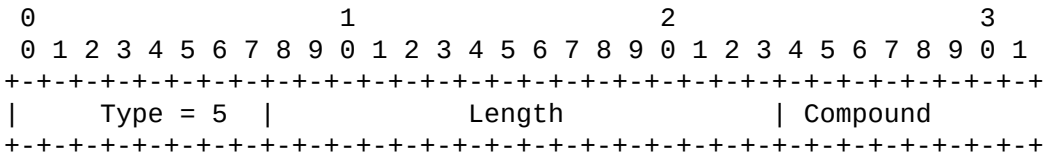
4.2.2.5 CM-Identification

Description

This Attribute is a compound attribute, consisting of a collection of sub-attributes. These sub-attributes contain information that can be used to uniquely identify a cable modem. Sub-attributes MUST include:

- Serial-Number
- Manufacturer-ID
- MAC-Address
- RSA-Public-Key

The CM-Identification MAY also contain optional Vendor-Defined Attributes.



Type

5

Length

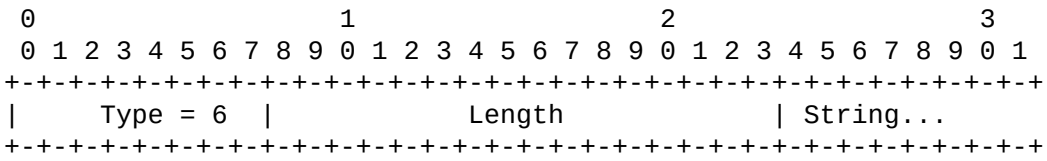
>= 126

4.2.2.6 Display-String

Description

This Attribute contains a textual message. It is typically used to explain a failure response, and might be logged by the receiver for later retrieval by an SNMP manager. Display strings MUST be no longer than 128 bytes.

A summary of the Display-String Attribute format is shown below. The fields are transmitted from left to right.



Type

6 for Display String

Length

>=0 and <= 128

String

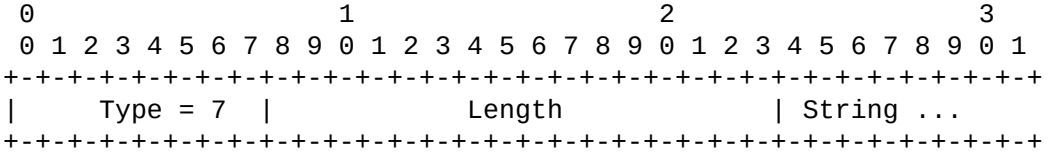
A string of characters. There is no requirement that the character string be null terminated; the length field always identifies the end of the string.

4.2.2.7 AUTH-Key

Description

The Authorization Key is an 20 byte quantity, from which a key encryption key, and two message authentication keys (one for upstream requests, and a second for downstream replies) are derived.

This Attribute contains either a 96 or a 128-octet quantity containing the Authorization Key RSA-encrypted with the CM's 768-bit or 1024-bit RSA public key. Details of the RSA encryption procedure are given in section 7.5. The ciphertext produced by the RSA algorithm will be the length of the RSA modulus, *i.e.*, either 96 or 128 octets.



Type

7 for AUTH-Key

Length

96 or 128

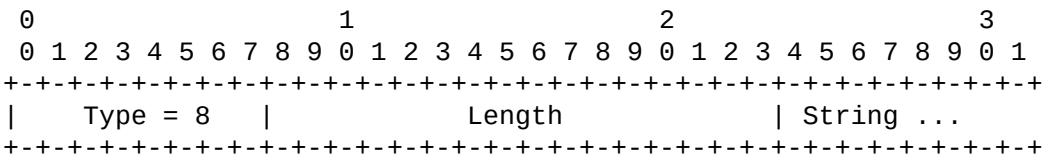
String

96 or 128-octet quantity representing an RSA-encrypted Authorization Key.

4.2.2.8 TEK

Description

This Attribute contains an 8-octet quantity that is a TEK DES key, encrypted with a Key Encryption Key derived from the Authorization Key. TEK keys are encrypted using the Encrypt-Decrypt-Encrypt (EDE) mode of two-key triple DES. See Section 7, Cryptographic Methods, for details.



Type

8 for TEK

Length

8

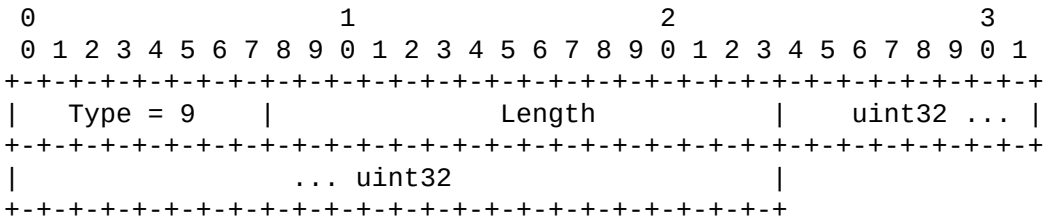
String

64-bit quantity representing a (two-key triple DES EDE mode) encrypted traffic encryption key.

4.2.2.9 Key-Lifetime

Description

This Attribute contains the lifetime, in seconds, of an Authorization Key or TEK. It is a 32-bit unsigned quantity representing the number of remaining seconds that the associated key will be valid.



Type

9 for Key-Lifetime

Length

4

uint32

32-bit quantity representing key lifetime

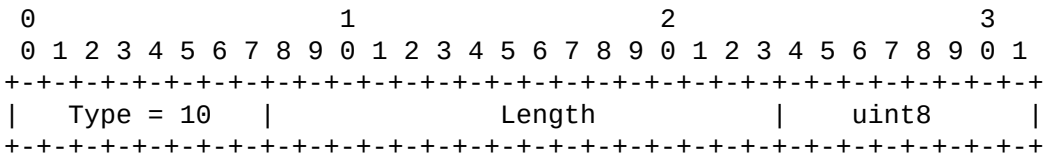
A key lifetime of zero indicates that the corresponding Authorization Key or traffic encryption key is not valid.

4.2.2.10 Key-Sequence-Number

Description

This Attribute contains a 4-bit sequence number for a TEK or Authorization Key. The 4-bit quantity, however, is stored in a single octet, with the high-order 4 bits set to 0.

A summary of the Key-Sequence-Number Attribute format is shown below. The fields are transmitted from left to right.



Type

10 for Key-Sequence-Number

Length

1

uint8

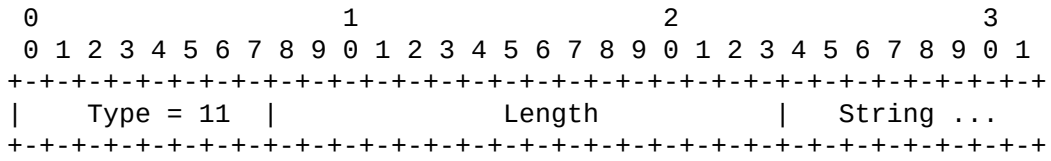
4-bit sequence number

4.2.2.11 HMAC-Digest

Description

This Attribute contains a keyed hash used for message authentication. The HMAC algorithm is defined in [RFC2104]. The HMAC algorithm is specified using a generic cryptographic hash algorithm. Baseline Privacy uses a particular version of HMAC that employs the Secure Hash Algorithm (SHA-1), defined in [FIPS-180-1][FIPS-180-1].

A summary of the HMAC-Digest Attribute format is shown below. The fields are transmitted from left to right.



Type

11 for HMAC-Digest

Length

20-octets

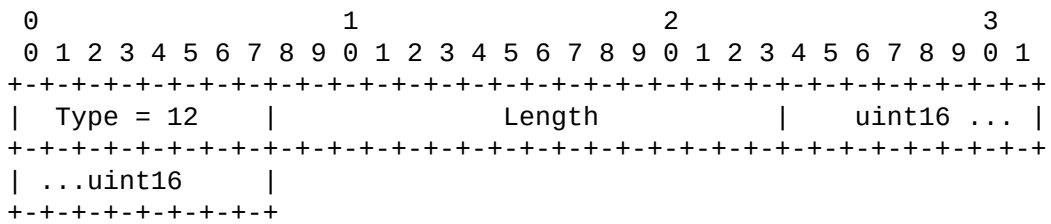
String

A 160-bit (20 octet) keyed SHA hash

4.2.2.12 SAID

Description

This Attribute contains a 14-bit Security Association ID (SAID) used by Baseline Privacy Plus as the security association identifier. The two high-order bits will be set to zero. Note that a CM's primary BPI+ SAID is equal to that CM's Primary SID.



Type

12 for SAID

Length

2

uint16

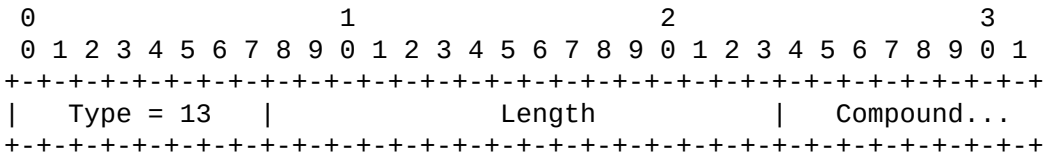
16-bit quantity representing a SAID

4.2.2.13 TEK-Parameters

Description

This Attribute is a compound attribute, consisting of a collection of sub-attributes. These sub-attributes represent all security parameters relevant to a particular generation of a SAID's TEK.

A summary of the TEK-Parameters Attribute format is shown below. The fields are transmitted from left to right.



Type

13 for TEK-Parameters

Length

33

Compound

The Compound field contains the following sub-Attributes:

Table 4-19. TEK-Parameters Sub-Attributes

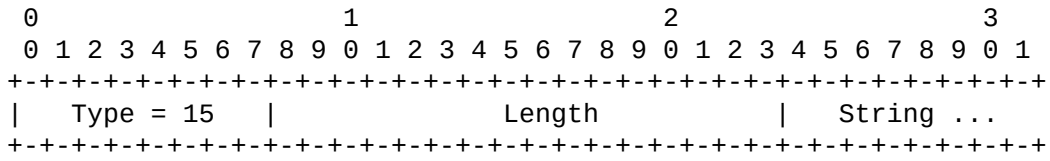
Attribute	Contents
TEK	TEK, encrypted (two-key triple DES-EDE mode) with the KEK
Key-Lifetime	TEK Remaining Lifetime
Key-Sequence-Number	TEK Sequence Number
CBC-IV	Cipher Block Chaining (CBC) Initialization Vector

4.2.2.14 CBC-IV

Description

This Attribute contains a 64-bit (8-octet) value specifying a Cipher Block Chaining (CBC) Initialization Vector.

A summary of the HMAC-Digest Attribute format is shown below. The fields are transmitted from left to right.



Type

15 for CBC-IV

Length

8 octets

String

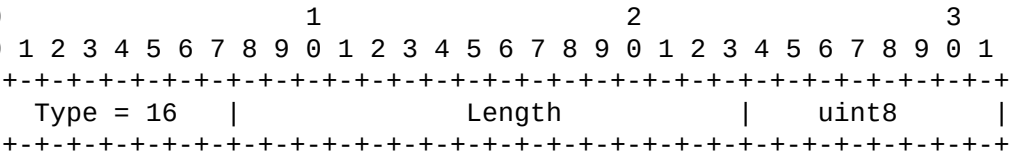
A 64-bit quantity representing a DES-CBC initialization vector.

4.2.2.15 Error-Code

Description

This Attribute contains a one-octet error code providing further information about an Authorization Reject, Key Reject, Authorization Invalid, or TEK Invalid.

A summary of the Error-Code Attribute format is shown below. The fields are transmitted from left to right.



Type

16 for Error-Code

Length

1

uint8

1-octet error code

A CMTS MUST include the Error-Code Attribute in all Authorization Reject, Authorization Invalid, Key Reject, TEK Invalid, and SA-MAP Reject messages. Table 4-20 lists code values for use with this Attribute. The CMTS MUST employ the nonzero error codes list below for SA-MAP Reject messages. The CMTS MAY employ the nonzero error codes listed below for the other BPI+ message types; it MAY, however, return a code value of zero (0). Error code values other than those defined in Table 4-20 MUST be ignored. Returning a code value of zero sends no additional failure information to the CM; for security reasons, this may be desirable.

Table 4-20. Error-Code Attribute Code Values

Error Code	Messages	Description
0	all	no information
1	Auth Reject, Auth Invalid	Unauthorized CM
2	Auth Reject, Key Reject	Unauthorized SAID
3	Auth Invalid	Unsolicited
4	Auth Invalid, TEK Invalid	Invalid Key Sequence Number
5	Auth Invalid	Message (Key Request) authentication failure
6	Auth Reject	Permanent Authorization Failure
7	Map Reject	not authorized for requested downstream traffic flow
8	Map Reject	downstream traffic flow not mapped to BPI+ SAID
9	Auth Reject	Time of day not acquired

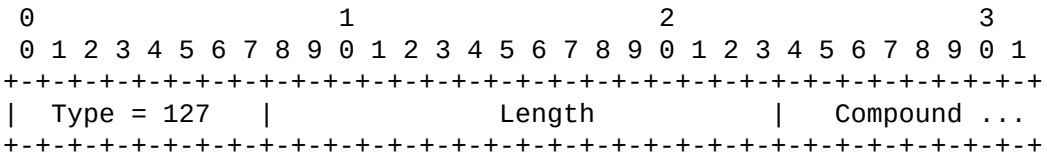
Error code 6, Permanent Authorization Failure, is used to indicate a number of different error conditions affecting the BPKM authorization exchange. These include:

- an unknown manufacturer; i.e., the CMTS does not have the CA certificate belonging to the issuer of a CM certificate
- CM certificate has an invalid signature
- ASN.1 parsing failure during verification of CM certificate
- CM certificate is on the “hot list”
- inconsistencies between certificate data and data in accompanying BPKM attributes
- CM and CMTS have incompatible security capabilities

Their common property is that the failure condition is considered permanent: any re-attempts at authorization would continue to result in Authorization Rejects. Details about the cause of a Permanent Authorization Failure MAY be reported to the CM in an optional Display-String Attribute that may accompany the Error-Code Attribute in Authorization Reject messages. The CMTS SHOULD provide the capability to administratively control whether additional detail is sent to the CM. The CMTS MAY log these Authorization failures, or even trap then to an SNMP manager.

4.2.2.16 Vendor-Defined

The Vendor-Defined Attribute is a compound attribute whose first sub-attribute MUST be the Manufacturer-ID Attribute. Subsequent Attribute(s) are user defined, with Type values as-signed by the vendor identified by the previous Manufacturer-ID Attribute.



Type

127 for Vendor-Defined

Length

>= 6

Compound

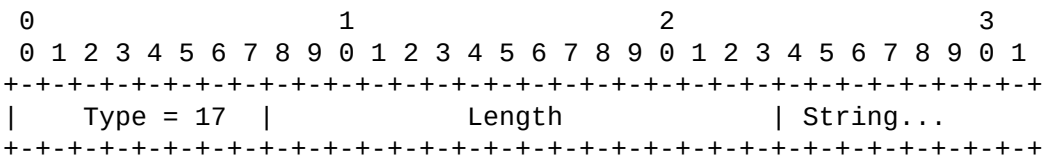
The first sub-attribute MUST be Manufacturer-ID. Subsequent attributes can include both universal Types (i.e., defined within this specification) and vendor-defined Types, specific to the vendor identified in the preceding Manufacturer-ID sub-attribute.

4.2.2.17 CA-Certificate

Description

This Attribute is a string attribute containing an X.509 CA Certificate, as defined in [X.509].

A summary of the CA-Certificate Attribute format is shown below. The fields are transmitted from left to right.



Type

17 for CA-Certificate

Length

Variable. Length MUST NOT cause resulting MAC management message to exceed the maximum allowed size.

String

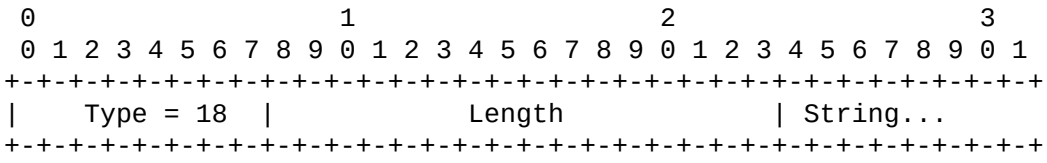
X.509 CA Certificate (DER-encoded ASN.1)

4.2.2.18 CM-Certificate

Description

This Attribute is a string attribute containing a cable modem's X.509 User Certificate, as defined in [X.509].

A summary of the CM-Certificate Attribute format is shown below. The fields are transmitted from left to right.



Type

18 for CM-Certificate

Length

Variable. Length MUST NOT cause resulting MAC management message to exceed the maximum allowed size.

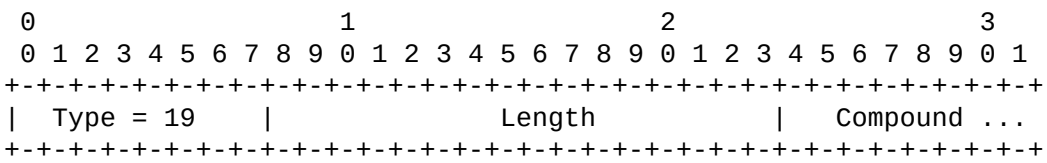
String

X.509 User Certificate (DER-encoded ASN.1)

4.2.2.19 Security-Capabilities

Description

The Security-Capabilities Attribute is a compound attribute whose sub-attributes identify the version of BPI+ a CM supports and the cryptographic suite(s) a CM supports.



Type

19 for Security-Capabilities

Length

>=9

Compound

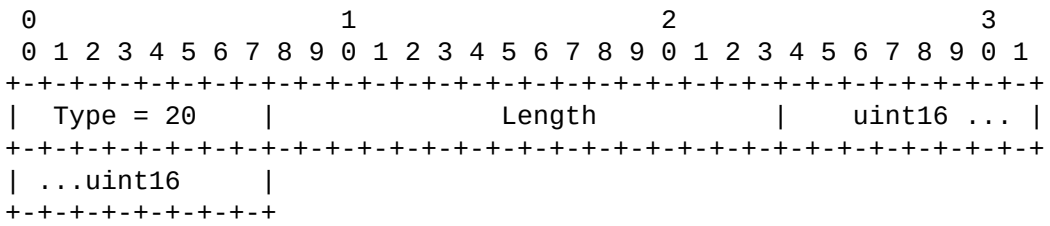
The Compound field contains the following sub-Attributes:

Table 4-21. Security-Capabilities Sub-Attributes

Attribute	Contents
Cryptographic-Suite-List	list of supported cryptographic suites
BPI-Version	version of BPI+ supported

4.2.2.20 Cryptographic-Suite

Description



Type

20 for Cryptographic-Suite

Length

2

Uint16

A 16-bit integer identifying a pairing of a data encryption algorithm (encoded in the left-most, most significant, byte) and a data authentication algorithm (encoded in the right-most, least significant, byte). Currently, 56-bit and 40-bit DES are the only algorithms specified for use within DOCSIS security, and neither are paired with a data authentication algorithm.

Table 4-22. Data Encryption Algorithm Identifiers

Value	Description
0	Reserved
1	CBC-Mode, 56-bit DES
2	CBC-Mode, 40-bit DES
3-255	Reserved

Table 4-23. Data Authentication Algorithm Identifiers

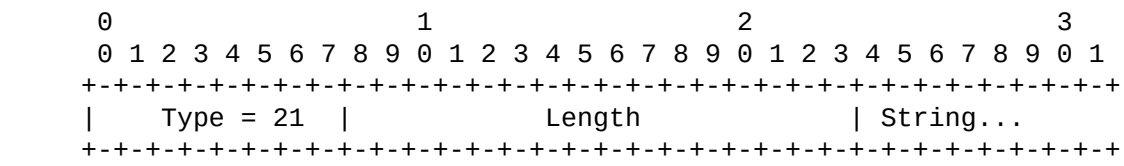
Value	Description
0	No Data Authentication
1-255	Reserved

Table 4-24. Cryptographic-Suite Attribute Values

Value	Description
256 (0x0100 hex)	CBC-Mode 56-bit DES & no data authentication
512 (0x0200 hex)	CBC-Mode 40-bit DES & no data authentication
all remaining values	Reserved

4.2.2.21 Cryptographic-Suite-List

Description



Type

21 for Cryptographic-Suite-List

Length

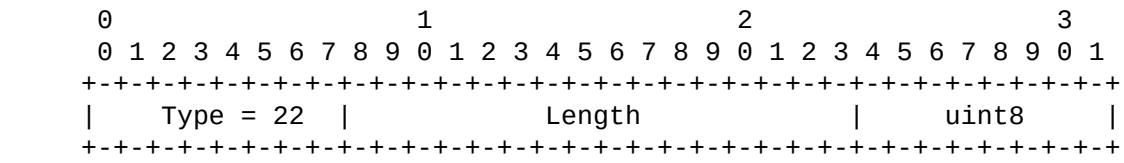
2*n, where n=number of cryptographic suites listed

Uint8

A list of byte pairs identifying a collection of cryptographic suites. Each byte pair represents a supported cryptographic suite, with an encoding identical to the value field of the Cryptographic-Suite Attribute (Section 4.2.2.20). The CMTS MUST NOT interpret the relative ordering of byte pairs in the list as a CM's preferences amongst the cryptographic suites it supports.

4.2.2.22 BPI-Version

Description



Type

22 for BPI-Version

Length

1

Uint8

A 1-octet code identifying a version of Baseline Privacy security.

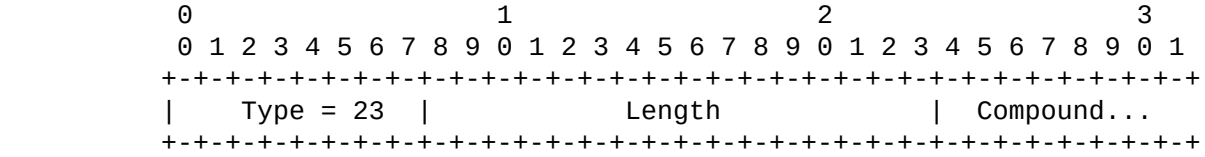
Table 4-25. BPI-Version Attribute Values

Value	Description
0	Reserved
1	BPI+
2-255	Reserved

4.2.2.23 SA-Descriptor

Description

The SA-Descriptor Attribute is a compound attribute whose sub-attributes describe the properties of a BPI+ Security Association. These properties include the SAID, the SA type, and the cryptographic suite employed within the SA.



Type

23 for SA-Descriptor

Length

14

Compound

The Compound field contains the following sub-Attributes:

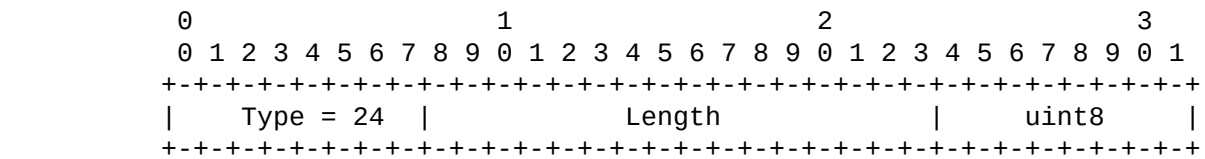
Table 4-26. SA-Descriptor Sub-Attributes

Attribute	Contents
SAID	Security Association ID
SA-Type	Type of SA
Cryptographic-Suite	pairing of data encryption and data authentication algorithms employed within the SA

4.2.2.24 SA-Type

Description

Identifies Type of SA. BPI+ defines three SA types: Primary, Static, Dynamic.



Type

24 for SA-Type

Length

1

Uint8

A 1-octet code identifying the value of SA-type as defined in Table 4-27.

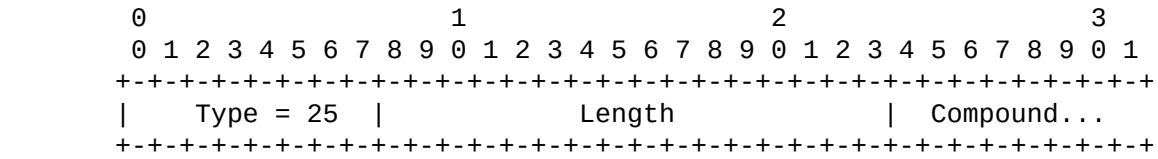
Table 4-27. SA-Type Attribute Values

Value	Description
0	Primary
1	Static
2	Dynamic
3-127	Reserved
128-255	Vendor-specific

4.2.2.25 SA-Query

Description

Compound Attribute used in SA Map Request to specify mapping query arguments. Query arguments include the query type and any addressing attributes particular to that query type - the addressing attributes identify a particular downstream traffic flow that a SA mapping is being requested for. Currently, the only query type specified is IP-Multicast, and the addressing argument associated with that type is an IP group address.



Type

25 for SA-Query

Length

11

Compound

The Compound field contains the following sub-Attributes:

Table 4-28. SA-Query Sub-Attributes

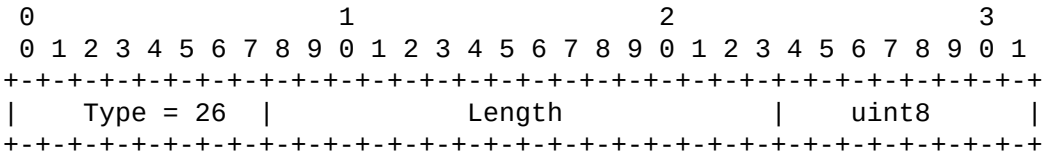
Attribute	Contents
SA-Query-Type	Type of Query
IP-Address	required if SA-Query-Type = IP-Multicast; contains an IP group address whose SA mapping is being requested.

4.2.2.26 SA-Query-Type

Description

This Attribute identifies an IP address used to identify an encrypted IP traffic flow. It is used, for example, to specify an IP multicast group address.

A summary of the IP-Address Attribute format is shown below. The fields are transmitted from left to right.



Type

26 for SA-Query-Type

Length

1

Uint8

A 1-octet code identifying the value of SA-Query-Type as defined in Table 4-29.

Table 4-29. SA-Query-Type Attribute Values

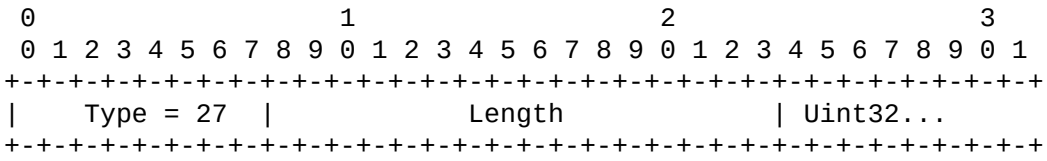
Value	Description
0	Reserved
1	IP Multicast
2-127	Reserved
128-255	Vendor-specific

4.2.2.27 IP-Address

Description

This Attribute identifies an IP address used to identify an encrypted IP traffic flow. It is used, for example, to specify an IP multicast group address.

A summary of the IP-Address Attribute format is shown below. The fields are transmitted from left to right.



Type

27 for IP-Address

Length

4

Uint32

Contains the 32-bit unsigned integer (in network-byte order) representing an IP address.

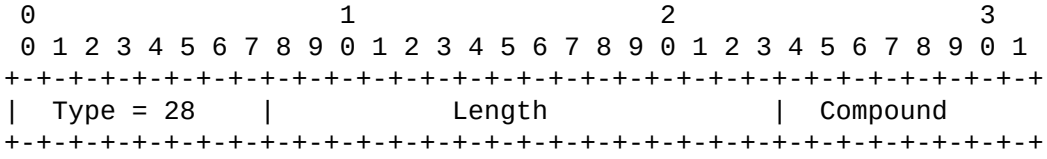
4.2.2.28 Download-Parameters

Description

This attribute is used in the CM Code File defined in the section D.3.1. This attribute is a compound attribute, consisting of a collection of sub-attributes.

Sub-attribute MAY include one or both of the following attribute(s) in this order.

- RSA-Public-Key (zero or one)
- CA-Certificate (zero, one or more)



Type

28

Length

>=0

5 Dynamic SA Mapping

5.1 Introduction

BPI+ *Dynamic Security Associations (Dynamic SAs)*, introduced in Section 2.1.3, are SAs that a CMTS establishes and eliminates, dynamically, in response to its enabling and disabling of specific downstream traffic flows. These traffic flows may be initiated by the actions of:

- a CPE (Customer Premise Equipment) device attached to one of the CMTS's client CMs
- an application server within the head end
- an OSS system
- other unspecified mechanisms

Regardless of what triggers the establishment of a Dynamic SA within the CMTS, client CMs need a mechanism for learning the mapping of a particular BPI+-protected downstream traffic flow to that flow's dynamically assigned BPI+ Security Association (and that SA's corresponding SAID).

The SA Mapping state machine, defined in this section, specifies how cable modems query a CMTS for the mapping of downstream traffic flows to Dynamic SAs. The state machine controls the transmission of SA Map Request messages to a CMTS.

DOCSIS 1.1 currently employs Dynamic SAs for a single service type: encrypting, and thus restricting access to, downstream IP multicast traffic. A CMTS can establish or eliminate Dynamic SAs in response to changes in IP group membership of downstream CPE devices. DOCSIS 1.1's IGMP management mechanisms ([DOCSIS1], Section 3.3.1, "Requirements for IGMP Management", or [DOCSIS9], Section 5.3.1, "Requirements for IGMP Management", can trigger the establishment of Dynamic SAs in the CMTS. IGMP management mechanisms in the CM MUST trigger BPI+ Map Request messages that query the CMTS for the mapping of an IP multicast group address to an SA.[12]

BPI+'s SA mapping mechanism MAY map an IP multicast group to a static SA, or even to a particular CM's Primary SA; a CMTS's response to a mapping request may return any of the three types of SAs. The SA mapping mechanism, however, is the only mechanism by which a CM can learn the identity of Dynamic SAs.

Section 5.4 will discuss in greater detail the particular use of the SA mapping mechanism to support the mapping of IP multicast traffic to Dynamic SAs. In the following two sections, however, we focus on the more general SA mapping mechanism.

Note that future enhancements to the DOCSIS service specifications may define additional applications of Dynamic SAs.

12. Reference updated per BPI-N-02098 by RKV on 8/23/02.

5.2 Theory of Operation

BPI+ defines three new BPKM messages to support CM querying for SA mappings: the SA Map Request, the SA Map Reply, and the SA Map Reject. A CM sends a Map Request to its CMTS to request the mapping of a known downstream flow to a SA. The Map Request carries BPI+ data attributes identifying the requesting CM and the downstream traffic flow whose SA mapping is being requested.

The CMTS may respond to a Map Request with *either*

- a Map Reply, providing the CM with the requested SA mapping *or*
- a Map Reject, signaling to the CM that either (1) the CM is not authorized to receive the traffic flow identified in the Map Request or (2) the requested traffic flow is not mapped to a BPI+ SA

If the CM does not receive any of the above responses within a configurable retry timeout period, it re-sends the Map Request. If no response is received after a configurable maximum number of retries, the CM gives up.

If the CM receives a Map Reject, it ceases all further attempts to obtain the mapping. In the case where access to the downstream traffic flow is mapped to a BPI+ SA, and the requesting CM is not authorized access for that SA, the CM and its attached CPE device will be denied access because the CM cannot obtain keying material needed to decrypt the downstream traffic flows encrypted under that SA. E.g., the user may be requesting a premium service that he or she is not subscribed to. In the case where the requested traffic flow is not encrypted (i.e., it is not mapped to a SA), the unencrypted traffic will simply be forwarded to the attached CPE device. E.g., the CM makes an SA-MAP request for the All-Hosts multicast address. Since multicast packets addressed to the All-Hosts multicast address are necessary for the proper operation of IGMP, there is no need to encrypt these packets.

If the CM receives a Map Reply identifying the BPI+ SA associated with the requested downstream traffic flow, the CM launches a TEK state machine for the SA, provided both (1) the CM is not already running a TEK state machine for that SA and (2) the CM supports the cryptographic suite identified in the Map Reply along with the Security Association ID (SAID) value.

The CM may already be running a TEK state machine if the mapped SA is:

- a Dynamic SA mapped to another protected traffic flow the CM already has access to
- the requesting CM's Primary SA
- a Static SA the CM learned about in a previously received Authorization Reply

Note that a CMTS MAY assign multiple traffic flows (i.e., IP multicast addresses) to the same SA. If more than one downstream traffic flow is being encrypted under the same Dynamic SA, a CM may already be running a TEK state machine for the SA identified in the Map Reply. Note also that the SA mapping returned in the Map Reply need not be a Dynamic SA: the requested traffic flow may be mapped to the CM's Primary SA or a Static SA.[13]

The Map Reply includes an SA-Descriptor Attribute which identifies both a SAID and the cryptographic suite employed within the SA. As is the case with Static SAs, the selection of a Dynamic

13. modified per bpi-n-02011, 07/25/02, ab

SA's cryptographic suite is typically made independent of the requesting CM's cryptographic capabilities. Thus, a CMTS MAY respond to a Map Request with an SA (either Static or Dynamic) that employs a cryptographic suite the requesting CM does not support. The CM MUST NOT start TEK state machines for Static or Dynamic SAs whose cryptographic suites the CM does not support. (A Primary SA, however, must employ a cryptographic suite that is supported by the CM to which the SA belongs.)

The TEK state machine controls the retrieval of the mapped SA's keying material. The CM will send Key Requests for the SA; the CMTS may respond to these key requests with:

- a Key Reply, providing the CM with the requested keying material
- a Key Reject, signaling to the CM it is not authorized for the requested mapped SAID
- an Authorization Invalid, signaling to the CM that authentication of the Key Request message failed

The receipt of a Key Reject forces the termination of the TEK state machine.

Note that there are two mechanisms for the CMTS to tell a client CM it is not authorized to access a particular traffic flow: responding to a Map Request with a Map Reject, and responding to a Key Request with a Key Reject. It is implementation dependent whether a CMTS checks a CM's authorization status prior to responding to a Map Request. By doing the check during the mapping exchange, a CM will be prevented from needlessly launching a TEK state machine and sending a Key Request for a SAID it is not authorized for.

5.3 SA Mapping State Model

The SA Mapping state model specifies the mechanism by which a CM learns the mapping of a traffic flow to a dynamic SA.

A state machine is started when, within the CM, an event, external to the SA Mapping State Model, triggers the need for a traffic-flow-to-SA mapping (for example, when a CM installs the permit filters for an IP multicast group as a result of the CM's IGMP management mechanisms). This external event generates an internal "Map" event in the SA Mapping state machine.

The state machine is terminated if the CM receives no response after sending the maximum number of retries, or when the CM determines it no longer requires the mapped SA's keying material. In this later case, an external event generates an internal "Unmap" event in the SA Mapping state machine, forcing it's termination. Thus, the state machine can be used not only to obtain the required mapping information, but also to track the period over which an external application using the SA Mapping mechanism (e.g., IGMP management) requires that mapping. Linkage of the Unmap event to an external event, and hence implementation of the Unmap event, is OPTIONAL.

As with the BPI+ Authorization and TEK state machines, the SA Mapping state machine is presented in graphical format, as a state flow model (Figure 5-1), and in a tabular format, as a state transition matrix (Table 5-1). And as with the previously defined state machines, the state transition matrix MUST be used as the definitive specification of protocol actions associated with each state transition.

If, through the SA Mapping mechanism, a CM learns it requires access to a dynamic SA's keying material, it must establish a TEK state machine for that dynamic SA. While the Authorization state

machine controls the establishment and termination of TEK state machines associated with the Primary and any Static SAIDs, it does not control the establishment and termination of TEK state machines associated with Dynamic SAs. CMs MUST implement the necessary logic to establish and terminate TEK state machines for the Dynamic SAs learned of through the SA Mapping mechanism. The BPI+ specification, however, does not define how CMs should manage their Dynamic SA's TEK state machines.

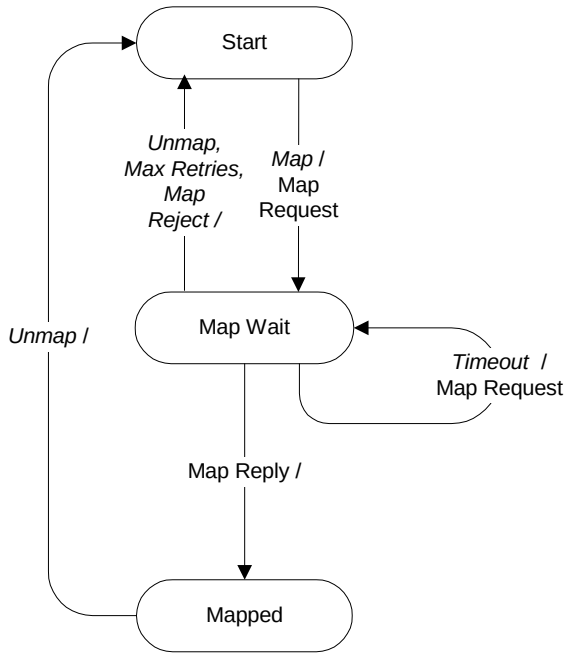


Figure 5-1. SA Mapping State Machine Flow Diagram

Table 5-1. Dynamic SAID State Transition Matrix

State *Event or Rcvd Message*	(A) Start	(B) Map Wait	(C) Mapped
(1) *Map*	Map Wait		
(2) *Unmap*		Start	Start
(3) *Map Reply*		Mapped	
(4) *Map Reject*		Start	
(5) *Timeout*		Map Wait	
(6) *Max Retries*		Start	

5.3.1 States

5.3.1.1 Start

The initial state of the finite state machine.

5.3.1.2 Map Wait

The CM has sent the CMTS a Map Request and is waiting for a response.

5.3.1.3 Mapped

The CM has received a Map Reply, learned the requested SA mapping.

5.3.2 Messages

5.3.2.1 SA Map Request (Map Request)

Sent by CM to CMTS to request a SA mapping.

5.3.2.2 SA Map Reply (Map Reply)

Positive CMTS response to Map Request containing the requested SA mapping.

5.3.2.3 SA Map Reject (Map Reject)

Negative CMTS response to CM's Map Request; signals to the CM that either (1) the CM is not authorized access to the traffic flow identified in the Map Request or (2) the requested traffic flow is not mapped to a BPI+ SA.

5.3.3 Events

5.3.3.1 Map

Triggers the start of the SA Mapping state machine. The Map event is linked to a CM event external to the BPI+ protocol.

5.3.3.2 Unmap

Triggers the termination of the SA Mapping state machine. The Unmap event is linked to a CM event external to the BPI+ protocol. Implementation of the Unmap event is OPTIONAL.

5.3.3.3 Map Reply

Cable modem receives a SA Map Reply message.

5.3.3.4 Map Reject

Cable modem receives a SA Map Reject message.

5.3.3.5 Timeout

Cable modem has timed out waiting for a response to an outstanding SA Map Request message.

5.3.3.6 Max Retries

Cable modem has sent the maximum number of retries and not received a response.

5.3.4 Parameters

All configuration parameter values are specified in the TFTP-downloaded parameter file (see Appendix A: TFTP Configuration File Extensions).

5.3.4.1 SA Map Wait Timeout

Timeout period between sending SA Map Request messages from SA Wait state. See A.1.1.1.8.

5.3.4.2 SA Map Max Retries

Maximum number of times CM retries SA Map Request before giving up.

5.3.5 Actions

Actions taken in association with state transitions are listed by <event/rcvd message> - <state> below:

1-A Start (*Map*) → Map Wait

- send SA Map Request
- set Map Request retry timer to SA Map Wait Timeout
- set Map Retry Count to 0

2-B Map Wait (*Unmap*) → Start

- clear Map Request retry timer
- terminate SA Mapping state machine

2-C Mapped (*Unmap*) → Start

- terminate SA Mapping state machine

3-B Map Wait (*Map Reply*) → Mapped

- clear Map Request retry timer

4-B Map Wait (Map Reject) → Start

- clear Map Request retry timer
- terminate SA Mapping state machine

5-B Map Wait (*Timeout*) → Map Wait

- send Map Request
- set Map Request retry timer to SA Map Wait Timeout
- increment Map Retry Count
- if Map Retry Count > SA Map Max Retries, generate Max Retries event

6-B Map Wait (*Max Retries*) → Start

- terminate SA Mapping state machine

5.4 IP Multicast Traffic and Dynamic SAs

DOCSIS 1.1 [DOCSIS1] specifies rules for the management of IGMP traffic in the CM and CMTS. These rules are designed to control the flow of IP multicast traffic across the cable network and across the CM/CPE interface so that

- a CMTS only forwards downstream traffic associated with an IP multicast group if a CPE device, attached to one of the CMTS's client CMs, is a member of that group *and*
- a CM only forwards across its CPE interface downstream traffic associated with an IP multicast group if an attached CPE device is a member of that group.

BPI+, operating in conjunction with the DOCSIS 1.1 RFI, controls access to IP multicast traffic flows by encrypting them and controlling the distribution of the multicast keying material required to decrypt the flows.

A CMTS may map downstream multicast flows to any of BPI+'s three classes of Security Associations: Primary, Static or Dynamic. If an IP multicast group's traffic is mapped to a Primary SA, only the single CM belonging to that SA can access that group. If mapped to a Static or Dynamic SA, then multiple CMs may access that group, although a CMTS may restrict a Static or Dynamic SA to a single CM.

When a DOCSIS 1.1 CM enables downstream forwarding of an IP multicast group (in response to receiving a Membership Report on its CPE interface), the CM MUST determine whether the IP multicast group's downstream traffic is encrypted and the BPI+ SAID associated with the encrypted downstream multicast flow. Once the CM has the associated SAID, it can launch a TEK state machine to retrieve the SA's keying material.

The CM uses BPI+'s SA Mapping mechanism to request from its CMTS the SA mapping for an IP multicast group it just joined. The SA Mapping state machine's *Map* event is triggered by the enabling of RF-to-CPE forwarding of the IP multicast group in the CM (see section 3.3.1.2 and Appendix L of [DOCSIS1]). A SA Map Reply informs the CM that the joined group is mapped to a BPI+ SA. If the group is mapped to the CM's Primary SA, the CM already has the required keying material. If the group is mapped to a Static or Dynamic SA, the CM determines whether it is already running a TEK state machine for that SA; if not it starts one.

The SA Mapping state machine defines an OPTIONAL Unmap event which terminates the SA Mapping state machine and MAY be used to indicate the CM no longer requires the mapped SA's keying material. In the case of the mapping of IP multicast traffic to a SA, the Unmap event could indicate that the CM has removed all IP multicast permit filters associated with IP multicast groups mapped to the SA in question. Thus, the SA Mapping state machine MAY be used to track the necessity of a CM to maintain keying material for a Dynamic SA mapped to one or more IP multicast groups.

TEK state machines corresponding to Primary and Static SAIDs are stopped according to the termination conditions defined in the Authorization and TEK state machines.

6 Key Usage

6.1 CMTS

After a CM completes DOCSIS MAC Registration, it initiates an Authorization exchange with its CMTS. The CMTS's first receipt of an Authorization Request message from the unauthorized CM initiates the activation of a new Authorization Key (AK), which the CMTS sends back to the requesting CM in an Authorization Reply message. This AK will remain active until it expires according to its predefined lifetime, *Authorization Key Lifetime*, a CMTS system configuration parameter (see Appendix A.2).

The CMTS MUST use keying material derived from the CM's Authorization Key for

- verifying the HMAC-Digest in Key Requests received from that CM
- encrypting (EDE mode two-key triple DES) the TEK in the Key Replies it sends to that CM (TEK is a sub-attribute of a Key Reply's TEK-Parameters Attribute)
- calculating the HMAC-Digests it writes into Key Replies, Key Rejects and TEK Invalids sent to that CM

The CMTS must always be prepared to send a CM an AK upon request. The CMTS MUST be able to support up to two simultaneously active AKs for each client CM. The CMTS has two active AKs during an Authorization Key transition period; the two active keys have overlapping lifetimes.

An Authorization Key transition period begins when the CMTS receives an Authorization Request from a CM and the CMTS has a single active AK for that CM. In response to this Authorization Request, the CMTS activates a second AK, which it sends back to the requesting CM in an Authorization Reply. The CMTS MUST set the active lifetime of this second AK to be the remaining lifetime of the first AK, plus the predefined *Authorization Key Lifetime*; thus, the second, "newer" key will remain active for one *Authorization Key Lifetime* beyond the expiration of the first, "older" key. The key transition period will end with the expiration of the older key. This is depicted in the top half of Figure 6-1.

The Authorization Key lifetime a CMTS reports in a Authorization reply MUST reflect, as accurately as an implementation permits, the remaining lifetimes of AK at the time the reply message is sent.

As long as the CMTS is in the midst of a CM's Authorization Key transition period, and thus is holding two active Authorization Keys for that CM, it will respond to Authorization Requests with the newer of the two active keys. Once the older key expires, an Authorization Request will trigger the activation of a new AK, and the start of a new key transition period.

If a CM fails to reauthorize before the expiration of its most current AK, the CMTS will hold no active Authorization keys for the CM and will consider the CM *unauthorized*. A CMTS MUST remove from its keying tables all TEKs associated with an unauthorized CM's Primary SA.

A CMTS MUST use a CM's active AK(s) to verify the HMAC-digest in Key Requests received from the CM. If a CMTS receives a Key Request while in an AK transition period, and the accompanying AK Key Sequence Number indicates the Request was authenticated with the newer of the two AKs, the CMTS identifies this as an *implicit acknowledgment* that the CM has obtained the newer of the CM's two active AKs.

A CMTS MUST use an active AK when calculating HMAC-Digests in Key Replies, Key Rejects and TEK Invalids, and when encrypting the TEK in Key Replies. When sending Key Replies, Key Rejects or TEK Invalids within a key transition period (i.e., when two active AKs are available), if the newer key has been implicitly acknowledged, the CMTS MUST use the newer of the two active AKs; if the newer key has not been implicitly acknowledged, the CMTS MUST use the older of the two active AKs.

The upper half of Figure 6-1 illustrates the CMTS's policy regarding its use of AKs.

The CMTS MUST maintain two sets of active traffic encryption keys (and their associated CBC initialization vectors) per SAID. They correspond to two successive generations of keying material, and have overlapping lifetimes. The newer TEK MUST have a key sequence number one greater than (modulo 16) that of the older TEK. Each TEK becomes active half way through the lifetime of its predecessor, and expires half way through the lifetime of its successor. Once a TEK's lifetime expires, the TEK becomes inactive and MUST no longer be used.

The CMTS transitions between the two active TEKs differently depending on whether the TEK is used for downstream or upstream traffic. For each of its SAIDs, the CMTS MUST transition between active TEKs according to the following rules:

- The CMTS MUST use the older of the two active TEKs for encrypting downstream traffic. At expiration of the older TEK, the CMTS MUST immediately transition to using the newer TEK for encryption.
- For decryption of upstream traffic, a transition period is defined that begins once the CMTS has sent the newer TEK to a CM within a Key Reply Message. The upstream transition period begins from the time the CMTS sends the newer TEK in a Key Reply Message and concludes once the older TEK expires. While in the transition period, the CMTS MUST be able to decrypt upstream frames using either the older or newer TEK.

Note that the CMTS encrypts with a given TEK for only the second half of that TEK's total lifetime. The CMTS is able, however, to decrypt with a TEK for the TEK's entire lifetime.

The KEY_SEQ field in the Baseline Privacy EH element identifies which of the two TEKs the upstream frame's packet data was encrypted with. The TOGGLE bit in the Privacy EH element, which is equal to the least significant bit of the KEY_SEQ field, can be used by the CMTS in identifying the encrypting TEK.

The upper half of Figure 6-2 illustrates this CMTS's management of a BPI+ Security Association's TEKs.

The CMTS is responsible for maintaining keying information for both primary and multicast SAIDs in the above manner. The Baseline Privacy Key Management protocol defined in this specification describes a mechanism for synchronizing this keying information between a CMTS and its client CMs. It is the responsibility of the CM to update its keys in a timely fashion; the CMTS will transition to a new downstream encryption key regardless of whether a client CM has retrieved a copy of that TEK.

The Key Replies sent by a CMTS contain TEK parameters (the TEK itself, a key lifetime, a key sequence number and a CBC IV) for the two active TEKs. The key lifetimes a CMTS reports in a Key Reply MUST reflect, as accurately as an implementation permits, the remaining lifetimes of these TEKs at the time the Key Reply message is sent.

6.2 Cable Modem

The CM is responsible for sustaining authorization with its CMTS and maintaining an active Authorization Key. A CM MUST be prepared to use its two most recently obtained AKs.

AKs have a limited lifetime and must be periodically refreshed. A CM refreshes its Authorization Key by re-issuing an Authorization Request to the CMTS. The Authorization state machine (Section 4.1.2) manages the scheduling of Authorization Requests for refreshing AKs.

A CM's Authorization state machine schedules the beginning of reauthorization a configurable length of time (the *Authorization Grace Time*) before the CM's latest AK is scheduled to expire. The Authorization Grace Time is configured to provide a CM with an authorization retry period that is sufficiently long to allow for system delays and provide adequate time for the CM to successfully complete an Authorization exchange before the expiration of its most current AK.

Note that the CMTS does not require knowledge of the Authorization Grace Time. The CMTS, however, tracks the lifetime of its Authorization Keys and MUST deactivate a key once it has expired.

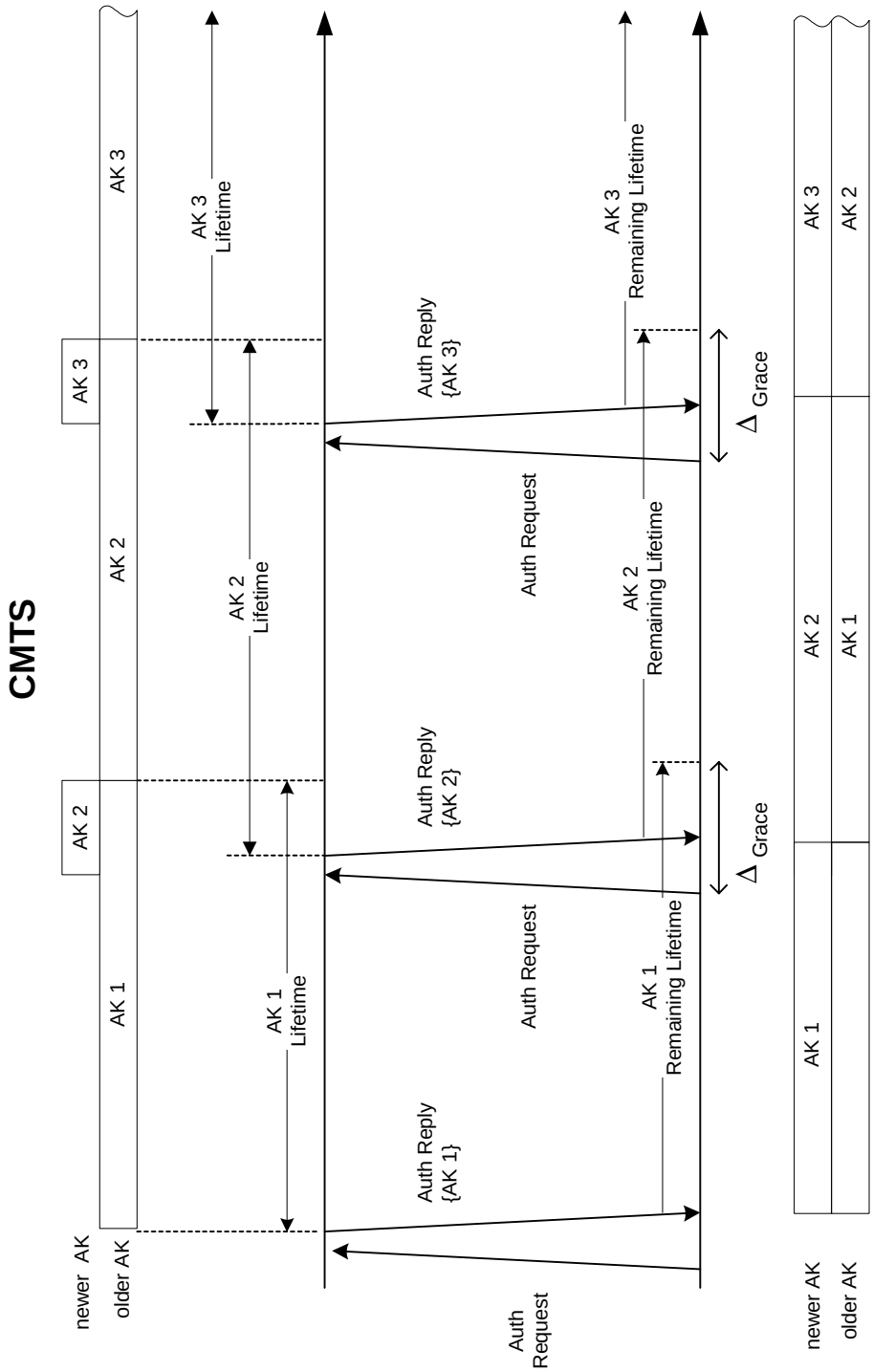


Figure 6-1. Authorization Key Management in CMTS and CM

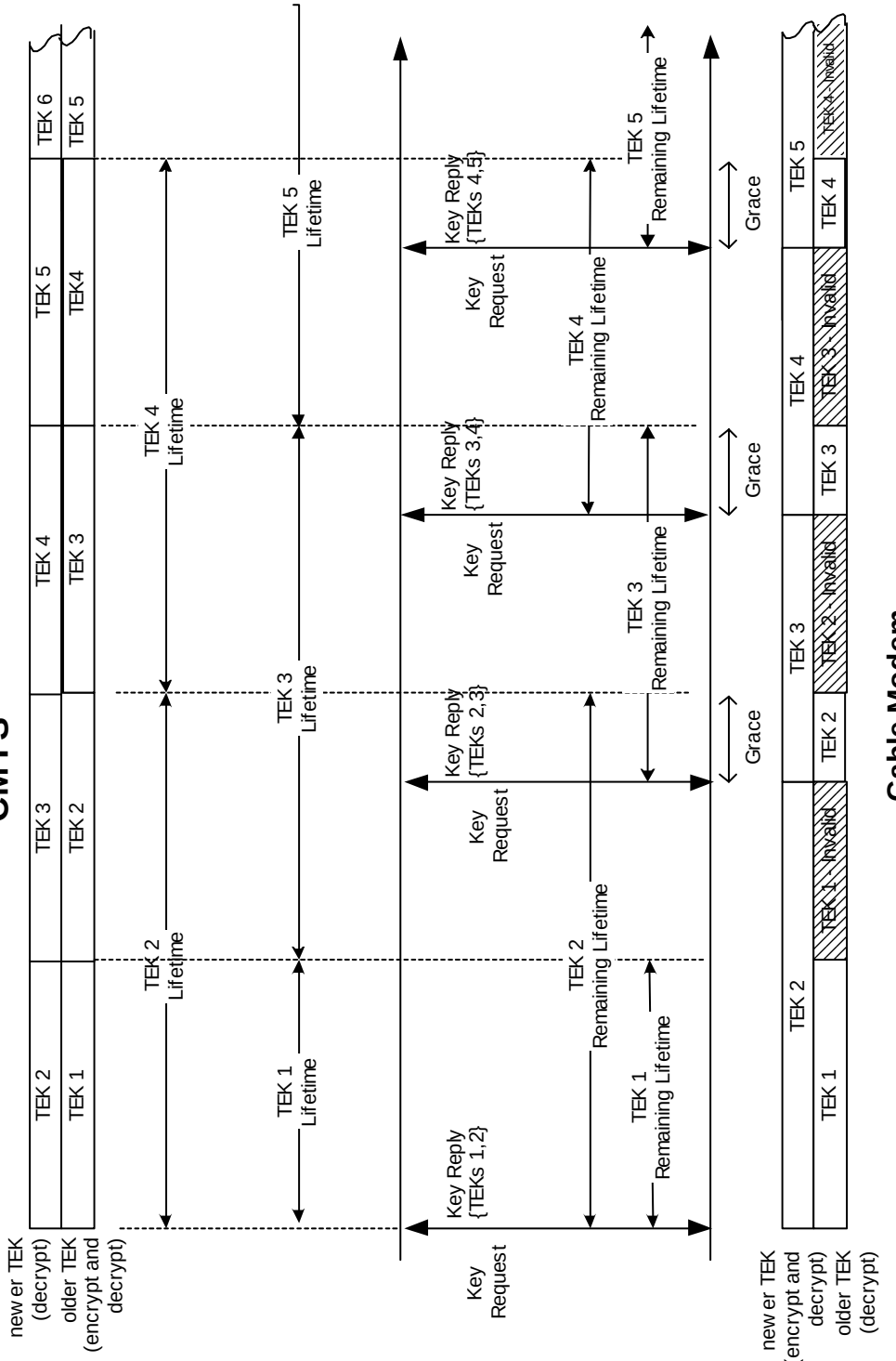


Figure 6-2. TEK Management in CMTS and CM

A cable modem MUST use the newer of its two most recent Authorization Keys when calculating the HMAC-Digests it attaches to Key Requests. It MUST be able to use either of its two most recent AKs to authenticate Key Replies, Key Rejects or TEK Invalids, and to decrypt a Key Reply's encrypted TEK. The CM uses the accompanying AK Key Sequence Number to determine which of the two AKs to use.

The lower half of Figure 6-1 illustrates a CM’s maintenance and usage of its Authorization Keys.

A CM MUST be capable of maintaining two successive sets of traffic keying material per authorized SAID. Through operation of its TEK state machines, a CM attempts to always maintain a SAID’s two most recent sets of traffic keying material.

For each of its authorized SAIDs, the cable modem:

- MUST use the newer of its two TEKs to encrypt newly received upstream traffic. Traffic already queued up MAY use either TEK (in no specific order) for a brief period of time covering the transition from the old to the new key.
- MUST be able to decrypt downstream traffic encrypted with either of the TEKs

The KEY_SEQ field in the Baseline Privacy EH element identifies the key sequence number of the TEK used to encrypt the PDU’s packet data. The TOGGLE bit in the Privacy EH element, which is equal to the least significant bit of the KEY_SEQ field, assists in distinguishing between two successive key generations.

6.3 Authentication of DOCSIS v1.1 Dynamic Service Requests

If a DOCSIS 1.1 CM is configured to run BPI+, the DOCSIS v1.1 RFI specification [DOCSIS1] requires CM and CMTS to include HMAC-Digests in all Dynamic Service Addition Requests (DSA-REQs), Dynamic Service Change Requests (DSC-REQs) and Dynamic Service Deletion Requests (DSD-REQs) they send to one another.

These Dynamic Service HMAC-Digests are keyed with the BPI+ message authentication keys, i.e., the message authentication keys derived from the BPI+ Authorization Key. CMs and CMTSs MUST use the current message authentication keys when generating and validating the HMAC-Digests contained in Dynamic Service Requests.

7 Cryptographic Methods

This section specifies cryptographic algorithms and key sizes BPI+ uses.

7.1 Packet Data Encryption

Baseline Privacy Plus MUST use the Cipher Block Chaining (CBC) mode of the US Data Encryption Standard (DES) algorithm [FIPS-46, FIPS-46-1, FIPS-74, FIPS-81] to encrypt the Packet Data field RF MAC Packet Data PDU Frames and the Fragmentation Payload and Fragmentation CRC Fields in MAC Fragmentation Frames.

BPI+ implementations running on DOCSIS 1.1 hardware (the predominant hardware/software configuration) MUST support 56-bit DES and MAY support 40-bit DES.

BPI+ supports 40-bit DES principally to permit interoperability with 40-bit DOCSIS 1.0 hardware upgraded to run BPI+. 40-bit DES is identical to 56-bit DES, with the exception that 16 bits of the 56-bit DES key are set to known, fixed values. If a CM or CMTS is running the optional 40-bit DES, it MUST mask off (to zero) the sixteen left-most bits of any 56-bit DES key prior to running encryption/ decryption operations. Note that the masked bits are the sixteen left-most bits that would be present AFTER the removal of every eighth bit from the 64-bit TEK (i.e., the so-called parity bits). DOCSIS 1.1 and 56-bit DOCSIS 1.0 hardware running BPI+ MAY implement 40-bit DES key masking in software.

CBC MUST be initialized with an initialization vector that is provided, along with other SAID key material, in a CMTS's Key Reply. Chaining is done block to block within a frame and reinitialized on a frame basis in order to make the system more robust to potential frame loss.

Residual termination block processing MUST be used to encrypt the final block of plaintext when the final block is less than 64 bits. Given a final block having n bits, where n is less than 64, the next-to-last ciphertext block is DES encrypted a second time, using the ECB mode, and the least significant n bits of the result are exclusive ORed with the final n bits of the payload to generate the short final cipher block. In order for the receiver to decrypt the short final cipher block, the receiver DES encrypts the next-to-last ciphertext block, using the ECB mode, and exclusive ORs the left-most n bits with the short final cipher block in order to recover the short final cleartext block. This encryption procedure is depicted in Figure 9.4 (pg. 195) of [SCHNEIER].

In the special case when the frame's to-be-encrypted plaintext is less than 64 bits, the initialization vector MUST be DES encrypted, and the left-most n bits of the resulting ciphertext corresponding to the number of bits of the payload MUST be exclusive ORed with the n bits of the payload to generate the short cipher block.[14]

14. This method of encrypting short payloads is vulnerable to attack: EXORing two sets of ciphertext encrypted in the above manner under the same set of keying material will yield the EXOR of the corresponding sets of plaintext. In the case of Packet Data PDUs Frame's, however, this is not an issue since all Frame's carrying protected user data will contain at least 20 bytes of IP header. In the case of Fragmentation Frames, a short frame carrying less than 8 bytes (64 bits) of ciphertext is possible; however, the final four bytes would be the encrypted Fragmentation CRC, and the three or fewer bytes before the encrypted Fragmentation CRC would be the encrypted Packet Data CRC.

7.2 Encryption of TEK

The CMTS encrypts the value fields of the TEK in the Key Reply messages it sends to client CMs. This field is encrypted using two-key triple DES in the encrypt-decrypt-encrypt (EDE) mode [SCHNEIER]:

encryption: $C = E_{k1}[D_{k2}[E_{k1}[P]]]$

decryption: $P = D_{k1}[E_{k2}[D_{k1}[C]]]$

P = Plaintext 64-bit TEK

C = Ciphertext 64-bit TEK

k1 = left-most 64 bits of the 128-bit KEK

k2 = right-most 64 bits of the 128-bit KEK

E[ ] = 56-bit DES ECB (electronic code book) mode encryption

D[ ] = 56-bit DES ECB decryption

Section 7.4 below describes how the KEK is derived from the Authorization key.

7.3 HMAC-Digest Algorithm

The keyed hash employed by the HMAC-Digest Attribute MUST use the HMAC message authentication method [RFC 2104] with the SHA-1 hash algorithm [FIPS-180-1].

Upstream and downstream message authentication keys are derived from the Authorization Key (see Section 7.4 below for details).

7.4 Derivation of TEKs, KEKs and Message Authentication Keys

The CMTS generates Authorization Keys, TEKs and IVs. A random or pseudo-random number generator MUST be used to generate Authorization Keys and TEKs. A random or pseudo-random number generator MAY also be used to generate IVs; regardless of how they are generated, IVs MUST be unpredictable. [RFC1750] provides recommended practices for generating random numbers for use within cryptographic systems.

[FIPS-81] defines DES keys as 8-octet (64-bit) quantities where the seven most significant bits (i.e., seven left-most bits) of each octet are the independent bits of a DES key, and the least significant bit (i.e., right-most bit) of each octet is a parity bit computed on the preceding seven independent bits and adjusted so that the octet has odd parity.

The keying material for two-key triple DES consists of two distinct (single) DES keys.

BPKM does not require odd parity. The BPKM protocol generates and distributes 8-octet DES keys of arbitrary parity, and it requires that implementations ignore the value of the least significant bit of each octet.

A key encryption key (KEK) and two message authentication keys are derived from a common Authorization Key. The following defines how these keys are derived:

KEK is the Key Encryption Key used to encrypt Traffic Encryption Keys.

HMAC_KEY_U is the message authentication key used in upstream Key Requests

HMAC_KEY_D is the message authentication key used in downstream Key Replies, Key Rejects and TEK Invalids.

SHA(x|y) denotes the result of applying the SHA function to the concatenated bit strings x and y.

Truncate(x,n) denotes the result of truncating x to its left-most n bits.

```
KEK = Truncate(SHA( K_PAD | AUTH_KEY ), 128)

HMAC_KEY_U = SHA( H_PAD_U | AUTH_KEY )

HMAC_KEY_D = SHA( H_PAD_D | AUTH_KEY )
```

Each _PAD_ is a 512 bit string:

```
K_PAD = 0x53 repeated 64 times.

H_PAD_U = 0x5C repeated 64 times.

H_PAD_D = 0x3A repeated 64 times.
```

7.5 Public-Key Encryption of Authorization Key

Authorization keys in Authorization Reply messages MUST be RSA public-key encrypted, using the cable modem's public key. BPI+ uses F4 (65537 decimal, or equivalently, 010001 hexadecimal) as its public exponent and a modulus length of 1024 bits. BPI+ employs the RSAES-OAEP encryption scheme specified in version 2.0 of the PKCS#1 standard [RSA3]. RSAES-OAEP requires the selection of: a hash function; a mask-generation function; and an encoding parameter string. The default selections specified in [RSA3] MUST be used when encrypting the authorization key. These default selections are: SHA-1 for the hash function; MGF1 with SHA-1 for the mask-generation function; and the empty string for the encoding parameter string.

Note that Baseline Privacy [DOCSIS2] employed the encryption scheme described in version 1.5 of the PKCS #1 standard [RSA1]. This is the same scheme as RSAES-PKCS1-v1_5 in [RSA3]. In order to maintain backwards compatibility, CMs and CMTSs MUST revert to RSAES-PKCS1-v1_5 for encrypting the authorization key when falling back to BPI.

The Baseline Privacy [DOCSIS2] protocol, whose support is required in DOCSIS 1.0 CMs, specifies an modulus length of 768 bits for its RSA keys. In order to enable software upgrades of DOCSIS 1.0 CM hardware to BPI+, the BPI+ protocol MUST support 768-bit as well as 1024-bit modulus lengths. DOCSIS 1.1 CMs, however, MUST employ RSA keys having a 1024-bit modulus length. To support interoperability with the upgraded v1.0 CMs, a DOCSIS 1.1 CMTS's BPI+ implementation MUST support 768-bit as well as 1024-bit modulus lengths.

7.6 Digital Signatures

The BPI+ employs the RSA Signature Algorithm [RSA3] with SHA-1 [FIPS186] for all three of its certificate types.

As with its RSA encryption keys, BPI+ uses F4 (65537 decimal, 010001 hexadecimal) as the public exponent for its signing operation. The DOCSIS Root CA will employ a modulus length of 2048 bits (256 octets) for signing the Manufacturer CA certificates it issues. Manufacturer CAs MUST employ signature key modulus lengths of at least 1024 bits, and no greater than 2048 bits.

7.7 Supporting Alternative Algorithms

The current BPI+ specification requires the use of 56-bit DES for encrypting packet data, two-key triple DES for encrypting traffic encryption keys, 1024-bit RSA for encrypting Authorization keys, and 1024-to-2048-bit RSA for signing BPI+ X.509 certificates. The choice of key lengths and algorithms, while appropriate for current threat models and hardware capabilities, may be inappropriate in the future.

For example, it is generally agreed that DES is approaching the end of its practical usefulness as the industry standard for symmetric encryption. NIST is currently overseeing the development and adoption of a new standard encryption algorithm, commonly referred to as the Advanced Encryption Standard, or AES. Given the nature of the security services BPI+ is being asked to support (basic privacy at a level better than or equal to that possible over dedicated wires, and conditional access to RF data transport services) as well as the protocol's flexible key management policy (i.e., setting of key lifetimes), DOCSIS-based service providers will be justified in the continued reliance on DES for, at least, the next five years. Nevertheless, at some future date, DOCSIS Cable modems will need to adopt a stronger traffic encryption algorithm, possibly AES.

Adopting a new algorithm for packet data encryption will not require a redesign of BPI+. The protocol's consistent use of Type/Length/Value encoding of BPKM attributes, MAC Header Extended Header elements, and security capabilities selection in the authorization exchange guarantee BPI+'s extensibility. In fact, changes in any of BPI+'s cryptographic algorithms, or associated key lengths, will have no impact on the overall structure and operation of the protocol.

8 Physical Protection of Keys in the CM and CMTS

BPI+ requires both CMs and CMTSs to maintain in their memory traffic encryption keys and CM Authorization Keys. A CM MUST also maintain in permanent, write-once memory an RSA key pair. Both CM and CMTS MUST deter unauthorized physical access to this keying material.

The level of physical protection of keying material BPI+ requires of CMs and CMTSs is specified in terms of the security levels defined in the FIPS PUBS 140-1, Security Requirements for Cryptographic Modules, standard [FIPS-140-1]. In particular, CMs and CMTSs MUST meet FIPS PUBS 140-1 Security Level 1 requirements.

FIPS PUBS 140-1 Security Level 1 requires minimal physical protection through the use of production-grade enclosures. The reader should refer to the FIPS document for the formal requirements; however, below is a summary of those requirements.

Under the FIPS PUBS 140-1 classification of “physical embodiments” of cryptographic modules, CMTSs and external CMs are *multiple-chip stand-alone cryptographic modules*. FIPS PIBS 140-1 specifies the following Security level 1 requirements for multiple-chip stand-alone modules:

- The chips shall be of production-grade quality, which shall include standard passivation techniques (i.e., a sealing coat over the chip circuitry to protect it against environmental or other physical damage).
- The circuitry within the module shall be implemented as a production grade multiple-chip embodiment (i.e., an IC printed circuit board, a ceramic substrate, etc.).
- The module shall be entirely contained within a metal or hard plastic production-grade enclosure, which may include doors or removable covers.

An internal CM (defined in [DOCSIS4]) would be classified as a FIPS PUBS 140-1 *multiple-chip embedded cryptographic module*; the Security Level 1 requirements for these devices are the two first bullets listed above.

This page intentionally left blank.

9 BPI+ X.509 Certificate Profile and Management

DOCSIS BPI+ shall employ X.509 version 3 digital certificates for authenticating key exchanges between CM and CMTS. X.509 is a general purpose standard; the BPI+ certificate profile, described here, further specifies the contents of the certificate's defined fields. The certificate profile also defines the hierarchy of trust defined for the management and validation of DOCSIS BPI+ certificates.

Except where otherwise noted in following subsections, DOCSIS BPI+ certificates MUST be in compliance with the IETF's PKIX standards [RFC3280].[15] DOCSIS's usage of X.509 certificates, however, is far more circumscribed than that of PKIX. The IETF's PKIX X.509 certificate profile is aimed at supporting an application-independent, certificate-based, key distribution mechanism across the public Internet. The PKIX X.509 certificate profile must support a wide range of communications environments, applications, and trust relationships.

In contrast, BPI+'s use of digital certificates is restricted to safeguarding MSOs from piracy of DOCSIS data communications services through enforcing conditional access to traffic encryption keys. The protected communications services fall into three categories:

- best-effort, high-speed, IP data services
- premium CBR (constant bit rate) data services
- access to premium IP multicast groups

Thus, while BPI+ draws heavily from the IETF's PKIX X.509 certificate profile effort, the BPI+ X.509 profile is significantly more prescribed.

The BPI+ X.509 Certificate Profile also draws extensively from the Secure Electronic Transaction (SET) standard [SET Book 2]. Both the overall organization of this section, and some of the section's contents reflect that standard.

9.1 BPI+ Certificate Management Architecture Overview

The DOCSIS BPI+ certificate management architecture, depicted in Figure 9-1, consists of a three-level hierarchy of trust supporting three types of X.509 version 3 certificates:

- a single, self-signed, DOCSIS Root CA certificate
- manufacturer CA certificates
- CM certificates

The DOCSIS Root Certification Authority serves as the root CA. The root CA issues certificates to subordinate CAs maintained by manufacturers. Manufacturer CAs issue certificates to cable modem end entities. Note that a single manufacturer may maintain multiple CAs (e.g., a different CA for each manufacturing plant).

Currently, the DOCSIS Root Certificate Authority also serves as the root CA to issue the Code Verification Certificate (CVC) for the Secure Software Downloading specified in Appendix D. However, there is no security reason to require the same root CA to issue both the Manufacturer CA

15. Reference updated per BPI-N-02098 by RKV on 8/23/02.

Certificate and the CVC. Therefore, the CVC may be issued by the different root Certificate Authority in the future.[16]

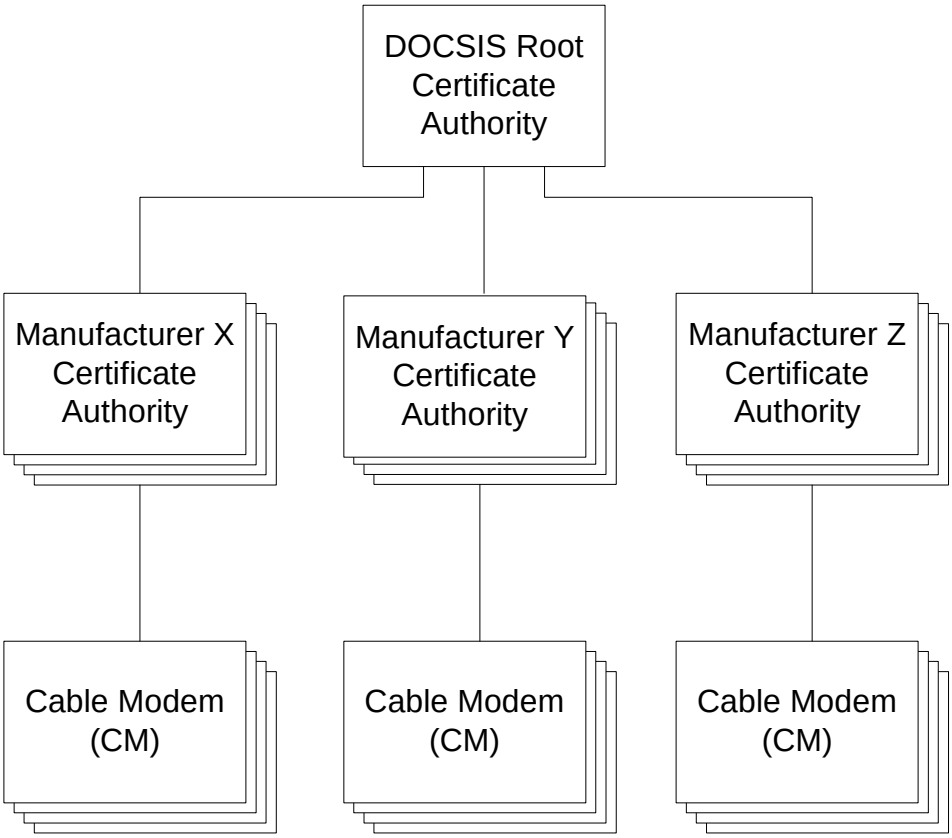


Figure 9-1. DOCSIS Certificate Management Architecture

The DOCSIS Root CA shall be kept under tight physical controls. It will be accessed infrequently to issue new Manufacturer CA certificates. The organization responsible for DOCSIS certification will be responsible for maintaining the DOCSIS Root CA. The DOCSIS Root CA shall generate and distribute to MSOs a Certificate Revocation List (CRL) identifying revoked manufacturer certificates. The manner in which CRLs are distributed to the MSOs is outside the scope of the BPI+ specification.

The organization maintaining the DOCSIS Root CA shall define a protocol for Manufacturer–generated certificates to the requesting Manufacturer CA. Specification of this protocol, however, is outside the scope of the BPI+ specification.

Manufacturers will be responsible for maintaining their own CA, from which they will issue CM certificates. A single manufacturer may maintain multiple Manufacturer CAs. Protocols for requesting certificates from a manufacturer CA and distributing the resulting certificates to the receiving Cable Modems shall be internal to that manufacturer, and thus outside the scope of the BPI+ specification. A Manufacturer CA MAY generate and distribute to MSOs CRLs; the manner in which these CRLs are distributed to MSOs is outside the scope of the BPI+ specification.

16. paragraph added per bpi-n-02008, 07/29/02, ab

9.2 Certificate Format

This section describes the X.509 version 3 certificate format and certificate extensions used in BPI+. Table 9-1 below summarizes the basic fields of an X.509 Version 3 certificate.

Table 9-1. X.509 Basic Certificate Fields

X.509 v3 Field	Description
tbsCertificate.version	Indicates the X.509 certificate version. Always set to v3 (value of 2)
tbsCertificate.serialNumber	Unique integer the issuing CA assigns to the certificate.
tbsCertificate.signature	OID and optional parameters defining algorithm used to sign the certificate. This field MUST contain the same algorithm identifier as the signatureAlgorithm field below.
tbsCertificate.issuer	Distinguished Name of the CA that issued the certificate
tbsCertificate.validity	Specifies when the certificate becomes active and when it expires.
tbsCertificate.subject	Distinguished Name identifying the entity whose public key is certified in the subject public key information field
tbsCertificate .subjectPublicKeyInfo	Field contains the public key material (public key and parameters) and the identifier of the algorithm with which the key is used.
tbsCertificate .issuerUniqueID	Optional field to allow reuse of issuer names over time.
tbsCertificate .subjectUnique ID	Optional field to allow reuse of subject names over time.
tbsCertificate.extensions	The extension data.
signatureAlgorithm	OID and optional parameters defining algorithm used to sign the certificate. This field MUST contain the same algorithm identifier as the signature field in tbsCertificate.
signatureValue	Digital signature computed upon the ASN.1 DER encoded tbsCertificate.

All certificates and CRLs described in this specification MUST be signed with the RSA signature algorithm, using SHA-1 as the one-way hash function. The RSA signature algorithm is described in PKCS #1 [RSA1]; SHA-1 is described in [FIPS-180-1]. This is just one example of how BPI+ restricts the values of the X.509 Certificate's basic fields. All of these restrictions are described below:

9.2.1 tbsCertificate.validity.notBefore and tbsCertificate.validity.notAfter

Cable Modem certificates will not be renewable, and, thus, must have a validity period greater than the operational lifetime of the cable modem. A Manufacturer CA certificate MUST be valid from the issuance date for a period defined by [DOCSIS5] or [DOCSIS10] and re-issued in a period defined by [DOCSIS5] or [DOCSIS10]. The DOCSIS Root CA certificate MUST be valid from the date when the DOCSIS Root CA starts operating for a period defined by the [DOCSIS5] or [DOCSIS10] and re-issued in a period defined by [DOCSIS5] or [DOCSIS10].[17]

17. References updated per BPI-N-02098 by RKV on 8/23/02.

This specification assumes the operational lifetime of a Cable Modem will not exceed twenty years. The validity period of a Cable Modem certificate MUST begin with the device's data of manufacture; the validity period SHOULD extend out to at least 20 years after that manufacturing date.

Validity periods MUST be encoded as UTCTime. UTCTime values MUST be expressed Greenwich Mean Time (Zulu) and MUST include seconds (i.e., times are YYMMDDHHMMSSZ), even where the number of seconds is zero. The year field (YY) MUST be interpreted as follows:

- where YY is greater than or equal to 50, the year shall be interpreted as 19YY
- where YY is less than 50, the year shall be interpreted as 20YY

9.2.2 tbsCertificate.serialNumber

The serial number MUST be a positive integer assigned by the CA to each certificate. It MUST be unique for each certificate issued by a given CA (i.e., the issuer name and serial number identify a unique certificate). CAs MUST force the serialNumber to be a non-negative integer. The Manufacturer SHOULD NOT impose or assume a relationship between the serial number of the certificate and the serial number of the modem to which the certificate is issued.

Given the uniqueness requirements above, serial numbers can be expected to contain long integers. Certificate users MUST be able to handle serialNumber values up to 20 octets. Conformant CAs MUST NOT use serialNumber values longer than 20 octets.

Note: Certificate users in the DOCSIS 1.1 system MUST be prepared to handle certificates that may already have negative, or zero, serial numbers, to ensure backwards compatibility.

9.2.3 tbsCertificate.signature and signatureAlgorithm

All certificates and CRLs described in this specification MUST be signed with the RSA signature algorithm, using SHA-1 as the one-way hash function. The RSA signature algorithm is described in PKCS #1 [RSA1]; SHA-1 is described in [FIPS-180-1].

The ASN.1 OID used to identify the "SHA-1 with RSA" signature algorithm is:

```
sha-1WithRSAEncryption OBJECT IDENTIFIER ::= {
      iso(1) member-body(2) us(840) rsadsi(113549) pkcs(1)
      pkcs-1(1) 5}
```

When the sha-1WithRSAEncryption OID appears within the ASN.1 type AlgorithmIdentifier, as is the case with both tbsCertificate.signature and signatureAlgorithm, the parameters component of that type is the ASN.1 type NULL.

9.2.4 tbsCertificate.issuer and tbsCertificate.subject

X.509 Names are SEQUENCES of RelativeDistinguishedNames, which are in turn SETs of AttributeTypeAndValue. AttributeTypeAndValue is a SEQUENCE of an AttributeType (an OBJECT IDENTIFIER) and an AttributeValue. The value of the countryName attribute MUST be a 2-character PrintableString, chosen from ISO 3166; all other AttributeValues MUST be encoded as either

T.61/TeletexString or PrintableString character strings. The PrintableString encoding MUST be used if the character string contains only characters from the PrintableString set. Specifically:

abcdefghijklmnopqrstuvwxyz
ABCDEFGHIJKLMNOPQRSTUVWXYZ
0123456789
'()+,-./:=? and space.

The T.61/TeletexString MUST be used if the character string contains other characters.

The following OIDs are needed for defining issuer and subject Names in BPI+ certificates:

```
id-at OBJECT IDENTIFIER ::= {joint-iso-ccitt(2) ds(5) 4}
id-at-commonName        OBJECT IDENTIFIER ::= {id-at 3}
id-at-countryName       OBJECT IDENTIFIER ::= {id-at 6}
id-at-localityName      OBJECT IDENTIFIER ::= {id-at 7}
id-at-stateOrProvinceName     OBJECT IDENTIFIER ::= {id-at 8}
id-at-organizationName        OBJECT IDENTIFIER ::= {id-at 10}
id-at-organizationalUnitName  OBJECT IDENTIFIER ::= {id-at 11}
```

The following subsections describe the format of the subject name field for each type of BPI+ certificate. The issuer name field of a certificate matches the subject name field of the issuing certificate. Any certificate transmitted by a CM in an Auth Info or Auth Request message MUST have name fields that conform to the indicated format. A CMTS MUST be capable of processing the name fields of a certificate if the name fields conform to the indicated format. A CMTS MAY choose to accept a certificate that has name fields that do not conform to the indicated format.

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. [RFC3280] recommends that certificate authorities restrict the encoding of name fields so that an implementation can declare a match or mismatch using simple binary comparison. BPI+ follows this recommendation. Accordingly, the DER-encoded tbsCertificate.issuer field of a BPI+ 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.[18]

9.2.4.1 DOCSIS Root Certificate

countryName=US
organizationName=Data Over Cable Service Interface Specifications
organizationalUnitName=Cable Modems
commonName=DOCSIS Cable Modem Root Certificate Authority

The countryName, organizationName, organizationalUnitName and commonName attributes MUST be included and MUST have the values shown. Other attributes are not allowed and MUST NOT be included.

18. Reference updated per Alex Katsnelson by RKV on 8/23/02.

9.2.4.2 DOCSIS Manufacturer Certificate

countryName=<Country of Manufacturer>
[stateOrProvinceName=<state/province>]
[localityName=<City>]
organizationName=<Company Name>
organizationalUnitName=DOCSIS
[organizationalUnitName=<Manufacturing Location>]
commonName=<Company Name> [<Serial Identifier>] Cable Modem Root Certificate Authority [<Serial Identifier>]

The countryName, organizationName, and commonName attributes MUST be included and MUST have the values shown.

The commonName MAY contain a serial identifier (e.g., 1, 2, ONE, TWO, A, B, I, II, etc.) to identify different Manufacturer CAs deployed by the same Manufacturers with the same Company Name.

The organizationalUnitName having the value “DOCSIS” MUST be included.

The organizationalUnitName representing manufacturing location SHOULD be included. If included, it MUST be preceded by the organizationalUnitName having value “DOCSIS.”

The stateOrProvinceName and localityName MAY be included.

Other attributes are not allowed and MUST NOT be included.

9.2.4.3 Cable Modem Certificate

countryName=<Country of Manufacturer>
organizationName=<Company Name>
organizationalUnitName=<manufacturing location>
commonName=<Serial Number>
commonName=<MAC Address>

To distinguish between the two commonNames, the commonName representing the “Serial Number” MUST precede the commonName representing “MAC Address”. Use of the Serial Number field is deprecated. If used, the Serial Number MUST be a unique cable modem identifier, but MAY be different from the serial number encoded in the BPKM attributes. The MAC address in the CM Certificate MUST be the same as the MAC address in the BPKM Attributes.

The characters employed in the PrintableString representation of CM serial numbers MUST be restricted to the following character subset:

- A-Z (0x41-0x5A)
- a-z (0x61-0x7A)
- 0-9 (0x30-0x39)
- “–” (0x2D)

The MAC Address is expressed as six pairs of hexadecimal digits separated by colons (:), e.g., “00:60:21:A5:0A:23”. The Alpha HEX characters (A-F) MUST be expressed as uppercase letters.

The organizationalUnitName in a Cable Modem certificate, which describes the modem’s manufacturing location, SHOULD be the same as the organizationalUnitName in the issuer Name describing a manufacturing location.

The countryName, organizationName, organizationalUnitName, and commonName (MAC Address) attributes MUST be included. The commonName (Serial Number) attribute MAY be included. Other attributes are not allowed and MUST NOT be included.

9.2.5 tbsCertificate.subjectPublicKeyInfo

The tbsCertificate.subjectPublicKeyInfo field contains the public key and the public key algorithm identifier. The RSA public key in the CM Certificate MUST be the same as the RSA public key in the BPKM Attributes.

The tbsCertificate.subjectPublicKeyInfo.algorithm field is an AlgorithmIdentifier structure. The AlgorithemIdentifier’s algorithm MUST be RSA encryption, identified by the following OID:

```
pkcs-1 OBJECT IDENTIFIER ::= { iso(1) member-body(2) us(840)
      rsadsi(113549) pkcs(1) 1}

rsaEncryption OBJECT IDENTIFIER ::= { pkcs-1 1}
```

The AlgorithmIdentifier’s parameters field MUST have ASN.1 type NULL.

The RSA public key shall be encoded using the ASN.1 type RSAPublicKey:

```
RSAPublicKey ::= SEQUENCE {
      modulus           INTEGER, -- n
      publicExponent    INTEGER, -- e -- }
```

where modulus is the modulus n, and publicExponent is the public exponent e. The DER encoded RSAPublicKey is the value of the BIT STRING tbsCertificate.subjectPublicKeyInfo.subjectPublicKey.

9.2.6 tbsCertificate.issuerUniqueID and tbsCertificate.subjectUniqueID

The issuerUniqueID and subjectUniqueID fields MUST be omitted for all three of BPI+’s certificate types.

9.2.7 tbsCertificate.extensions

Cable Modem certificates and DOCSIS Manufacturer CA certificates are not required to include any extensions; this is true even for extensions mandated by [RFC3280]. Cable Modem certificates and DOCSIS Manufacturer CA certificates may include extensions as described in sections 9.2.7.1 and 9.2.7.2, respectively. The section 9.2.7.3 specifies the requirements on the extensions of DOCSIS Root CA certificate. Extensions included in BPI+ certificates MUST conform to [RFC3280].[19]

9.2.7.1 Cable Modem Certificates

Cable Modem certificates MAY contain noncritical extensions; they MUST NOT contain critical extensions. If the KeyUsage extension is present, the digitalSignature and keyEncipherment bits MUST be turned on, keyCertSign and cRLSign bits MUST be turned off, and all other bits SHOULD be turned off. The basicConstraints extension MAY appear as a non-critical extension in cable modem certificates.

9.2.7.2 DOCSIS Manufacturer CA Certificates

DOCSIS Manufacturer CA certificates MAY contain the Basic Constraints extension and/or the Key Usage extension. If included, these extensions MAY appear as a critical extension or as a noncritical extension.

DOCSIS Manufacturer CA certificates MAY contain noncritical extensions; they MUST NOT contain critical extensions other than, possibly, the Basic Constraints extension and the Key Usage extension.

If the Key Usage extension is present in a DOCSIS Manufacturer CA certificate, the keyCertSign bit MUST be turned on, cRLSign bit MAY be turned on, and all other bits SHOULD be turned off.

If the Basic Constraints extension is present, the cA MUST be set to TRUE and the pathLenConstraint MUST be set to 0.

9.2.7.3 DOCSIS Root CA Certificate

DOCSIS Root CA certificate MUST contain the Basic Constraints extension and the Key Usage extension as critical extensions.

DOCSIS Root CA certificate MAY contain noncritical extensions; they MUST NOT contain critical extensions other than the Basic Constraints extension and the Key Usage extension.

For the KeyUsage extension, the keyCertSign bit MUST be turned on, cRLSign bit MAY be turned on, and all other bits SHOULD be turned off.

For the Basic Constraints extension, the cA MUST be set to TRUE and the pathLenConstraint MUST be set to 1.

9.2.8 signatureValue

In all three BPI+ certificate types, the signatureValue contains the RSA (with SHA-1) signature computed over the ASN.1 DER encoded tbsCertificate. The ASN.1 DER encoded tbsCertificate is used as input to the RSA signature function. The resulting signature value is ASN.1 encoded as a BIT STRING and included in the Certificate's signatureValue field.

---

19. References updated per Alex Katsnelson by RKV on 8/23/02.

9.3 Cable Modem Certificate Storage and Management in the CM

Manufacturer-issued CM certificates MUST be stored in CM permanent, write-once memory. CMs that have factory-installed RSA private/public key pairs MUST also have factory-installed CM certificates. CMs that rely on internal algorithms to generate an RSA key pair MUST support a mechanism for installing a manufacturer-issued CM certificate following key generation.

The root CA public key for the CVC verification, which the CM uses to verify the Code Verification Certificate (CVC) for the Secure Software Download defined in Appendix D, MUST be placed into the CM's non-volatile memory. While the DOCSIS Root CA for the cable modem certificate chain currently issues the CVC, a different root CA may issue the CVC in the future. Therefore, the CM MUST NOT use the root CA public key for the CVC verification embedded in the CM's non-volatile memory in order to verify the cable modem certificate chain.[20]

The CA certificate of the Manufacturer CA that signed the CM certificate MUST be stored in the cable modem's non-volatile memory. The cable modem MUST be capable of updating or replacing the Manufacturer CA certificate via the DOCSIS code download file (see Appendix D). The Manufacturer CA certificate MAY be embedded into the CM software.

In the case where the Manufacturer CA certificate is embedded into the CM software, if a manufacturer issues CM certificates with multiple CA certificates the CM memory must include ALL of that manufacturer's CA certificates. The specific Manufacturer CA certificate installed by the CM (i.e., advertised in Authentication Information messages and returned by the MIB object) will be that identifying the issuer of that modem's CM certificate.

9.4 Certificate Processing and Management in the CMTS

BPKM employs digital certificates to allow CMTSs to verify the binding between a CM's identity (encoded in an X.509 digital certificate's subject names) and its public key. The CMTS does this by validating the CM certificate's certification path or chain. This path will typically consist of three chained certificates: starting with the CM Certificate, the path leads to the certificate of the Manufacturer CA that issued the CM Certificate, and ends at the DOCSIS Root CA's self-signed certificate (Figure 9-2). Validating the chain means verifying the Manufacturer CA Certificate's signature with the DOCSIS Root CA's public key and then verifying the CM Certificate's signature with the public key of the Manufacturer CA.

20. paragraph replaced per bpi-n-02008, 07/25/02, ab

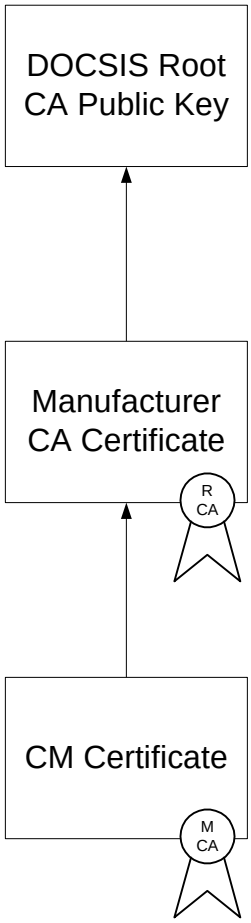


Figure 9-2. CM Certification Chain

BPI+ requires that CMTSs support administrative controls that allow the operator to override certification chain validation by specifying a Manufacturer CA or CM Certificate to be trusted or untrusted. A detailed description of these administrative controls on CMTS certificate management is provided in BPI+'s associated OSS document [DOCSIS8]. This section specifies the management model for the exercise of these controls, as well as the processing a CMTS undertakes to assess a CM Certificate's validity, and thus verify the binding between the CM's identity and its public key.

9.4.1 CMTS Certificate Management Model

The CMTS maintains copies of Root CA, Manufacturer CA and Cable Modem Certificates, which it obtains through either provisioning or BPKM messaging. Each certificate a CMTS learns of MUST be marked as being in one of four states: Untrusted, Trusted, Chained or Root. Only the DOCSIS Root CA Certificate (a self-signed certificate containing the DOCSIS Root CA's trusted public key) MUST be marked as Root. However, a CMTS MAY support multiple Root CA Certificates. Root Certificate(s) MUST be provisioned within a CMTS and the CMTS MUST support the function to show the entire Root Certificate(s) and/or its thumbprint so that the operator can verify the Root Certificate(s).

A CMTS learns of Manufacturer CA Certificates through either the CMTS's provisioning interface or through receipt and processing of client CMs' Authentication Information messages. Regardless of how a CMTS obtains its Manufacturer CA Certificates, the CMTS MUST mark them as either Untrusted, Trusted or Chained. If a Manufacturer CA Certificate *is not* self-signed, the CMTS marks the certificate as Chained. The CMTS, however, MUST support administrative controls that allow an operator to override the Chained marking and specify that a given Manufacturer CA Certificate is Trusted or Untrusted.

If a Manufacturer CA Certificate *is* self-signed, the CMTS marks the certificate as either Trusted or Untrusted, according to administratively controlled CMTS policy. A self-signed Manufacturer CA Certificate whose signature cannot be verified MUST be marked as Untrusted. CMTS trusting of self-signed Manufacturer CA Certificates MUST be configurable. Default trusting of self-signed Manufacturer CA Certificates is NOT RECOMMENDED in commercially operational systems; default trusting would primarily be used to support certification and other testing modes. The CMTS MUST mark the CM Certificate as Chained unless overridden by CMTS's administrative control.

A CMTS obtains copies of Cable Modem Certificates in the Authorization Requests it receives from client CMs. Cable Modem Certificates MUST be issued by a Manufacturer CA; thus, unless overridden by CMTS administrative control, the CMTS will mark CM Certificates as Chained. An operator may, as part of the modem provisioning process, specify that a given CM's certificate be marked as Untrusted or Trusted.

9.4.2 Certificate Validation

The CMTS validates the certification paths of Manufacture CA and CM Certificates using the following criteria. Note that the criteria are iterative and require a CMTS to validate the certification path of a Chained Manufacturer CA certificate before it can validate the certification path of a CM Certificate issued by that Manufacturer CA.

The CMTS labels Manufacturer CA and Cable Modem Certificates as *Valid* or *Invalid* if their certification paths are valid or invalid, respectively. Trusted certificates are Valid; this is true even if the current time does not fall within the Trusted certificate's validity period. Untrusted certificates are Invalid.

A Chained certificate is Valid if:

1. the certificate chains to either a Root, Trusted, or Valid certificate; *and*
2. the certificate's signature can be verified with the issuer's public key; *and*
3. the current time falls within the validity period of each Chained or Root certificate within the certificate chain (note that BPI+ does not require the nesting of validity periods, i.e., a certificate's entire validity period need not fall within the validity period of it's issuing certificate); *and*
4. the certificate is not on a *hot list* of CM and Manufacturer CA Certificates (see Section 9.4.4).
5. in the case of a CM certificate, the CM MAC address encoded in its tbsCertificate.subject field and RSA public key encoded on its tbsCertificate.subjectPublicKeyInfo field match the CM MAC address and RSA public key encoded in the Authorization Request's BPKM Attributes; *and*

6. in the case of a CM Certificate, if the KeyUsage extension is present, the digitalSignature and/or keyAgreement bits are turned on, the keyEncipherment bit is turned on, and the keyCertSign and cRLSign bits are off; in the case of a Manufacturer CA Certificate, if the KeyUsage extension is present, the keyCertSign bit is turned on.[21]

Whether criteria 3 above is ignored MUST be subject to administrative control.

If validity period checking is ENABLED and the time of day has not been acquired by the CMTS, a (non-permanent) authorization reject message MUST be returned in response to a BPI+ style authorization request.

If a Chained Certificate certificate does not satisfy any one of the above validity criteria, it is identified as being Invalid.

If a CMTS marks a CM Certificate as being either Untrusted or Invalid, the CMTS MUST reject the corresponding CM's Authorization Requests.

9.4.3 Certificate Thumbprints

Thumbprints are collision-resistant one-way hash functions (e.g. SHA-1) of certificates. They provide a compact way to identify certificates. A CMTS MAY keep Thumbprints of CM and Manufacturer CA certificates it holds or has validated. Using Thumbprints, a CMTS can cache the results of an earlier validation operation: by matching the Thumbprint of a newly offered certificate with that of a cached Thumbprint, it can quickly determine the validity of the offered certificate.

9.4.4 Manufacturer CA and CM Certificate Hot Lists

When validating certificate chains, the CMTS is not required to check a certificate's revocation status (i.e., check for the certificate's presence on an up-to-date CRL). The CMTS, however, MUST be capable of maintaining *hot lists* of known, untrusted, Manufacturer CA and CM certificates. Certificates on these hot lists may include certificates revoked by their issuers; however, they may also include valid certificates that the MSO operating the CMTS chooses to mark as "untrusted".

Definition of procedures and protocols for maintaining a CMTS's Manufacturer CA certificate and CM certificate hot lists are outside the scope of the BPI+ specification.

21. edited 07/25/02 per bpi-n-02011, ab
edited 07/30/02 per bpi-n-02113, ab

Appendix A TFTP Configuration File Extensions

All of a CM's Baseline Privacy configuration parameter values are specified in the configuration file TFTP-downloaded by the CM during RF MAC initialization. Baseline Privacy configuration setting fields are included in both the CM MIC and CMTS MIC calculations, and in a CM's registration requests. Refer to [DOCSIS1] for the order in which Baseline Privacy configuration setting fields are included in the CMTS MIC's MD5 digest.

A.1 Encodings

The following type/length/value encodings MUST be used for any Baseline Privacy configuration settings included in the configuration file. The Baseline Privacy configuration settings in the RF MAC CM registration requests MUST be the same as those included in the configuration file. All multi-octet quantities are in network-byte order, *i.e.*, the octet containing the most-significant bits is the first transmitted on the wire.

A.1.1 Baseline Privacy Configuration Setting

The combination of RFI 1.1's Privacy Enable configuration setting ([DOCSIS1] section C.1.1.16) and the Privacy Support Modem Capability Setting ([DOCSIS1] section C.1.3.1.6) controls whether Baseline Privacy Plus is enabled or disabled in a CM. If the operator intends to provision a CM to operate in BPI+ mode using the default BPI Configuration Parameter(s) specified in the Table A-1, the corresponding Baseline Privacy Configuration subsetting(s) in the configuration file MAY be omitted. If the configuration file does not contain all the necessary BPI+ parameters, the CM MUST use the default value(s) specified in the Table A-1 for the missing parameter(s). On the other hand, if the operator intends to provision a CM to operate in BPI+ mode using the BPI Configuration Parameter(s) different from the default value(s) in the Table A-1, the corresponding Baseline Privacy Configuration subsetting(s) MUST be present. The Baseline Privacy Configuration setting MAY be present if Baseline Privacy Plus is disabled. The separate Privacy Enable parameter allows an operator to disable or re-enable Baseline Privacy by toggling a single configuration parameter, thus not requiring the removal or re-insertion of the larger set of Baseline Privacy Configuration parameters.

This field defines the parameters associated with Baseline Privacy operation. It is composed of a number of encapsulated type/length/value fields. The type fields defined are only valid within the encapsulated Baseline Privacy configuration setting string.

type	length	value
BP_CFG	n	

[DOCSIS1] defines the specific value of BP_CFG.

A.1.1.1 Internal Baseline Privacy Encodings

A.1.1.1.1 Authorize Wait Timeout

The value of the field specifies retransmission interval, in seconds, of Authorization Request messages from the Authorize Wait state.

sub-type	length	value
1	4	

Valid Range: 1 - 30

A.1.1.1.2 Reauthorize Wait Timeout

The value of the field specifies retransmission interval, in seconds, of Authorization Request messages from the Authorize Wait state.

sub-type	length	value
2	4	

Valid Range: 1 - 30

A.1.1.1.3 Authorization Grace Time

The value of this field specifies the grace period for re-authorization, in seconds.

sub-type	length	value
3	4	

Valid Range: 1 - 6,047,999

A.1.1.1.4 Operational Wait Timeout

The value of this field specifies the retransmission interval, in seconds, of Key Requests from the Operational Wait state.

sub-type	length	value
4	4	

Valid Range: 1 - 10

A.1.1.1.5 Rekey Wait Timeout

The value of this field specifies the retransmission interval, in seconds, of Key Requests from the Rekey Wait state.

sub-type	length	value
5	4	

Valid Range: 1 - 10

A.1.1.1.6 TEK Grace Time

The value of this field specifies grace period, in seconds, for rekeying the TEK.

sub-type	length	value
6	4	

Valid Range: 1 - 302399

A.1.1.1.7 Authorize Reject Wait Timeout

The value of this field specifies how long a CM waits (seconds) in the Authorize Reject Wait state after receiving an Authorization Reject.

sub-type	length	value
7	4	

Valid Range: 1 - 600

A.1.1.1.8 SA Map Wait Timeout

The value of this field specifies the retransmission interval, in seconds, of SA Map Requests from the Map Wait state.

sub-type	length	value
8	4	

Valid Range: 1 - 10

A.1.1.1.9 SA Map Max Retries

The value of this field specifies the maximum number of Map Request retries allowed.

sub-type	length	value
9	4	

Valid Range: 0 - 10

A.2 Parameter Guidelines

Below are recommended ranges and values for Baseline Privacy's various configuration and operational parameters. These ranges and default values may change as service providers gain operational experience running Baseline Privacy.

Table A-1. Recommended Operational Ranges for BPI Configuration Parameters

System	Name	Description	Minimum Value	Default Value	Maximum Value
CMTS	Authorization Lifetime	Lifetime, in seconds, CMTS assigns to new Authorization Key	1 day (86,400 sec.)	7 days (604,800 sec.)	70 days (6,048000 sec.)
CMTS	TEK Lifetime	Lifetime, in seconds, CMTS assigns to new TEK	30 min. (1800 sec.)	12 hours (43,200 sec.)	7 days (604,800 sec.)
CM	Authorize Wait Timeout	Auth Req retransmission interval from Auth Wait state	2 sec.	10 sec.	30 sec.
CM	Reauthorize Wait Timeout	Auth Req retransmission interval from Reauth Wait state	2 sec.	10 sec.	30 sec.
CM	Authorization Grace Time	Time prior to Authorization expiration CM begins re-authorization	5 min. (300 sec.)	10 min. (600 sec.)	35 days (3,024,000 sec).
CM	Operational Wait Timeout	Key Req retransmission interval from Op Wait state	1 sec.	10 sec.	10 sec.
CM	Rekey Wait Timeout	Key Req retransmission interval from Rekey Wait state	1 sec.	10 sec.	10 sec.
CM	TEK Grace Time	Time prior to newer TEK expiration CM begins rekeying	5 min. (300 sec)	1 hour (3,600 sec.)	3.5 days (302,399 sec)
CM	Authorize Reject Wait	Delay before re-sending Auth Request after receiving Auth Reject	10 sec.	60 sec.	10 min. (600 sec.)
CM	SA Map Wait Timeout	Map Request retransmission interval from Map Wait state	1 sec.	1 sec.	10 sec.
CM	SA Map Max Retries	Maximum number of times CM retries SA Map Request before giving up	0	4	10

The valid range (vs. recommended operational range) for Authorization and TEK lifetimes are:

- Authorization Lifetime Valid Range: 1 - 6,048,000 seconds
- TEK Lifetime Valid Range: 1 - 604,800 seconds

Note that valid ranges defined for each of BPI's configuration parameters extend below the recommended operational ranges. For the purposes of protocol testing, it is useful to run the BPI protocol with timer values well below the low end of the recommended operational ranges. The shorter timer values "speed up" BPI's clock, causing BPI protocol state machine events to occur far more rapidly than they would under an "operational" configuration. While BPI implementations need not be designed to operate efficiently at this accelerated BPI pace, the protocol implementation SHOULD operate correctly under these shorter timer values. Table A-2 provides a list of shortened parameter values which are likely to be employed in protocol conformance and certification testing.

Table A-2. Shortened BPI Parameter Values for Protocol Testing

Authorization Lifetime	5 min. (300 sec.)
TEK Lifetime	3 min. (180 sec.)
Authorization Grace Time	1 min. (60 sec.)
TEK Grace time	1 min. (60 sec.)

The TEK Grace Time MUST be less than half the TEK lifetime.

This page intentionally left blank.

Appendix B Example Messages, Certificates and PDUs (informative)

This appendix presents numerical examples which may be useful to implementors of the specification. The examples walk through a typical key exchange: Authorization Info, Authorization Request, Authorization Reply, Key Request, and Key Reply. Details of the cryptographic calculations are provided at each step, and example certificates are included. The examples also include several PacketPDUs, encrypted using the keying material derived in the example key exchange.

This appendix is informative only and does not constitute any part of the specification.

B.1 Notation

In the examples here, packets are represented as a stream of octets, each octet in hex notation, sometimes with a text annotation. The order of transmission for the octets is left to right, top to bottom. For example, consider the following representation of a packet:

00 01 02 03	Description #1
04 05	
06 07 08	Description #2

The packet consists of 9 octets, represented in hex notation as “00”, “01”, ..., “08”. The octet represented by “00” is transmitted first, and the octet represented by “08” is transmitted last.

In the discussion of the examples, integer values are represented in either hex notation using an “0x” prefix or in decimal notation with no prefix. For example, the hex notation 0x12345 and the decimal notation 74565 represent the same integer value. All integer values are non-negative. Thus, 0xff represents the integer having value 255, not a negative value.

The BPKM protocol generates and distributes 8-octet DES keys and 16-octet triple-DES keys, without correcting the least significant bit of each octet for parity. Implementations extract a 56-bit key from an 8-octet key and a 112-bit key from a 16-octet key by ignoring the value of the least significant bit of each octet. In the examples here, keys are represented without parity correction.

B.2 Authentication Info

The CM sends the following Authentication Info message:

0c 01 02 94	Auth Info header
11 02 91	CA Certificate header
30 82 02 8d 30 82 01 f6 . . . 81 87 19 61 72 20 19 1e	CA Certificate

The code field has value 0x0c, which identifies this as an Authentication Info message. The Length field has value 0x294 (660), which is the number of octets that follow the Length field.

The only attribute is the CA Certificate. Details of the certificate are given below.

B.2.1 CA Certificate details

The fields of the CA Certificate in the Authorization Info message above break down as follows:

30 82 02 8d	certificate header
30 82 01 f6	tbsCertificate header
a0 03 02 01 02	version
02 08 01 02 03 04 05 06 07 08	serial number
30 0d 06 09 2a 86 48 86 f7 0d 01 01 05 05 00	signature
30 81 88	issuer header
31 0b 30 09 06 03 55 04 06 13 02 55 53	country name
31 0f 30 0d 06 03 55 04 0a 13 06 4e 6f 72 74 65 6c	organization name
31 0f 30 0d 06 03 55 04 0b 13 06 44 4f 43 53 49 53	organizational unit name
31 1f 30 1d 06 03 55 04 0b 13 16 42 75 69 6c 64 69 6e 67 20 31 2c 20 41 6e 64 6f 76 65 72 20 4d 41	organizational unit name
31 36 30 34 06 03 55 04 03 13 2d 4e 6f 72 74 65 6c 20 43 61 62 6c 65 20 4d 6f 64 65 6d 20 52 6f 6f 74 20 43 65 72 74 69 66 69 63 61 74 65 20 41 75 74 68 6f 72 69 74 79	common name
30 1e	validity header
17 0d 39 39 30 31 32 30 31 36 30 35 30 30 5a	not before
17 0d 34 39 31 32 33 31 32 33 35 39 35 35 5a	not after
30 81 88	subject header
31 0b 30 09 06 03 55 04 06 13 02 55 53	country name
31 0f 30 0d 06 03 55 04 0a 13 06 4e 6f 72 74 65 6c	organization name
31 0f 30 0d 06 03 55 04 0b 13 06 44 4f 43 53 49 53	organizational unit name
31 1f 30 1d 06 03 55 04 0b 13 16 42 75 69 6c 64 69 6e 67 20 31 2c 20 41 6e 64 6f 76 65 72 20 4d 41	organizational unit name
31 36 30 34 06 03 55 04 03 13 2d 4e 6f 72 74 65 6c 20 43 61 62 6c 65 20 4d 6f 64 65 6d 20 52 6f 6f 74 20 43 65 72 74 69 66 69 63 61 74 65 20 41 75 74 68 6f 72 69 74 79	common name
30 81 9f	subject public key info header
30 0d 06 09 2a 86 48 86 f7 0d 01 01 01 05 00	public key algorithm type
03 81 8d 00 30 81 89	public key header

02 81 81 00 af d1 86 c8 17 45 02 bc e5 59 b4 15 ac 95 87 7b 89 f5 8b f8 3b 8a 8b ef 67 cf 9e 00 47 d5 f1 06 42 55 36 a1 d1 8c dc cb 81 bb 31 8d 35 f7 6d 11 a0 91 9b 31 3d b9 71 38 46 15 c8 81 c4 51 06 7b d7 8a 70 be c1 28 0d 78 80 3c 44 a6 5e 35 5f 6e 46 2f 80 41 28 78 63 6c 86 cc d0 b3 58 ca bc 07 d5 19 3e 8a a2 1c 7e ff 0d 16 2b 0f bd a5 5e 60 93 64 09 80 24 76 ed e4 a9 e3 81 26 0c de 8a 89	public key modulus
02 03 01 00 01	public key exponent
30 0d 06 09 2a 86 48 86 f7 0d 01 01 05 05 00	signature algorithm
03 81 81 00 81 4d db 31 e2 31 d2 6c f5 21 29 93 4a ce cb 6c fb 8b fc 3d ef 4b e8 4a 8a db f7 d8 e3 70 1d 3c ff ba 71 70 c4 82 24 9f 12 b5 d4 3e 3a 4d 20 64 2f ab 8b 05 27 9a 34 24 33 24 d4 7e bc 41 07 34 7a a6 51 12 29 55 e7 9b 5b e5 6b 79 bb 31 04 2f d1 c6 d3 7f 32 a2 b5 cc 99 23 09 97 1a 21 44 fa 25 3b f4 4b d6 00 cf e9 1b a9 be 9b 88 f8 90 fd 59 77 80 41 7d cb ca bf 81 87 19 61 72 20 19 1e	signature value

Some of the fields in this example are the same in all CA certificates. These fields are:

- version: v3
- signature: SHA-1 with RSA, null parameters
- subject first organizational unit name: “DOCSIS”
- public key algorithm type: RSA encryption, null parameters
- public key exponent: 3-octet integer, value 0x10001
- signature algorithm: SHA-1 with RSA, null parameters

This is an example of a self-signed CA certificate. The issuer name and the subject names are identical. In this example, the matching name fields are:

- country name: “US”
- organization name: “Nortel”
- first organizational unit name: “DOCSIS”
- second organizational unit name: “Building 1, Andover MA”
- common name: “Nortel Cable Modem Root Certificate Authority”

The other fields are example values. Some of these are:

- serial number: integer of 8 octets, value 0x0102030405060708 (other CA certificates may use a different length)
- not before: 1999-01-20 16:05:00 GMT
- not after: 2049-12-31 23:59:55 GMT
- public key modulus: integer of 1024 bits, value 0x00afd1...8a89 (other CA certificates may use an integer of length 1024 to 2048 bits, inclusive)

- signature value: bit string of length 1024 bits, representing the integer value 0x00814d...191e (Other CA certificates may use a bit string of length 1024 to 2048 bits, inclusive; the length matches that of the issuer's modulus. The signature is computed over the portion of the certificate that begins with the tbsCertificate header and ends with the public key exponent, inclusive.)

B.3 Authorization Request

The CM sends the following Authorization Request:

04 72 03 40	Auth Request header
05 00 ad	CM-Identification header
01 00 0c 30 30 30 30 30 30 31 32 33 34 35 36	Serial Number
02 00 03 00 00 ca	Manufacturer ID
03 00 06 00 00 ca 01 04 01	MAC Address
04 00 8c 30 81 89 02 81 81 00 e0 e0 6c 8d be b2 8b c9 f3 a6 3d a1 12 ea f7 99 f7 3d 3e fa a3 b1 e2 42 95 71 b5 71 d2 32 7a da 10 40 e2 5b 09 74 69 08 78 46 37 71 34 3e 69 a7 37 6d f8 70 1d aa a5 34 b0 33 a3 43 ac 4d eb 41 5e 0a 8a fd a6 0a 4b 09 7f 5a 18 f2 9e c2 22 a6 6b 9a 69 73 22 d5 37 c9 63 b0 88 f5 60 5d 99 16 33 54 53 30 ed 35 de 0c 87 3b 54 ba 59 22 3e b2 79 90 96 61 db f3 4a 37 18 4c 7f a8 ca ee d6 31 02 03 01 00 01	RSA Public Key
12 02 7a	CM Cerfiticate header
30 82 02 76 30 82 01 df . . . 19 c9 f1 dc 30 b8 d3 d5	CM Certificate
13 00 0b	Security Capabilities header
15 00 04 01 00 02 00	Cryptographic Suite List
16 00 01 01	BPI Version
0c 00 02 22 60	SAID

The Code field has value 0x04, which identifies this as an Authorization Request packet. The Identifier field has value 0x72; this is an example value. The Length field has value 0x0340 (832), which is the number of octets that follow the Length field.

The first attribute is the CM Identification. It is a compound attribute consisting of the following sub-attributes: Serial Number, Manufacturer ID, MAC Address, and RSA Public Key. Example values are shown for these sub-attributes.

The RSA Public Key is DER-encoded and is similar to the example in section 2.2 of [RSA2]. The modulus is a 1024-bit integer represented using 0x81 (129) octets. In this example, the value of the modulus is:

```
0x00e0e06c8d ... caeed631.
```

Notice that 0x00 is the most significant octet of the modulus and 0x31 is the least significant. The exponent is an integer made up of 3 octets and having value 0x010001.

The next attribute is the CM Certificate. Details of the certificate are given below. Note that some fields of the CM Certificate must match sub-attributes of the CM Identification; these sub-attributes are the MAC Address and RSA Public Key.

The next attribute is the Security Capabilities attribute. It is a compound attribute consisting of the Cryptographic Suite List and the BPI Version. In this example, two Cryptographic Suites are listed: 56-bit DES with no authentication, and 40-bit DES with no authentication. The BPI Version is BPI+.

The final attribute is the CM's Primary SAID, whose value is equal to its Primary SID. In this example, the Primary Said has value 0x2260.

B.3.1 CM Certificate details

The fields of the CM Certificate in the Authorization Info message above break down as follows:

30 82 02 76	certificate header
30 82 01 df	tbsCertificate header
a0 03 02 01 02	version
02 08 01 01 01 01 01 01 01 01	serial number
30 0d 06 09 2a 86 48 86 f7 0d 01 01 05 05 00	signature
30 81 88	issuer header
31 0b 30 09 06 03 55 04 06 13 02 55 53	country name
31 0f 30 0d 06 03 55 04 0a 13 06 4e 6f 72 74 65 6c	organization name
31 0f 30 0d 06 03 55 04 0b 13 06 44 4f 43 53 49 53	organizational unit name
31 1f 30 1d 06 03 55 04 0b 13 16 42 75 69 6c 64 69 6e 67 20 31 2c 20 41 6e 64 6f 76 65 72 20 4d 41	organizational unit name
31 36 30 34 06 03 55 04 03 13 2d 4e 6f 72 74 65 6c 20 43 61 62 6c 65 20 4d 6f 64 65 6d 20 52 6f 6f 74 20 43 65 72 74 69 66 69 63 61 74 65 20 41 75 74 68 6f 72 69 74 79	common name
30 1e	validity header
17 0d 39 39 30 33 32 33 31 36 35 38 33 34 5a	not before
17 0d 34 39 31 32 33 31 32 33 35 39 35 30 5a	not after
30 72	subject header
31 0b 30 09 06 03 55 04 06 13 02 55 53	country name

31 0f 30 0d 06 03 55 04 0a 13 06 4e 6f 72 74 65 6c	organization name
31 1f 30 1d 06 03 55 04 0b 13 16 42 75 69 6c 64 69 6e 67 20 31 2c 20 41 6e 64 6f 76 65 72 20 4d 41	organizational unit name
31 15 30 13 06 03 55 04 03 13 0c 30 30 30 30 30 30 31 32 33 34 35 36	common name (serial number)
31 1a 30 18 06 03 55 04 03 13 11 30 30 3a 30 30 3a 43 41 3a 30 31 3a 30 34 3a 30 31	common name (MAC address)
30 81 9f	subject public key info header
30 0d 06 09 2a 86 48 86 f7 0d 01 01 01 05 00	public key algorithm type
03 81 8d 00 30 81 89	public key header
02 81 81 00 e0 e0 6c 8d be b2 8b c9 f3 a6 3d a1 12 ea f7 99 f7 3d 3e fa a3 b1 e2 42 95 71 b5 71 d2 32 7a da 10 40 e2 5b 09 74 69 08 78 46 37 71 34 3e 69 a7 37 6d f8 70 1d aa a5 34 b0 33 a3 43 ac 4d eb 41 5e 0a 8a fd a6 0a 4b 09 7f 5a 18 f2 9e c2 22 a6 6b 9a 69 73 22 d5 37 c9 63 b0 88 f5 60 5d 99 16 33 54 53 30 ed 35 de 0c 87 3b 54 ba 59 22 3e b2 79 90 96 61 db f3 4a 37 18 4c 7f a8 ca ee d6 31	public key modulus
02 03 01 00 01	public key exponent
30 0d 06 09 2a 86 48 86 f7 0d 01 01 05 05 00	signature algorithm
03 81 81 00 19 b0 2b e5 2c 37 4a af 34 cb c9 59 62 68 88 05 8a 91 5b d4 c6 fa 2e 19 ab 98 42 33 68 9d fc e4 76 23 84 8d 4a be ff bf 34 cf e0 fb 93 96 01 8b 89 d9 86 42 5e cf 6d e6 68 2e 44 99 56 6a cc f1 2c b9 5b 30 21 08 22 f5 11 b1 38 ba 6e b5 62 f0 3a dc f1 2e c4 61 95 2f 16 c8 27 63 b6 e8 69 a6 1c e1 4f 1a 8c 65 cb 57 5e 13 ce db 7f 27 f9 c1 6e bf 2f 75 77 9e a9 87 19 c9 f1 dc 30 b8 d3 d5	signature value

Some of the fields in this example are the same for all CM Certificates. These fields are:

- version: v3
- signature: SHA-1 with RSA, null parameters
- issuer first organizational unit name: “DOCSIS”
- public key algorithm type: RSA encryption, null parameters
- public key exponent: 3-octet integer, value 0x10001
- signature algorithm: SHA-1 with RSA, null parameters

The issuer name of the CM certificate matches the subject name of the CA certificate. In this example, the matching issuer-name fields are:

- country name: “US”
- organization name: “Nortel”
- first organizational unit name: “DOCSIS”

- second organizational unit name: “Building 1, Andover MA”
- common name: “Nortel Cable Modem Root Certificate Authority”

The other fields are example values. Some of these are:

- serial number: integer of 8 octets, value 0x0101010101010101 (other CM certificates may use a different length)
- not before: 1999-03-23 16:58:34 GMT
- not after: 2049-12-31 23:59:50 GMT
- subject country name: “US”
- subject organization name: “Nortel”
- subject organizational unit name: “Building 1, Andover MA”
- subject first common name (serial number): “000000123456” (other CM certificates may use a different length string. The value matches the Serial Number attribute of the Authorization Request message)
- subject second common name (MAC address): “00:00:CA:01:04:01” (All CM certificates use a string of this length. The value matches the MAC Address attribute of the Authorization Request message.)
- public key modulus: integer of length 1024 bits, value 0x00e0e0...d631 (other CM certificates may use an integer of length 768 or 1024 bits)
- signature value: bit string of length 1024 bits, representing the integer value 0x0019b0...d3d5 (Other CM certificates may use a bit string of length 1024 to 2048 bits, inclusive; the length matches that of the issuer’s modulus. The signature is computed over the portion of the certificate that begins with the tbsCertificate header and ends with the public key exponent, inclusive.)

B.4 Authorization Reply

The CMTS sends the following Authorization Reply:

05 72 00 9f	Auth Reply header
07 00 80 a2 cb ad c8 34 27 71 47 06 d5 10 0c 07 94 90 bf e6 44 1b 0c 90 0d b4 ed 9c 39 aa 05 a0 c1 ef 54 4b cc fb 3a 7a 22 81 c0 dc c6 6e 39 a4 91 1c ba bf b0 ed 47 10 f2 f4 13 f9 09 33 c6 ae a3 45 67 c8 38 0f c3 9a 12 be d5 27 27 39 77 fb 98 03 39 50 39 99 f5 b6 ad b5 85 f9 16 d0 ff c6 2a ff 9f 38 73 6f 35 44 21 ad 9e e1 a5 91 4d 34 06 1d bb c9 b6 8f 8a 17 9e be c6 c9 40 eb 81 f0 62 d8 18	Auth Key
09 00 04 00 09 3a 80	Key Lifetime
0a 00 01 07	Key Sequence number
17 00 0e	SA Descriptor header
0c 00 02 22 60	SAID
18 00 01 00	SA Type
14 00 02 01 00	Cryptographic Suite

The Code field has value 0x05, which identifies this as an Authorization Reply packet. The Identifier field has value 0x72, matching the Identifier field of the Authorization Request. The Length field has value 0x009f (159), which is the number of octets that follow the Length field.

The first attribute is the Authorization Key. The attribute contains an authorization key which has been RSA-encrypted using the public key in the Authorization Request message. The RSA-encrypted authorization key is an integer made up of 0x80 (128) octets. In this example, the value of the RSA-encrypted authorization key is:

0xa2cbadc8 ... f062d818.

Notice that 0xa2 is the most significant octet of the RSA-encrypted authorization key and 0x18 is the least significant. Details of the RSA encryption calculation are given below.

The second attribute is the Key Lifetime. In this example, the value is 0x00093a80 (604800) seconds, or 7 days.

The third attribute is the Key Sequence Number. In this example, the value is 0x07.

The remaining attributes are SA Descriptors. Each SA Descriptor is a compound attribute consisting of the following sub-attributes: SAID, SA Type, and Cryptographic Suite. In this example, a single SA Descriptor is included, corresponding to the SAID in the Authorization Request. The SA Type is Primary, and the Cryptographic Suite is 56-bit DES with no authentication.

The CM and CMTS each derive a key encryption key and two message authentication keys from the authorization key, using hashing. Details of the hashing calculations are given below. Here are the values of these keys for this example:

Authorization key	4e 85 27 ff c4 12 72 8e 61 84 de c9 20 b6 e0 64 f0 bc 0b 75
Key encryption key	76 b4 d4 2f 14 98 59 6a ab fe 72 94 15 7c 7d 62
Message authentication key, upstream	fe b9 f1 e2 46 a7 6d 7c a7 7b 5e b0 98 25 fd 0b 57 ca 90 c7
Message authentication key, downstream	93 d3 9d 70 c3 b6 f5 92 c4 6b d3 92 76 46 f4 f1 90 3a 52 fd

B.4.1 RSA encryption details

The CMTS generates a random authorization key of 20 octets. In this example, the value of the authorization key is:

```
4e 85 27 ff c4 12 72 8e 61 84 de c9 20 b6 e0 64 f0 bc 0b 75
```

The authorization key is encrypted using the RSAES-OAEP scheme in [RSA3]. This section gives details of the scheme as applied to this example. The scheme makes use of a mask-generating function (MGF) which is based on hashing; details are given in a later section.

The authorization key is padded into a 107-octet block DB:

```
DB =

da 39 a3 ee 5e 6b 4b 0d 32 55 bf ef 95 60 18 90 af d8 07 09 00 00
00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 01 42
85 27 ff c4 12 72 8e 61 84 de c9 20 b6 e0 64 f0 bc ob 75
```

To form DB, the authorization key is prefaced with an octet of value 1, and the result is placed in the last 21 octets of the block. The first 20 octets of the block are the result of performing a hash operation on a zero-length string; these 20 octets have the same value in every Authorization Reply and are not unique to this example. The remaining 66 octets of the block are set to 0.

The CMTS generates a random string of 20 octets called the SEED. The SEED is independently generated for each Authorization Reply. In this example, the SEED has value:

```
SEED =

ad 9c af 8d f8 26 fe af b5 df fd 95 de 7e 97 cc e9 4b 6d 6d
```

The SEED is input to the MGF to generate DB_MASK, a block of 107 octets:

```
DB_MASK =

de 10 c9 59 41 c9 ea 72 a4 35 68 79 d2 53 85 db 13 7b a6 3b 37
ac 86 06 7c b5 ec 97 d2 d0 9e 01 30 2b 10 91 3a ec 3f d9 a1 2f c4 e9
8d 18 88 95 f6 9c ea 17 23 9f 5d d5 f1 4d 25 8e 9e 6d 7d 3c ca
55 fe 0e ee 2d 0d 7e 5b 64 b6 79 44 76 c 3f 6e ac 99 3a ae 14 3e
9a 8e df 3c 36 79 58 b2 fa 13 72 58 4c ca 04 a1 af c7 c4 62
```

DB and DB_MASK are exclusive-or'd together to produce MASKED_DB, which has 107 octets:

```
MASKED_DB =

04 29 6a b7 1f a2 a1 7f 96 60 d7 96 47 33 9d 2d bc a3 a1 32 37 ac
86 06 7c b5 ec 97 d2 d0 9e 01 30 2b 10 91 3a ec 3f d9 a1 2f c4 e9
8d 18 88 95 f6 9c ea 17 23 9f 5d d5 f1 4d 25 8e 9e 6d 7d 3c ca 55
fe 0e ee 2d 0d 7e 5b 64 b6 79 44 76 cc 3f 6e ac 99 3a ae 14 3f d4
0b f8 c3 f2 6b 2a 3c 9b 97 ac 91 6c 7c e4 c5 5f 7b cf 17
```

MASKED_DB is input to the MGF to generate SEED_MASK, a block of 20 octets:

```
SEED_MASK =

b4 b6 f1 bf a6 b3 a1 7e 95 82 d3 b8 93 71 b6 7f 45 31 9e 82
```

SEED and SEED_MASK are exclusive-or'd together to produce MASKED_SEED, which has 20 octets:

```
MASKED_SEED =

19 2a 5e 32 5e 95 5f d1 20 5d 2e 2d 4d 0f 21 b3 ac 7a f3 ef
```

MASKED_SEED and MASKED_DB are concatenated, and the result is prefaced with a single octet of value 0. This results in a 128-octet block called EM:

```
EM =

00 19 2a 5e 32 5e 95 5f d1 20 5d 2e 2d 4d 0f 21 b3 ac 7a f3 ef 04
29 6a b7 1f a2 a1 7f 96 60 d7 96 47 33 9d 2d bc a3 a1 32 37 ac 86
06 7c b5 ec 97 d2 d0 9e 01 30 2b 10 91 3a ec 3f d9 a1 2f c4 e9 8d
18 88 95 f6 9c ea 17 23 9f 5d d5 f1 4d 25 8e 9e 6d 7d 3c ca 55 fe
0e ee 2d 0d 7e 5b 64 b6 79 44 76 cc 3f 6e ac 99 3a ae 14 3f d4 0b
f8 c3 f2 6b 2a 3c 9b 97 ac 91 6c 7c e4 c5 5f 7b cf 17
```

To perform RSA encryption, EM is interpreted as the integer value:

0x00192a5e32 ... 5f7bcf17 .

Notice that 0x00 is the most significant octet and 0x17 is the least significant.

The RSA encryption is performed as the operation $Y = M^E \bmod N$, where:

M is the integer value of the block EM (0x00192a5e32 ... 5f7bcf17); E is the integer value of the exponent of the RSA public key (0x010001); N is the integer value of the modulus of the RSA public key (oxe0e06c8d ... caeed631); Y is the integer value of the RSA-encrypted authorization key (0xa2cbadc8 ... f062d818).

B.4.2 RSA decryption details

Here is a table that lists the private-key parameters that match the RSA public key in the example Authorization Request message:

Parameter	Property	Value
D (private exponent)	$M^{DE} \bmod N = M$	6b 1f 1d 36 ec 77 7b 15 a9 c6 30 27 71 ae 92 62 3a 9f 67 47 d8 00 9d ca a0 0b f9 a6 0d be 54 3d 5a 6e be 25 25 bc d9 67 da 7b 80 5f a1 c6 75 67 dd 84 ba 4b 16 26 ba e9 fd 61 ab cd 49 e0 18 47 37 9f 56 08 2d d9 16 81 ff 7d d0 7e 01 8f d4 84 d3 e8 eb 27 48 c3 6c dc a9 01 b7 e5 24 28 d1 6c 67 03 a7 63 fb fa 79 d8 08 6a e1 de 3d 12 7a 36 20 25 01 d1 08 11 0c cd 80 44 3c fd c5 c4 db d1
P (prime factor)	N = PQ	f1 6b dd 2f dd d8 df 80 30 e6 9c d3 4e 46 5e 9f 42 62 b1 66 86 57 1b ca 87 9c cf fd 1c b6 26 76 95 35 bf 0b fb 51 af 0f 46 1c 5e cb 82 a0 83 bf 46 c9 3b d6 4e 7a 5d bf 03 05 69 27 31 6d 65 bd

Q (prime factor)	N = PQ	ee 74 cb a3 d0 90 2d 8a e9 e7 10 dd b4 65 2e 91 22 09 52 72 ab bd 32 31 4e d7 d0 2b 4b 13 57 20 6b f9 a4 57 b1 47 59 67 86 a6 8c 2c c1 f3 8b ba 8a 6b b1 62 5d 43 5a 71 db d0 33 43 97 99 17 85
D_p (CRT exponent)	$D_p = D \bmod (P - 1)$	a6 35 dc d2 57 aa 38 35 c9 74 fc 03 7e a0 74 04 b1 6f c1 33 14 ca 64 17 cb c5 ea 6c 18 98 4f 62 d4 d7 6b f0 93 d6 68 ef db 15 2d 2e 6f 80 93 33 dd 48 2e 2a 1d 5d a1 ad 20 27 59 7d e2 49 af 01
D_q (CRT exponent)	$D_q = D \bmod (Q - 1)$	cf f1 9c 30 33 cd b7 59 7f 96 57 f7 ee bb 99 bb 48 a2 36 7a f7 57 1a f1 32 df 32 92 be 7a 94 2d 1a db ed bb e7 45 e0 2a 4e 9a e8 7c 93 7a 4e 2c 93 4f 4c b6 09 bc 95 9f da df 9a 04 e4 ab c5 7d
U_p (CRT constant)	$PU_p \bmod Q = 1$	08 17 0c 11 bc aa 2f 96 80 8b 31 95 6d 2e b8 3c ee 2e 05 88 ab 9e fc 53 24 c4 04 b8 7e 1d 01 db 2d f2 2c 06 b0 cd 04 6b 1c 14 d8 d0 4f c9 a0 ae 1b c9 80 88 be 42 0a 52 4a ef 62 3c 8b dd c5 37

Each value in the table represents the octets of an integer, with the most significant octet shown first. For example, the private exponent D has the integer value:

0x6b1f1d36 ... c5c4dbd1.

The CM can decrypt the authorization key with or without using the Chinese Remainder Theorem (CRT). Decryption using the CRT is more complicated, but it may be a faster operation.

To decrypt without using the CRT, the CM performs the operation $M = Y^D \bmod N$. D is the private exponent in the table, and Y and N are as described in the preceding section. The resulting value matches the value of M in the preceding section, that is, it is the integer value the block EM formed by the CMTS. The CM decodes the authorization key from EM by inverting the procedure used by the CMTS to form EM, as described in [RSA3].

To decrypt using the CRT, the CM first computes two intermediate quantities:

$A = Y^{Dp} \bmod P;$

$B = Y^{Dq} \bmod Q.$

P and Q are the prime factors of the modulus, and D_p and D_q are private exponents related to these factors, all with values shown in the table. The CM computes the value of M as:

$M = A + ((B - A)U_p \bmod Q)P;$

U_p is a constant derived from the prime factors, with value as shown in the table. The resulting value of M matches the value that would be computed using the operation $M = Y^D \bmod N$.

B.4.3 Hashing details

The authorization key is hashed using the SHA-1 algorithm [FIPS-180-1] to produce the key encryption key (KEK), the message authentication key for upstream, and the message authentication key for downstream.

The discussion here represents a hash calculation using a table that shows the input to the hash function and the resulting hash value. For reference, here is such a table that describes the example in Appendix B of [FIPS-180-1]:

Hash input	61 62 63 64 62 63 64 65 63 64 65 66 64 65 66 67 65 66 67 68 66 67 68 69 67 68 69 6a 68 69 6a 6b 69 6a 6b 6c 6a 6b 6c 6d 6b 6c 6d 6e 6c 6d 6e 6f 6d 6e 6f 70 6e 6f 70 71
Hash value	84 98 3e 44 1c 3b d2 6e ba ae 4a a1 f9 51 29 e5 e5 46 70 f1

B.4.3.1 KEK

The KEK is computed using the following hash calculation:

Hash input	53 53 53 53 53 53 53 53 53 53 53 53 53 53 53 53 53 53 53 53 53 53 53 53 53 53 53 53 53 53 53 53 53 53 53 53 53 53 53 53 53 53 53 53 53 53 53 53 53 53 53 53 53 53 53 53 53 53 53 53 53 53 53 53 4e 85 27 ff c4 12 72 8e 61 84 de c9 20 b6 e0 64 f0 bc 0b 75
Hash value	76 b4 d4 2f 14 98 59 6a ab fe 72 94 15 7c 7d 62 b0 df e6 3b

The input is the octet 0x53, repeated 64 times, followed by the 20 octets of the authorization key. The order in which the octets of the authorization key are digested is the same as the order in which they appear in the EM encryption block.

The hash value is 20 bytes long. The first 16 bytes are the KEK.

B.4.3.2 Message authentication keys

The upstream message authentication key is computed using the following hash calculation:

Hash input	5c 5c 5c 5c 5c 5c 5c 5c 5c 5c 5c 5c 5c 5c 5c 5c 5c 5c 5c 5c 5c 5c 5c 5c 5c 5c 5c 5c 5c 5c 5c 5c 5c 5c 5c 5c 5c 5c 5c 5c 5c 5c 5c 5c 5c 5c 5c 5c 5c 5c 5c 5c 5c 5c 5c 5c 5c 5c 5c 5c 5c 5c 5c 5c 4e 85 27 ff c4 12 72 8e 61 84 de c9 20 b6 e0 64 f0 bc 0b 75
Hash value	fe b9 f1 e2 46 a7 6d 7c a7 7b 5e b0 98 25 fd 0b 57 ca 90 c7

The input is the octet 0x5c, repeated 64 times, followed by the 20 octets of the authorization key. The order in which the octets of the authorization key are digested is the same as in the KEK calculation.

The hash value is 20 octets long. The 20 octets make up the upstream message authentication key.

The downstream message authentication key is computed using the following hash calculation:

Hash input	3a 3a 3a 3a 3a 3a 3a 3a 3a 3a 3a 3a 3a 3a 3a 3a 3a 3a 3a 3a 3a 3a 3a 3a 3a 3a 3a 3a 3a 3a 3a 3a 3a 3a 3a 3a 3a 3a 3a 3a 3a 3a 3a 3a 3a 3a 3a 3a 3a 3a 3a 3a 3a 3a 3a 3a 3a 3a 3a 3a 3a 3a 3a 3a 4e 85 27 ff c4 12 72 8e 61 84 de c9 20 b6 e0 64 f0 bc 0b 75
Hash value	93 d3 9d 70 c3 b6 f5 92 c4 6b d3 92 76 46 f4 f1 90 3a 52 fd

This is similar to the computation for the upstream case, except that value 0x3a replaces value 0x5c.

B.4.3.3 Mask-generation function

The mask-generation function (MGF) is built out of SHA-1 hash operations. Each hash operation generates 20 octets of mask data. The number of hash operations performed depends on the size of the mask that is needed.

Quantity SEED_MASK is formed by applying the MGF to MASKED_DB. Since SEED_MASK is 20 octets long, this requires only one hash operation:

Hash input	04 29 6a b7 1f a2 a1 7f 96 60 d7 96 47 33 9d 2d bc a3 a1 32 37 ac 86 06 7c b5 ec 97 d2 d0 9e 01 30 2b 10 91 3a ec 3f d9 a1 2f c4 e9 8d 18 88 95 f6 9c ea 17 23 9f 5d d5 f1 4d 25 8e 9e 6d 7d 3c ca 55 fe 0e ee 2d 0d 7e 5b 64 b6 79 44 76 cc 3f 6e ac 99 3a ae 14 3f d4 0b f8 c3 f2 6b 2a 3c 9b 97 ac 91 6c 7c e4 c5 5f 7b cf 17 00 00 00 00
Hash value	b4 b6 f1 bf a6 b3 a1 7e 95 82 d3 b8 93 71 b6 7f 45 31 9e 82

The input data to the hash operation are the 107 octets MASKED_DB followed by four octets of value 0. The output of the hash operation is the value of SEED_MASK.

Quantity DB_MASK is formed by applying the MGF to SEED. Since DB_MASK is 107 octets long, this requires six hash operations:

Hash input	ad 9c af 8d f8 26 fe af b5 df fd 95 de 7e 97 cc e9 4b 6d 6d 00 00 00 00
Hash value	de 10 c9 59 41 c9 ea 72 a4 35 68 79 d2 53 85 bd 13 7b a6 3b

Hash input	ad 9c af 8d f8 26 fe af b5 df fd 95 de 7e 97 cc e9 4b 6d 6d 00 00 00 01
Hash value	37 ac 86 06 7c b5 ec 97 d2 d0 9e 01 30 2b 10 91 3a ec 3f d9

Hash input	ad 9c af 8d f8 26 fe af b5 df fd 95 de 7e 97 cc e9 4b 6d 6d 00 00 00 02
Hash value	a1 2f c4 e9 8d 18 88 95 f6 9c ea 17 23 9f 5d d5 f1 4d 25 8e

Hash input	ad 9c af 8d f8 26 fe af b5 df fd 95 de 7e 97 cc e9 4b 6d 6d 00 00 00 03
Hash value	9e 6d 7d 3c ca 55 fe 0e ee 2d 0d 7e 5b 64 b6 79 44 76 cc 3f

Hash input	ad 9c af 8d f8 26 fe af b5 df fd 95 de 7e 97 cc e9 4b 6d 6d 00 00 00 04
Hash value	6e ac 99 3a ae 14 3e 9a 8e df 3c 36 79 58 b2 fa 13 72 58 4c

Hash input	ad 9c af 8d f8 26 fe af b5 df fd 95 de 7e 97 cc e9 4b 6d 6d 00 00 00 05
Hash value	ca 04 a1 af c7 c4 62 3a df 6f 33 ec e2 cd 2c 7f b7 7e 48 19

The input data to each hash operation are the 20 octets of SEED followed by a four-octet value. The four-octet value counts the integer values 0, 1, 2, 3, 4, 5 on successive hash operations. The outputs of the six hash operations are concatenated into a 120-octet result, and the first 107 octets of the result make up DB_MASK.

B.5 Key Request

The CM sends the following Key Request:

07 73 00 d0	Key Request Header
05 00 ad	CM-Identification header
01 00 0c 30 30 30 30 30 30 31 32 33 34 35 36	Serial Number
02 00 03 25 53 41	Manufacturer ID
03 00 06 00 00 ca 01 04 01	MAC Address
04 00 8c 30 81 89 02 81 81 00 e0 e0 6c 8d be b2 8b c9 f3 a6 3d a1 12 ea f7 99 f7 3d 3e fa a3 b1 e2 42 95 71 b5 71 d2 32 7a da 10 40 e2 5b 09 74 69 08 78 46 37 71 34 3e 69 a7 37 6d f8 70 1d aa a5 34 b0 33 a3 43 ac 4d eb 41 5e 0a 8a fd a6 0a 4b 09 7f 5a 18 f2 9e c2 22 a6 6b 9a 69 73 22 d5 37 c9 63 b0 88 f5 60 5d 99 16 33 54 53 30 ed 35 de 0c 87 3b 54 ba 59 22 3e b2 79 90 96 61 db f3 4a 37 18 4c 7f a8 ca ee d6 31 02 03 01 00 01	RSA public key
0a 00 01 07	Key Sequence Number
0c 00 02 22 60	SAID
0b 00 14 86 b8 33 b7 48 9c 4b a1 51 67 44 d7 a6 e6 ca 21 33 f5 22 9e	HMAC digest

The Code field has value 0x07, which identifies this as a Key Request packet. The Identifier field has value 0x73; this is an example value, obtained by incrementing the Identifier value in the Authorization Request. The Length field has value 0x00d0 (208), which is the number of octets that follow the Length field.

The first attribute is the CM Identification. This is a compound attribute, identical to that in the Authorization Request.

The second attribute is the Key Sequence Number, which identifies the authorization key. The value is identical to that in the Authorization Reply.

The third attribute is the SAID for which a key is being requested. This SAID value was contained in the Authorization Reply.

The final attribute is the HMAC Digest. The digest consists of 20 octets. It is computed using the upstream message authentication key. The digest is performed over all octets of the Key Request packet, excluding the 23 octets of the HMAC Digest attribute itself. Details of the digest calculation are given below.

B.5.1 HMAC digest details

The HMAC digest is computed using the HMAC authentication method defined in [RFC2104], with SHA-1 as the hash function. Example calculations of HMAC using SHA-1 are presented in [RFC2202].

The discussion here represents an HMAC calculation using a table that shows the key, the input to the HMAC function, and the resulting HMAC digest. For reference, here is a table that describes test case #2 of the HMAC-SHA-1 examples in [RFC2202]:

Key	4a 65 66 65
HMAC input	77 68 61 74 20 64 6f 20 79 61 20 77 61 6e 74 20 66 6f 72 20 6e 6f 74 68 69 6e 67 3f
HMAC digest	ef fc df 6a e5 eb 2f a2 d2 74 16 d5 f1 84 df 9c 25 9a 7c 79

The HMAC digest of the Key Request packet is computed using the following HMAC calculation:

Key	fe b9 f1 e2 46 a7 6d 7c a7 7b 5e b0 98 25 fd 0b 57 ca 90 c7
HMAC input	07 73 00 d0 05 00 ad 01 00 0c 30 30 30 30 30 30 31 32 33 34 35 36 02 00 03 25 53 41 03 00 06 00 00 ca 01 04 01 04 00 8c 30 81 89 02 81 81 00 e0 e0 6c 8d be b2 8b c9 f3 a6 3d a1 12 ea f7 99 f7 3d 3e fa a3 b1 e2 42 95 71 b5 71 d2 32 7a da 10 40 e2 5b 09 74 69 08 78 46 37 71 34 3e 69 a7 37 6d f8 70 1d aa a5 34 b0 33 a3 43 ac 4d eb 41 5e 0a 8a fd a6 0a 4b 09 7f 5a 18 f2 9e c2 22 a6 6b 9a 69 73 22 d5 37 c9 63 b0 88 f5 60 5d 99 16 33 54 53 30 ed 35 de 0c 87 3b 54 ba 59 22 3e b2 79 90 96 61 db f3 4a 37 18 4c 7f a8 ca ee d6 31 02 03 01 00 01 0a 00 01 07 0c 00 02 22 60
HMAC digest	86 b8 33 b7 48 9c 4b a1 51 67 44 d7 a6 e6 ca 21 33 f5 22 9e

The key is the upstream message authentication key. The input consists of all octets of the Key Request packet, excluding the HMAC Digest attribute. The octets of the digest are the contents of the HMAC Digest attribute.

B.6 Key Reply

The CMTS sends the following Key Reply:

08 73 00 68	Key Reply header
0a 00 01 07	Key Sequence Number (authorization key)
0c 00 02 22 60	SAID
0d 00 21	TEK Parameters header
08 00 08 b6 4d 54 8c 3f 6b 25 69	TEK Key
09 00 04 00 00 a8 c0	Key Lifetime
0a 00 01 02	Key Sequence Number (TEK)

0f 00 08 81 0e 52 8e 1c 5f da 1a	DES CBC IV
0d 00 21	TEK Parameters header
08 00 08 5e bd 03 aa 5e d5 e2 94	TEK Key
09 00 04 00 01 51 80	Key Lifetime
0a 00 01 03	Key Sequence Number (TEK)
0f 00 08 25 35 67 c3 09 21 8c 2c	DES CBC IV
0b 00 14 a5 e3 33 25 ea 72 f8 50 1c 2a b6 65 45 6b cc de 8b 4f 22 02	HMAC Digest

The Code field has value 0x08, which identifies this as a Key Reply packet. The Identifier has 0x73, matching the value in the Key Request. The Length field has value 0x68 (104), which is the number of octets that follow the Length field.

The Key Sequence Number attribute identifies the authorization key. It matches the value in the Key Request.

The SAID attribute identifies the SAID for with a TEK is being supplied. It matches the value in the Key Request.

Two TEK Parameters attributes are included, the first for the older generation of key parameters and the second for the newer. Each TEK Parameters attribute is a compound attribute consisting of the following sub-attributes: TEK Key, Key Lifetime, Key Sequence Number, and DES CBC IV.

The TEK Key consists of 8 octets. It contains the TEK, encrypted using triple-DES-ECB with the KEK derived from the authorization key. Details of the triple-DES-ECB calculation are given below.

The Key Lifetime sub-attribute refers to the TEK. In this example, the value for the older TEK is 0x0000a8c0 (43200) seconds, or 12 hours, and the value for the newer TEK is 0x00015180 (86400) seconds, or 24 hours.

The Key Sequence Number sub-attribute identifies the TEK. In this example, the value for the older TEK is 0x02, and the value for the newer TEK is 0x03.

The DES CBC IV sub-attribute consists of 8 octets. It specifies the Initialization Vector to be used with the TEK.

The final attribute is the HMAC Digest. It consists of 20 octets. It is computed in a manner similar to that in the Key Reply, except that the downstream message authentication key is used instead of the upstream key. Details of the HMAC calculation are given below.

After the CM processes the Key Reply packet, the CM and CMTS each share two generations of TEK and IV. Here are the values of these parameters for this example:

Older TEK	e6 60 0f d8 85 2e f5 ab
Older IV	81 0e 52 8e 1c 5f da 1a
Newer TEK	b1 d7 4f c9 64 68 f7 58
Newer IV	25 35 67 c3 09 21 8c 2c

B.6.1 TEK encryption details

The CMTS generates a random TEK of 8 octets. In this example, the value of the TEK is:

```
e6 60 0f d8 85 2e f5 ab.
```

This is the first TEK of the Key Reply message.

The TEK is encrypted using triple-DES-ECB encryption. The encryption key is the KEK:

```
76 b4 d4 2f 14 98 59 6a ab fe 72 94 15 7c 7d 62.
```

Triple-DES-ECB encryption is described here in terms of several iterations of DES-ECB encryption or decryption. DES-ECB is defined in [FIPS-81].

The discussion here represents a DES-ECB encryption or decryption operation using a table that shows the key, the input, and the output. For reference, here are tables that describe the example in Table B1 of [FIPS-81]:

Mode	ECB encryption
Key	01 23 45 67 89 ab cd ef
DES input	4e 6f 77 20 69 73 20 74
DES output	3f a4 0e 8a 98 4d 48 15

Mode	ECB decryption
Key	01 23 45 67 89 ab cd ef
DES input	3f a4 0e 8a 98 4d 48 15
DES output	4e 6f 77 20 69 73 20 74

Note: [FIPS-81] calls for the least significant bit of each octet in the key to be adjusted so that the octet has odd parity. This is evident in the key in the above example. The BPKM protocol does not require odd parity. BPKM generates and distributes 8-octet DES keys of arbitrary parity, and it requires that implementations ignore the value of the least significant bit of each octet.

The TEK is triple-DES-ECB encrypted using the following three DES-ECB operations:

Mode	ECB encryption
Key	76 b4 d4 2f 14 98 59 6a
DES input	e6 60 0f d8 85 2e f5 ab
DES output	c3 94 31 f5 8d f9 1d bf

Mode	ECB decryption
Key	ab fe 72 94 15 7c 7d 62
DES input	c3 94 31 f5 8d f9 1d bf
DES output	44 b0 94 4e ab 04 4c 23

Mode	ECB encryption
Key	76 b4 d4 2f 14 98 59 6a
DES input	44 b0 94 4e ab 04 4c 23
DES output	b6 4d 54 8c 3f 6b 25 69

The first and third operations are DES-ECB encryption; the key for each is the first eight octets of the KEK. The second operation is DES-ECB decryption; the key is the last eight octets of the KEK. The input to the first operation is the TEK to be encrypted. The input to the second operation is the output of the first, and the input to the third operation is the output of the second. The output of the third operation is the encrypted TEK; this is conveyed in the TEK Key sub-attribute of the Key Reply message.

B.6.2 HMAC details

The HMAC digest of the Key Reply packet is computed by a method similar to that of the Key Request packet. The key is the downstream message authentication key. Here are the details of the HMAC calculation:

Key	93 d3 9d 70 c3 b6 f5 92 c4 6b d3 92 76 46 f4 f1 90 3a 52 fd
HMAC input	08 73 00 68 0a 00 01 07 0c 00 02 22 60 0d 00 21 08 00 08 b6 4d 54 8c 3f 6b 25 69 09 00 04 00 00 a8 c0 0a 00 01 02 0f 00 08 81 0e 52 8e 1c 5f da 1a 0d 00 21 08 00 08 5e bd 03 aa 5e d5 e2 94 09 00 04 00 01 51 80 0a 00 01 03 0f 00 08 25 35 67 c3 09 21 8c 2c
HMAC digest	a5 e3 33 25 ea 72 f8 50 1c 2a b6 65 45 6b cc de 8b 4f 22 02

B.7 Packet PDU encryption

The first 12 octets of the Packet PDU, containing the Ethernet/802.3 destination and source addresses (DA/SA), are not encrypted. The remaining octets of the Packet PDU are encrypted using DES-CBC mode with special handling of residual termination blocks that are less than 64 bits. The combination of DES-CBC and residual block processing ensures that the encryption does not change the length of the packet. The encryption key is the TEK corresponding to the key sequence number of the packet's Privacy Extended Header.

The specification describes the residual block processing as follows:

> "Given a final block having n bits, where n is less than 64, the next-to-last ciphertext block is DES encrypted a second time, using the ECB mode, and the least significant n bits of the result are exclusive ORed with the final n bits of the payload to generate the short final cipher block. ... In the special case where the Packet Data PDU payload is less than 64 bits, the initialization vector is DES encrypted, and the leftmost n bits of the resulting ciphertext corresponding to the number of bits of the payload are exclusive ORed with the n bits of the payload to generate the short cipher block."

An alternative description of this procedure, which is equivalent to the description in the specification, is as follows:

> Given a final block having n bits, where n is less than 64, the n bits are padded up to a block of 64 bits by appending 64-n bits of arbitrary value to the right of the n payload bits. The resulting block is DES encrypted using the CFB64 mode, with the next-to-last ciphertext block serving as initialization vector for the CFB64 operation. The leftmost n bits of the resulting ciphertext are used as the short cipher block. ... In the special case where the Packet Data PDU payload is less than 64 bits, the procedure is the same as for a short final block, with the provided initialization vector serving as the initialization vector for the DES-CFB64 operation.

The alternative description produces the same ciphertext as does the description in the specification. In the alternative description, however, no mention is made of combining ECB encryption with exclusive ORing. These operations are internal to CFB64, just as they are internal to CBC. The alternative description is convenient here because it allows residual block processing to be illustrated using CFB64 examples in [FIPS-81].

The Packet PDU includes the DA, SA, and Type/Len fields. In the examples here, no effort is made to use correct values for these fields. As a result, the examples here are not valid packets suitable for transmission. The intent of the examples is to illustrate encryption details only.

In these examples, the TEK and IV are taken from the example Key Reply packet described above.

B.7.1 CBC only

When the number of octets to be encrypted is a multiple of 8, the encryption mode is DES-CBC as defined in [FIPS-81]. The encryption key and IV are as conveyed in the Key Reply packet.

The discussion here represents a DES-CBC encryption using a table that shows the key, IV, plaintext input, and ciphertext output. For reference, here is a table that describes the example in Table C1 of [FIPS-81]:

Mode	CBC
Key	01 23 45 67 89 ab cd ef
IV	12 34 56 78 90 ab cd ef
Plaintext	4e 6f 77 20 69 73 20 74 68 65 20 74 69 6d 65 20
Ciphertext	e5 c7 cd de 87 2b f2 7c 43 e9 34 00 8c 38 9c 0f

Suppose that the Packet PDU, prior to encryption, is as follows:

DA	01 02 03 04 05 06
SA	f1 f2 f3 f4 f5 f6
Type/Len	00 01
User Data	02 03 04 05 06 07 08 09 0a 0b
CRC	88 41 65 06

The DES-CBC encryption is performed as follows:

Mode	CBC
Key	e6 60 0f d8 85 2e f5 ab
IV	81 0e 52 8e 1c 5f da 1a
Plaintext	00 01 02 03 04 05 06 07 08 09 0a 0b 88 41 65 06
Ciphertext	0d da 5a cb d0 5e 55 67 9f 04 d1 b6 41 3d 4e ed

The Packet PDU, after encryption, looks like this:

DA	01 02 03 04 05 06
SA	f1 f2 f3 f4 f5 f6
Type/Len	0d da
User Data	5a cb d0 5e 55 67 9f 04 d1 b6
CRC	41 3d 4e ed

B.7.2 CBC with residual block processing

When the number of octets to be encrypted is greater than 8 and is not a multiple of 8, the encryption mode is a combination of DES-CBC and DES-CFB64.

Encryption begins in DES-CBC mode. DES-CBC is used to process as many complete DES blocks as are present. The encryption key and IV are as conveyed in the Key Reply packet.

After the DES-CBC encryption, there are 1 to 7 octets which have not been encrypted. These octets are encrypted using DES-CFB64 mode. DES-CFB64 is “64-bit Cipher Feedback Mode,” defined in [FIPS-81]. The encryption key is as in the Key Reply packet. The IV is the last 8 octets of ciphertext produced by the DES-CBC processing.

The discussion here represents a DES-CFB64 encryption using a table that shows the key, IV, plaintext input, and ciphertext output. For reference, here is a table that describes the example in Table D3 of [FIPS-81]:

Mode	CFB64
Key	01 23 45 67 89 ab cd ef
IV	12 34 56 78 90 ab cd ef
Plaintext	4e 6f 77 20 69 73 20 74 68 65 20 74 69 6d 65 20
Ciphertext	f3 09 62 49 c7 f4 6e 51 a6 9e 83 9b 1a 92 f7 84

Suppose that the Packet PDU, prior to encryption, is as follows:

DA	01 02 03 04 05 06
SA	f1 f2 f3 f4 f5 f6
Type/Len	00 01
User Data	02 03 04 05 06 07 08 09 0a 0b 0c 0d 0e
CRC	91 d2 d1 9f

The total number of octets to be encrypted is 19. The first 16 octets are processed using DES-CBC encryption, and the last 3 octets using DES-CFB64 encryption.

The DES-CBC encryption is performed as follows:

Mode	CBC
Key	e6 60 0f d8 85 2e f5 ab
IV	81 0e 52 8e 1c 5f da 1a
Plaintext	00 01 02 03 04 05 06 07 08 09 0a 0b 0c 0d 0e 91
Ciphertext	0d da 5a cb d0 5e 55 67 51 47 46 86 8a 71 e5 77

The DES-CFB64 encryption is performed as follows:

Mode	CFB64
Key	e6 60 0f d8 85 2e f5 ab
IV	51 47 46 86 8a 71 e5 77
Plaintext	d2 d1 9f 00 00 00 00 00
Ciphertext	ef ac 88 e8 ee 80 33 14

The key is the same as used for the DES-CBC encryption operation. The IV is the last 8 octets of ciphertext generated by the DES-CBC operation.

Notice that 5 octets of value 0 have been appended to the 3 plaintext octets. The values of these appended plaintext octets have no effect on the values of the first 3 ciphertext octets, which are the only ciphertext octets we are interested in. Arbitrary values can be used instead of 0 for the appended plaintext octets.

The Packet PDU, after encryption, looks like this:

DA	01 02 03 04 05 06
SA	f1 f2 f3 f4 f5 f6
Type/Len	0d da
User Data	5a cb d0 5e 55 67 51 47 46 86 8a 71 e5
CRC	77 ef ac 88

B.7.3 Runt frame

When the number of octets to be encrypted is less than 8, the encryption mode is DES-CFB64. The encryption key and IV are as conveyed in the Key Reply packet.

Suppose that the Packet PDU, prior to encryption, is as follows:

DA	01 02 03 04 05 06
SA	f1 f2 f3 f4 f5 f6
Type/Len	00 01
User Data	02
CRC	88 ee 59 7e

The DES-CFB64 encryption is performed as follows:

Mode	CFB64
Key	e6 60 0f d8 85 2e f5 ab
IV	81 0e 52 8e 1c 5f da 1a
Plaintext	00 01 02 88 ee 59 7e 00
Ciphertext	17 86 a8 03 a0 85 75 01

Notice that an octet of value 0 has been appended to the 7 plaintext octets. The value of this appended plaintext octet has no effect on the values of the first 7 ciphertext octets, which are the only ciphertext octets we are interested in. An arbitrary value can be used instead of 0 for the appended plaintext octet.

The Packet PDU, after encryption, looks like this:

DA	01 02 03 04 05 06
SA	f1 f2 f3 f4 f5 f6
Type/Len	17 86
User Data	a8
CRC	03 a0 85 75

B.7.4 40-bit key

The BPKM protocol always generates and distributes 56-bit DES keys. When 40-bit encryption is required, the 56-bit DES key is converted within an implementation to a 40-bit key by masking off (to zero) 16 of the 56 bits of a TEK.

A TEK has 8 octets, each octet containing 7 bits of key and 1 parity bit. Here is the procedure for converting a TEK to a 40-bit key:

- the first two octets of the TEK are set to 0;
- the two most significant bits of the third octet of the TEK are set to 0;
- the remaining five octets of the TEK are unchanged.

For example, if the TEK distributed by the BPKM protocol is:

```
ff ff ff ff ff ff ff ff,
```

then the conversion to 40 bits yields the TEK

```
00 00 3f ff ff ff ff ff.
```

Except for this conversion of the TEK value, the procedure for 40-bit encryption of a Packet PDU is identical to the case of 40-bit encryption.

To illustrate 40-bit encryption, a previous example of Packet PDU is repeated here, with the TEK converted to 40 bits.

Suppose that the Packet PDU, prior to encryption, is as follows:

DA	01 02 03 04 05 06
SA	f1 f2 f3 f4 f5 f6
Type/Len	00 01
User Data	02 03 04 05 06 07 08 09 0a 0b 0c 0d 0e
CRC	91 d2 d1 9f

The total number of octets to be encrypted is 19. The first 16 octets are processed using DES-CBC encryption, and the last 3 octets using DES-CFB64 encryption.

The DES-CBC encryption is performed as follows:

Mode	CBC
Key	00 00 0f d8 85 2e f5 ab
IV	81 0e 52 8e 1c 5f da 1a
Plaintext	00 01 02 03 04 05 06 07 08 09 0a 0b 0c 0d 0e 91
Ciphertext	44 c8 4a 41 14 67 56 a2 dc 64 8f b0 dc 1e 1e 86

The key is the TEK conveyed in the Key Reply message, converted to a 40-bit key. The IV is as conveyed in the Key Reply message.

The DES-CFB64 encryption is performed as follows:

Mode	CFB64
Key	00 00 0f d8 85 2e f5 ab
IV	dc 64 8f b0 dc 1e 1e 86
Plaintext	d2 d1 9f 00 00 00 00 00
Ciphertext	f1 42 aa a3 e4 9b eb 29

The key is the same as used for the DES-CBC encryption operation. The IV is the last 8 octets of ciphertext generated by the DES-CBC operation.

The Packet PDU, after encryption, looks like this:

DA	01 02 03 04 05 06
SA	f1 f2 f3 f4 f5 f6
Type/Len	44 c8
User Data	4a 41 14 67 56 a2 dc 64 8f b0 dc 1e 1e
CRC	86 f1 42 aa

B.8 Encryption of Packet PDU with Payload Header Suppression

These examples show how encryption is applied to a Packet PDU when Payload Header Suppression (PHS) is applied. The examples use an RTP Voice over IP payload. In the examples, no effort is made to use correct values for the fields of the Packet PDU. As a result, the examples here are not valid packets suitable for transmission. The intent of the examples is to illustrate encryption details only.

B.8.1 Downstream

Suppose that the Packet PDU, after PHS and prior to encryption, is as follows:

DA	01 02 03 04 05 06
SA	f1 f2 f3 f4 f5 f6
RTP header	21 22 23 24 25 26 27 28 29 2a 2b 2c
Voice data	31 32 33 34 35 36 37 38 39 3a
CRC	93 86 b3 b9

PHS has removed the Type/Len field that would otherwise be included in the Ethernet/802.3 header. The User Data consists of the RTP header and the voice data. Encryption is applied beginning with the first octet of the RTP header and ending with the last octet of the CRC, as follows:

Mode	CBC
Key	e6 60 0f d8 85 2e f5 ab
IV	81 0e 52 8e 1c 5f da 1a
Plaintext	21 22 23 24 25 26 27 28 29 2a 2b 2c 31 32 33 34 35 36 37 38 39 3a 93 86
Ciphertext	b4 55 da c8 39 1e 0c ed 15 cf b5 79 0a c3 24 5e cf 0f 52 c0 69 f5 f6 6e

Mode	CFB64
Key	e6 60 0f d8 85 2e f5 ab
IV	cf 0f 52 c0 69 f5 f6 6e
Plaintext	b3 b9 00 00 00 00 00 00
Ciphertext	3e 31 de ea 96 6a 88 6b

The Packet PDU, after encryption, looks like this:

DA	01 02 03 04 05 06
SA	f1 f2 f3 f4 f5 f6
RTP header	b4 55 da c8 39 1e 0c ed 15 cf b5 79
Voice data	0a c3 24 5e cf 0f 52 c0 69 f5
CRC	f6 6e 3e 31

B.8.2 Upstream

Suppose that the Packet PDU, after PHS and prior to encryption, is as follows:

RTP header	21 22 23 24 25 26 27 28 29 2a 2b 2c
Voice data	31 32 33 34 35 36 37 38 39 3a
CRC	65 cf fe 89

PHS has removed the DA, SA, and Type/Len fields that would otherwise be included in the Ethernet/802.3 header. The User Data consists of the RTP header and the voice data. The first 12 octets of the User Data are not encrypted. Encryption is applied beginning with the first octet of the voice data and ending with the last octet of the CRC, as follows:

Mode	CBC
Key	e6 60 0f d8 85 2e f5 ab
IV	81 0e 52 8e 1c 5f da 1a
Plaintext	31 32 33 34 35 36 37 38
Ciphertext	d6 88 87 66 1f 66 04 79

Mode	CFB64
Key	e6 60 0f d8 85 2e f5 ab
IV	d6 88 87 66 1f 66 04 79
Plaintext	39 3a 65 cf fe 89 00 00
Ciphertext	c0 07 20 8e 3b 0b b1 b9

The Packet PDU, after encryption, looks like this:

RTP header	21 22 23 24 25 26 27 28 29 2a 2b 2c
Voice data	d6 88 87 66 1f 66 04 79 c0 07
CRC	20 8e 3b 0b

B.9 Fragmented packet encryption

When a packet is fragmented, each fragment is independently encrypted using DES-CBC with residual block processing. The TEK and IV for each fragment is the same TEK and IV used for encrypting an unfragmented Packet PDU. All octets of a fragment are encrypted, including the 12 octets carrying the Ethernet/802.3 destination and source addresses (DA/SA) of the Packet PDU.

In the example here, no effort is made to use meaningful values for the fields of the packet. As a result, the example here is not a valid packet suitable for transmission. The intent of the example is to illustrate encryption details only.

In this example, the TEK and IV are taken from the example Key Reply packet described above.

Suppose that packet is divided into two fragments, as follows:

Fragment 1 payload	01 02 03 04 05 06 f1 f2 f3 f4 f5 f6 00 01 02 03 04 05
Fragment 1 CRC	b4 2b 6d d4

Fragment 2 payload	06 07 08 09 0a 0b 0c 0d
Fragment 2 CRC	48 34 45 36

The first fragment is encrypted using DES-CBC and DES-CFB64, as follows:

Mode	CBC
Key	e6 60 0f d8 85 2e f5 ab
IV	81 0e 52 8e 1c 5f da 1a
Plaintext	01 02 03 04 05 06 f1 f2 f3 f4 f5 f6 00 01 02 03
Ciphertext	47 41 0f 4f fd 78 47 6e c8 1a 67 4e 26 0c 20 c5

Mode	CFB64
Key	e6 60 0f d8 85 2e f5 ab
IV	c8 1a 67 4e 26 0c 20 c5
Plaintext	04 05 b4 2b 6d d4 00 00
Ciphertext	56 6d 5c 58 2f 56 dc 39

The first fragment, after encryption, looks like this:

Fragment 1 payload	47 41 0f 4f fd 78 47 6e c8 1a 67 4e 26 0c 20 c5 56 6d
Fragment 1 CRC	5c 58 2f 56

The second fragment is encrypted using DES-CBC and DES-CFB64, as follows:

Mode	CBC
Key	e6 60 0f d8 85 2e f5 ab
IV	81 0e 52 8e 1c 5f da 1a
Plaintext	06 07 08 09 0a 0b 0c 0d
Ciphertext	d8 55 0f 59 9d 19 d9 c6

Mode	CFB64
Key	e6 60 0f d8 85 2e f5 ab
IV	d8 55 0f 59 9d 19 d9 c6
Plaintext	48 34 45 36 00 00 00 00
Ciphertext	b4 5f 3e 95 0e e4 d7 df

The second fragment, after encryption, looks like this:

Fragment 2 payload	d8 55 0f 59 9d 19 d9 c6
Fragment 2 CRC	b4 5f 3e 95

This page intentionally left blank.

Appendix C BPI/BPI+ Interoperability

Baseline Privacy Plus is an enhancement to the original requirements of Baseline Privacy. The specification has added improvements where needed to increase system security and to address performance concerns in the original specification. The original architecture and design of Baseline Privacy has been maintained where possible.

The evolution to DOCSIS 1.1 features and Baseline Privacy Plus was not intended to immediately obsolete DOCSIS 1.0 systems and the use of Baseline Privacy. A DOCSIS system's transition to DOCSIS 1.1 compliance may be incremental. In the meantime and thereafter, DOCSIS 1.0 Baseline Privacy and DOCSIS 1.1 Baseline Privacy Plus units may coexist within a DOCSIS system.

C.1 DOCSIS v1.0/v1.1 Interoperability[22]

BPI/BPI+ Interoperability requirements are a subset of overall DOCSIS v1.0/v1.1 Interoperability requirements defined in Appendix G of [DOCSIS1] or [DOCSIS9]. Interoperability requirements defined by [DOCSIS1] or [DOCSIS9] for provisioning and registration should be followed.

C.2 DOCSIS BPI/BPI+ Interoperability Requirements

BPI/BPI+ interoperability requirements are summarized in the following table. A Baseline Privacy Plus system SHOULD be backward compatible with Baseline Privacy according to this table. There are four unit capabilities defined here from the Baseline Privacy specification and supported by these interoperability requirements.

1. Cable Modem Termination System:
    a) CMTS BPI: Baseline Privacy with 56-bit DES, and will accept both a 768 and 1024 bit public key modulus.
    b) CMTS BPI – 40bit: Baseline Privacy with 40-bit DES, and will accept both a 768 and 1024 bit public key modulus. DES can only operate in 40-bit mode.
2. Cable Modem:
    a) CM BPI: Baseline Privacy with 56-bit DES, and either a 768 or 1024 bit public key modulus.
    b) CM BPI – 40bit: Baseline Privacy with 40-bit DES, and either a 768 or 1024 bit public key modulus. DES can only operate in 40-bit mode.

As defined in this specification, Baseline Privacy Plus introduces two additional unit types.

1. CMTS BPI+: Baseline Privacy Plus with 56-bit DES, and will accept both a 768 and 1024 bit public key modulus.
2. CM BPI+: Baseline Privacy Plus with 56-bit DES, and a 1024 bit public key modulus.

22. References updated per BPI-N-02098 by RKV on 8/23/02.

The CMTS and the CM negotiate the BPI/BPI+ compatible mode using the Privacy Support Modem Capability TLV (type 5.6) in the REG-REQ and REG-RSP messages. The requirements for BPI/BPI+ interoperability are:

a) A CMTS MUST accept public keys with a modulus of both 768 and 1024 bits from a CM during authorization.

b) If a CM with Baseline Privacy Plus (CM BPI+) is provisioned with a DOCSIS 1.0 style configuration file, the CM sets the Privacy Support Modem Capability TLV (type 5.6) to either BPI Support (0) or BPI+ Support (1) depending on its capability in that situation (cf. [DOCSIS1], section G.2.1, or [DOCSIS9], Section G.1.1).[23]

c) When a CMTS with Baseline Privacy Plus (CMTS BPI+) receives the Privacy Support Modem Capability TLV set to BPI Support (type 5.6, value 0) or no type 5.6 TLV in the REG-REQ message from the CM, the CMTS MUST fall back into a Baseline Privacy compatible mode of operation [DOCSIS2] for communications with that CM.

d) When a CMTS with Baseline Privacy Plus is operating in a system that supports both BPI and BPI+ CMs, the TFTP server MUST include the following two kinds of configuration files;

   •Configuration file with all of the BPI parameters (type 17.1 through 17.7) for the CMs provisioned to operate in BPI mode, and

   • Configuration file with all of or a part of the BPI+ parameters for the CMs provisioned to operate in BPI+ mode.

e) When a CM with Baseline Privacy Plus (CM BPI+) receives the Privacy Support Modem Capability TLV set to BPI Support (type 5.6, value 0) or no type 5.6 TLV in the REG-RSP message from the CMTS, the CM MUST fall back into a Baseline Privacy mode of operation [DOCSIS2] to communicate with the CMTS.

Note that, as specified in Appendix D, the DOCSIS 1.1 CM always verifies downloaded operational software as specified in Appendix D regardless of the Privacy Support setting (type 5.6) in the REG-RSP message and the Privacy Enable setting (type 4.7 or 29) in the CM configuration file.

23. Reference updated per BPI-N-02098 by RKV on 8/23/02.

Table C-1. BPI/BPI+ Interoperability Matrix

	CM BPI	CM BPI – 40bit	CM BPI+
CMTS BPI	Domestic BPI configuration. 768 or 1024-bit RSA modulus.	768 or 1024 bit RSA modulus. CMTS software zeros TEK bits to 40-bit standard	CM falls back into BPI mode with 1024-bit RSA modulus
CMTS BPI-40bit	768 or 1024 bit RSA modulus. CMTS software zeros TEK bits to 40-bit standard	768 or 1024 bit RSA modulus. All 40-bit compatibility handled by MAC chips.	CM falls back into BPI mode with 1024-bit RSA modulus. CMTS software zeros TEK bits to 40-bit standard
CMTS BPI+	CMTS falls back into BPI mode. 768 or 1024-bit RSA modulus.	768 or 1024 bit RSA modulus. CMTS software zeros TEK bits to 40-bit standard	Full BPI+ mode or BPI mode depending on configuration file and CMTS setting. 1024-bit RSA modulus.

C.3 BPI 40-bit DES Export Mode Considerations

The Baseline Privacy Plus specification is backward compatible with the 40-bit DES export mode of Baseline Privacy. The burden of compliance is placed on the CMTS. Not all DOCSIS equipment vendors will ever have the need to operate in a system with 40-bit DES capable BPI units. Therefore, compliance is up to the individual CMTS manufacturer. A CMTS SHOULD support backward compatibility to 40-bit DES Baseline Privacy. If it does, it MUST do so according to this specification document.

a) When a CMTS is sending or receiving encrypted data between itself and a CM that uses 40-bit DES, the CMTS MUST zero the appropriate bits of its TEKs before encrypting or decrypting corresponding traffic data. The appropriate bits of the TEK MUST be zeroed according to the 40-bit TEK requirement of Baseline Privacy.

b) When encrypted traffic is to be passed between a CMTS with only 40-bit DES capability and a CM with a 56-bit DES capability, the CMTS MUST provide a 40-bit compliant TEK in the Key Reply Message to the CM.

The method a CMTS uses to recognize which CMs in a system are capable of 56-bit DES or only 40-bit DES, is left up to the individual system operator and CMTS vendor to accomplish in the manner that best fits their situation. One method for obtaining this information would be from the CM vendors, based on CM serial numbers, MAC address, manufacture dates, or some other device tracking mechanism. Once collected, the information would be incorporated into the CMTS database of information stored on each CM.

An alternative method for obtaining this information is with a DOCSIS BPI MIB defined for this purpose.

C.4 System Operation

C.4.1 CMTS with BPI Capability

A CMTS with BPI capability will always provision CMs using DOCSIS 1.0 style TFTP configuration files and BPI configuration settings. Both the BPI and BPI+ CMs will receive the BPI settings and each CM will only attempt to register as a DOCSIS 1.0 CM with BPI capability. If a CM returns a Modem Capability of BPI+ in the registration request, the CMTS will respond with this capability removed and force the CM to BPI compatibility.

C.4.2 CMTS with BPI+ Capability

A CMTS with DOCSIS 1.1 BPI+ capability MUST be capable of operating in both BPI and BPI+ compatible modes and to adjust according to the capability of each client CM. When the CMTS has BPI+ capability and the system simultaneously supports BPI and BPI+ CMs, both DOCSIS 1.0 and DOCSIS 1.1 configuration files MUST be available to deliver the BPI+ and BPI configuration settings to the appropriate CMs. A BPI capable CM will receive a DOCSIS 1.0 configuration file with BPI settings. It will then register with BPI Modem Capability.

Appendix D Verifying Downloaded Operational Software

D.1 Introduction

The DOCSIS system supports the remote downloading of code to its network cable modems. The source and integrity of the downloaded code is important to the overall operation and security of the DOCSIS system.

The software download module is an attractive target for an attacker. If an attacker were able to mount a scalable attack against the software download module, he could potentially install code to disable all the CMs within a domain, or disrupt service on a wide scale. To thwart these attacks, the attacker must be forced to overcome several security barriers.

D.2 Overview

The requirements defined in this section address these primary security goals for the code download process:

- The CM should have a means to authenticate that the originator of any download code is a known and trusted source.
- The CM should have a means to verify that the downloaded code has not been altered from the original form in which it was provided by the trusted source.
- The process should strive to simplify the MSO's code file handling requirements and provide mechanisms for the MSO to upgrade or downgrade the code version of cable modems on their network.
- The process must also allow the option for an MSO to dictate and control their own policies first-hand, with respect to (a) which code files will be accepted by cable modems within their network domain, and (b) security controls defining the security of the process on their network.
- Cable modems must be able to move freely between systems controlled by different MSO organizations.
- Root CA Public Key (optional): an updated Root CA Public Key that replaces the Root CA Public Key currently stored in the CM.
- Manufacturer Certificate(s) (optional): One or more X.509 compliant Manufacturer Certificate(s) that replaces the Manufacturer Certificates currently stored in the CM.

This document limits its scope to these primary system security requirements, but acknowledges that in some cases additional security may be desired. The concerns of individual MSOs or cable modem manufacturers may result in additional security related to the distribution and installation of code into a cable modem or other DOCSIS network element. This specification does not restrict the use of further protections, as long as they do not conflict with the intent and guidelines of this specification.

There are multiple levels of protection required to successfully protect and verify the code download.

- The manufacturer of the CM code always applies a digital signature to the code file; a signature that is verified with a certificate chain that extends up to the DOCSIS root. The manufacturer's signature authenticates the source and integrity of the code file to the CM. Additional control parameters are included in the code file to control access to the CM.

- Though the manufacturer must always sign their code file, an MSO may later apply their code signature in addition to the manufacturer's signature. The CM must verify both signatures with a certificate chain that extends up to the DOCSIS root before accepting a code file.
- OSS mechanisms for the provisioning and control of the CM are important to the proper execution of this process. The code upgrade capability of a CM is enabled during the provisioning and registration process. Code downloads are initiated during the provisioning and registration process; or can be initiated in normal operation using an SNMP command.

The code file is built using a PKCS#7 compliant structure that has been defined in a specific format for use with DOCSIS cable modems. Included in the DOCSIS PKCS#7 structure is the:

- code image: the upgrade code image
- Code Verification Signature (CVS): the digital signature over the code image and any other authenticated attributes as defined in the DOCSIS PKCS#7 structure
- Code Verification Certificate (CVC): an X.509 compliant certificate structure that is used to deliver and validate the public code verification key that will verify the signature over the code image. The DOCSIS Certificate Authority, a trusted party, whose public key is already stored in the cable modem, signs the certificate. The X.509 certificate is defined in a specific format for use with DOCSIS cable modems.

Figure D-1 shows the basic steps required for the signing of a code image when the code file is signed only by the CM manufacturer; and when the code file is signed by the CM manufacturer and co-signed by an MSO.

In the DOCSIS system each cable modem will receive a trusted public key from the DOCSIS Root Certificate Authority. The code manufacturer will build the code file by signing the code image using a DOCSIS PKCS#7 digital signature structure with a DOCSIS X.509 certificate. The code file is then sent to the MSO. The MSO, in possession of a DOCSIS root public key, SHOULD verify that the code file is from a trusted DOCSIS manufacturer and has not been modified. At this point, the MSO has the option of loading the code file on the tftp server as-is, or adding their signature and their MSO CVC to the code file. During the code upgrade process, the CM will access the code file from the tftp server and verify the code image before installing.

While the DOCSIS Root CA for the cable modem certificate chain currently serves as the Root CA for the Secure Software Download, the different Root CA may be used in the future. Therefore, the CM MUST NOT assume that the Manufacturer CVC and Co-signer CVC are issued by the DOCSIS Root CA for the cable modem certificate chain.[24]

24. added per bpi-n-02008, 07/25/02, ab

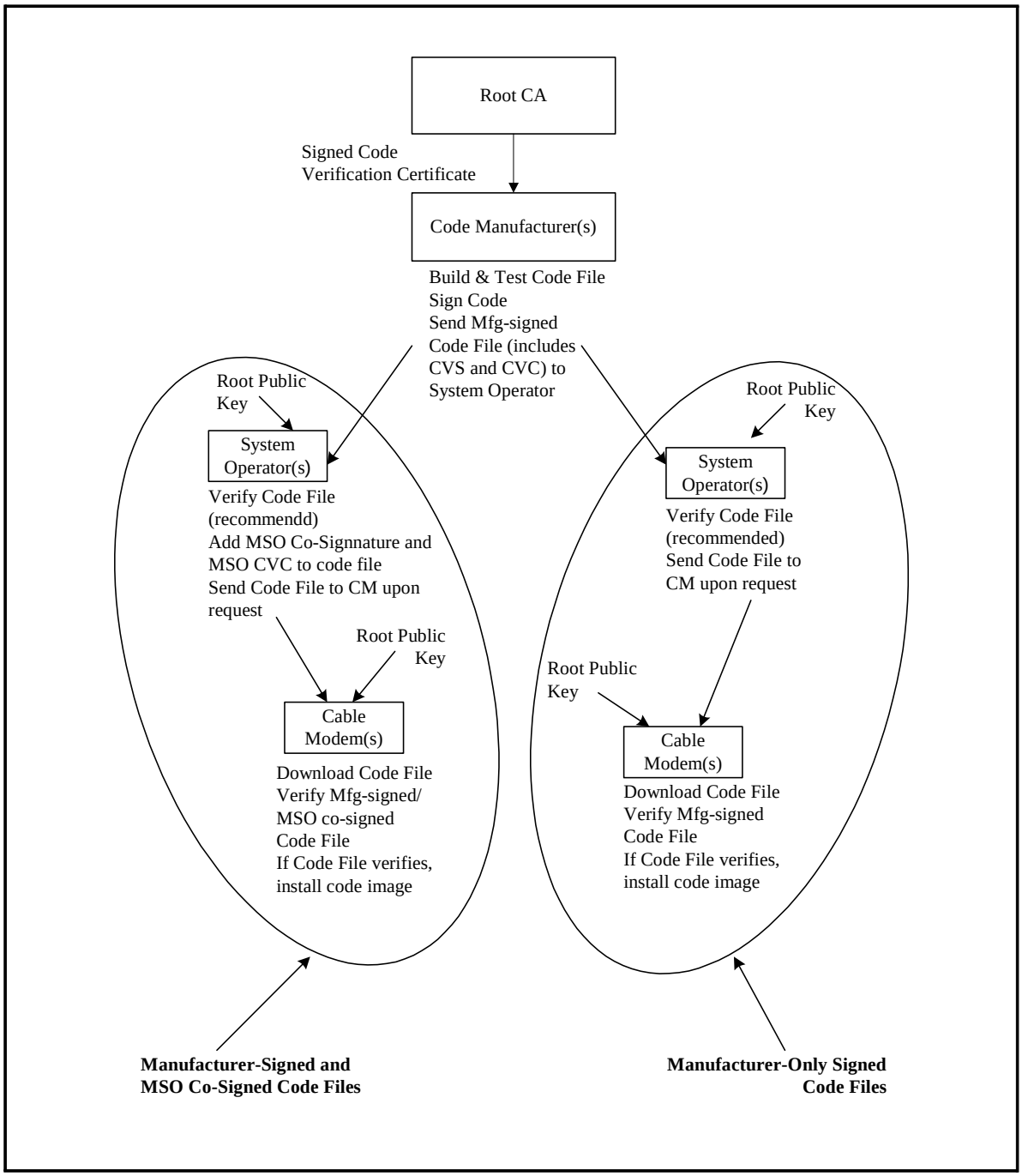


Figure D-1. Typical Code Validation Hierarchy

D.3 Code Upgrade Requirements[25]

The following sections define requirements in support of the code upgrade verification process. All DOCSIS 1.1 or 2.0 code upgrades MUST be prepared and verified as defined in this specification. All DOCSIS 1.1 or 2.0 certified cable modems MUST verify code upgrades according to this specification, regardless of whether it is operating in a DOCSIS 2.0, DOCSIS 1.1, or DOCSIS 1.0 compliant mode. All DOCSIS 1.1 and DOCSIS 2.0 certified cable modems MUST verify code upgrades according to this specification regardless of whether Baseline Privacy is enabled or disabled.

D.3.1 Code File Requirements

A single file is used to encapsulate the code for the cable modem. The code file is a DOCSIS PKCS#7 signed data message that includes:

1. the Manufacturer's Code Verification Signature (CVS)
2. the Manufacturer's Code Verification Certificate (CVC) signed by the DOCSIS root CA
3. the code image (compatible with the destination cable modem) as signed content
4. optionally, when the MSO co-signs the code file:
    a) the MSOs CVS
    b) the MSOs CVC signed by the DOCSIS root CA
5. optional Root CA Public Key for the CVC verification
6. optional Manufacturer Certificate(s)

The code file MUST comply with [PKCS#7] and MUST be DER encoded. The code file MUST match the structure shown in Table D-1. An example is shown in Appendix B.

25. Section D.3 updated per BPI-N-02098 by RKV on 8/26/02.

Table D-1. Code File Structure

Code File	Description
PKCS#7 Digital Signature{	
ContentInfo	
contentType	SignedData
SignedData()	EXPLICIT signed-data content value; includes CVS and X.509 CVC
}	
SignedContent{	
DownloadParameters {	Mandatory TLV format (Type 28) defined in the section 4.2.2.28. (Length is zero if there is no sub-TLVs.)
RootCAPublicKey()	Optional TLV for the Root CA Public Key for CVC Verification, formatted according to the RSA-Public-Key TLV format (Type 4) defined in the section 4.2.2.4.
MfgCerts()	Optional TLV for one or more DER-encoded Manufacturer Certificate(s) each formatted according to the CA-Certificate TLV format (Type 17) defined in the section 4.2.2.17.
}	
CodeImage()	Upgrade code image
}	

If when downloading the Root CA Public Key and/or the Manufacturer Certificate as a part of the CM Code File, the Root CA Public Key and/or the Manufacturer CA Certificates MAY be contained in the RootCAPublicKey field and/or the MfgCerts field as specified in the Table D-1 respectively, and separated from the actual cable modem code image contained in the CodeImage field.

This makes it possible to clearly discriminate the code image from other parameters in the code download file. This makes it possible to change the Root CA Public Key, the Manufacturer CA Certificates or SignedData parameters in the code download file without disrupting or changing the code image that the cable modem will receive. This allows one to verify that the code image has not changed even though the code download file changed because of a change in the Root CA Public Key, the Manufacturer CA Certificates or SignedData parameters.

D.3.1.1 DOCSIS PKCS#7 Signed Data

The software upgrade file will contain the information in a PKCS#7 Signed Data content type as shown below. Though maintaining compliance to [PKCS#7], the structure used by DOCSIS has been restricted in format to ease the processing a CM does to validate the signature. The PKCS#7 Signed Data MUST be DER encoded and exactly match the structure shown in Table D-2 except for any change in order required to DER encode (*e.g.*, the ordering of SET OF attributes). The CM SHOULD reject the PKCS#7 signature if the PKCS#7 Signed Data does not match the DER encoded structure represented in Table D-2.[26]

D.3.1.1.1 Code Signing Keys

The PKCS#7 digital signature uses the RSA Encryption Algorithm [RSA3] with SHA-1 [FIPS186]. The RSA key modulus for code signing is 1024-bits, 1536-bits, or 2048-bits in length. The CM MUST be able to verify DOCSIS code file signatures that are signed using either modulus size. The public exponent is F4 (65537 decimal).

26. edited 07/24/02 per bpi-n-02011, ab

Table D-2. DOCSIS PKCS#7 Signed Data

PKCS#7 Field	Description
Signed Data {	
version	version = 1
digestAlgorithmIdentifiers	SHA-1
contentInfo	
contentType	data (SignedContent is concatenated at the end of the PKCS#7 structure)
certificates {	DOCSIS Code Verification Certification (CVC)
mfgCVC	REQUIRED for all code files
msoCVC	OPTIONAL; required for MSO co-signatures
} *end certificates*	
SignerInfo{	
MfgSignerInfo {	REQUIRED for all code files
version	version = 1
issuerAndSerialNumber	from the signer's certificate
issuerName	distinguished name of the certificate issuer
CountryName	US
organizationName	Data Over Cable Service Interface Specifications
organizationalUnitName	Cable Modems
commonName	DOCSIS Cable Modem Root Certificate Authority
certificateSerialNumber	from CVC; Integer, size (1..20) octets
digestAlgorithm	SHA-1
authenticatedAttributes	
contentType	data; contentType of signedContent
signingTime	UTCTime (GMT), YYMMDDhhmmssZ
messageDigest	digest of the content as defined in [PKCS#7]
digestEncryptionAlgorithm	rsaEncryption
encryptedDigest	
} *end mfg signer info*	
MsoSignerInfo {	OPTIONAL; required for MSO co-signatures
version	version =1
issuerAndSerialNumber	from the signer's certificate
issuerName	distinguished name of the certificate issuer
CountryName	US
organizationName	Data Over Cable Service Interface Specifications
organizationalUnitName	Cable Modems
commonName	DOCSIS Cable Modem Root Certificate Authority
certificateSerialNumber	from CVC; Integer, size (1..20) octets
digestAlgorithm	SHA-1
authenticatedAttributes	
contentType	data; contentType of signedContent
signingTime	UTCTime (GMT), YYMMDDhhmmssZ
messageDigest	digest of the content as defined in [PKCS#7]
digestEncryptionAlgorithm	rsaEncryption
encryptedDigest	
} *end mso signer info*	
} *end signer info*	
} *end signed data*	

D.3.1.1.2 Code Verification Certificate Format

The format used for the CVC is X.509 compliant. However, in this case, the X.509 structure has been restricted to ease the processing a CM does to validate the certificate and extract the public key used to verify the CVS. The CVC MUST be DER encoded and exactly match the structure shown in Table D-3 except for any change in order required to DER encode (*e.g.*, the ordering of SET OF attributes). The CM SHOULD reject the CVC if it does not match the DER encoded structure represented in Table D-3.

The CVC also requires the addition of the Key Purpose ID for “code-signing” within an Extended Key Usage field.

```
-- extended key usage extension OID and syntax
id-ce-exKeyUsage OBJECT IDENTIFIER ::= {id-ce 37}
ExtKeyUsageSyntax ::= SEQUENCE SIZE (1..MAX) OF KeyPurposeId
KeyPuposeID ::= OBJECT IDENTIFIER
```

The DOCSIS CVC MUST contain one, and only one, extension field: the extended key usage extension. The extended key usage extension MUST be flagged as critical. The key usage extension MUST contain the code purpose OID for code signing. If the extended key usage extension is not present, or is not flagged critical, or includes any key purpose OID other than, or in addition to, the code-signing purpose ID, the CM MUST halt the validation process and discard the CVC.

```
-- extended key purpose OIDs
id-kp-codeSigning OBJECT IDENTIFIER ::= { id-kp 3 }
```

Table D-3. DOCSIS X.509 Compliant Code Verification Certificate

X.509 Certificate Field	Description
Certificate {	
tbsCertificate	
version	v3(2)
serialNumber	Integer, size (1..20) octets
signature	SHA-1 with RSA, null parameters
issuer	
countryName	US
organizationName	Data Over Cable Service Interface Specifications
organizationalUnitName	Cable Modems
commonName	DOCSIS Cable Modem Root Certificate Authority
validity	
notBefore	utcTime (GMT), YYMMDDhhmmssZ
notAfter	utcTime (GMT), YYMMDDhhmmssZ
subject	
countryName	<country of subject company>
organizationName	<subject code-signing agent>
organizationalUnitName	DOCSIS
commonName	Code Verification Certificate
subjectPublicKeyInfo	
algorithm	RSA encryption, null parameters
subjectPublicKey	1024-bit, 1536-bit, or 2048-bit modulus
extensions	
extKeyUsage	
critical	true
keypurposeId	id-kp-codeSigning
signatureAlgorithm	SHA-1 with RSA, null parameters
signature Value	
} *end certificate*	

D.3.1.1.3 Certificate Revocation

This specification does not require or define the use of certificate revocation lists (CRLSs). The cable modem is not required to support CRLs. MSOs may want to define and use CRLs outside of the DOCSIS HFC network to help manage code files provided to them by manufacturers.

However, there is a method for revoking certificates based on the validity start date of the certificate (described in section D 3.2.2). This method requires that an updated CVC be delivered to the cable

modem with an updated validity start time. Once the CVC is successfully validated, the X.509 validity start time will update the CM's current value of cvcAccessStart.

To expedite the delivery of an updated CVC without requiring the cable modem to process a code upgrade, the CVC MAY be delivered in either the CM's configuration file or an SNMP MIB. The format of a DOCSIS CVC is the same whether it is in a code file, configuration file, or SNMP MIB.

D.3.1.2 Signed Content

The signed content field of the code file contains the code image and the download parameters field, which possibly contains two additional optional items - a DOCSIS Root CA Public key and Manufacturer Certificate.

The final code image is in a format compatible with the destination cable modem. In support of the PKCS#7 signature requirements, the code content is typed as data; *i.e.*, a simple octet string. The format of the final code image is not specified here and will be defined by each manufacturer according to their requirements.

Each manufacturer SHOULD build their code with additional mechanisms that verify an upgrade code image is compatible with the destination cable modem. The CM SHOULD NOT install the upgraded code image unless the code image has been verified as being compatible with the CM.

If included in the signed content field, the DOCSIS Root CA Public key is intended to replace the DOCSIS Root CA Public key currently stored in the CM. If the code download and installation specified in section D.3.5.1 is successful, then the CM MUST replace its currently stored DOCSIS Root CA Public key with the DOCSIS Root CA Public key received in the signed content field. This new DOCSIS Root CA Public key will then be used for subsequent CVC verification.

If included in the signed content field, the Manufacturer Certificate(s) is intended to replace the Manufacturer Certificate(s) currently stored in the CM. If the code download and installation specified in section D.3.5.1 is successful, then the CM MUST replace its currently stored Manufacturer Certificate(s) with the Manufacturer Certificate(s) received in the signed content field. The new Manufacturer Certificate(s) will then be sent to the CMTS during the subsequent BPI+ initialization.

D.3.2 Code File Access Controls

In addition to the cryptographic controls provided by the digital signature and the X.509 certificate, special control values are included in the code file for the cable modem to check before it will validate a code image. The conditions placed on the values of these control parameters MUST be satisfied before the CM will validate the CVC or the CVS, and accepts the code image.

D.3.2.1 Subject Organization Names

The cable modem will recognize up to two names, at any one time, that it considers a trusted code-signing agent in the subject field of a code file CVC. These include,

- The cable modem manufacturer: The manufacturer name in the manufacturer's CVC subject field MUST exactly match the manufacturer name stored in the CM's non-volatile memory by the manufacturer. A manufacturer CVC MUST always be included in the code file.
- A co-signing agent: DOCSIS and the manufacturer allows another trusted organization to co-sign code files destined for their cable modems. In most cases this is the MSO controlling the current operating domain of the cable modem. The organization name of the co-signing agent is communicated to the cable modem via a co-signer's CVC in the configuration file when initializing the cable modem's code verification process. The co-signer's organization name in the co-signer's CVC subject field MUST exactly match the co-signer's organization name previously received in the co-signer's initialization CVC and stored by the CM.

The CM MAY compare organization names using a binary comparison.

D.3.2.2 Time Varying Controls

In support of the code upgrade process, The CM MUST keep two UTC time values associated with each code-signing agent. One set MUST always be stored and maintained for the cable modem's manufacturer. While the cable modem is assigned a code co-signing agent, the cable modem MUST also store and maintain a separate set of time values for the co-signing agent.

These values are used to control code file access to the cable modem by individually controlling the validity of the CVS and the CVC. These values are:

codeAccessStart: a 12-byte UTC time value referenced to Greenwich Mean Time (GMT).

cvcAccessStart: a 12-byte UTC time value referenced to GMT.

UTCTime values in the CVC MUST be expressed as Greenwich Mean Time (GMT) and MUST include seconds. That is, they MUST be expressed in the following form: YYMMDDhhmmssZ. The year field (YY) MUST be interpreted as follows:

- where YY is greater than or equal to 50, the year shall be interpreted as 19YY
- where YY is less than 50, the year shall be interpreted as 20YY

These values will always be referenced to Greenwich Mean Time (GMT), so the final ASCII character (Z) can be removed when stored by the CM as codeAccessStart and cvcAccessStart. The CM MUST maintain each of these time values in a format that contains equivalent time information and accuracy to the 12 character UTV format (i.e., YYMMDDhhmmss). The CM MUST accurately compare these stored values with UTC time values delivered to the CM in a CVC. These requirements are discussed later in this specification.

The values of codeAccessStart and cvcAccessStart corresponding to the cable modem's manufacturer MUST NOT decrease. The value of codeAccessStart and cvcAccessStart corresponding to the co-signing agent MUST NOT decrease as long as the co-signing agent does not change and the CM maintains that co-signer's time-varying control values.

D.3.3 Cable Modem Code Upgrade Initialization

Before the cable modem can upgrade code, it should be properly initialized. Its manufacturer first initializes the cable modem. Every time a cable modem registers on a DOCSIS network, it MUST check its current initialization state with respect to the operational needs of the particular network. It may be necessary for the cable modem to reinitialize at registration; particularly if the cable modem has moved from one network to another.

D.3.3.1 Manufacturer Initialization

It is the responsibility of the manufacturer to correctly install the initial code version in the CM.

In support of code upgrade verification, values for these parameters MUST be loaded into the CM's non-volatile memory:

1. CM manufacturer's organizationName
2. Manufacturer's time-varying control values:
    a) codeAccessStart initialization value
    b) cvcAccessStart initialization value

The organization name of the cable modem manufacturer MUST always be present in the cable modem. The cable modem manufacturer's organizationName MAY be stored in the cable modems code image. Under normal conditions the manufacturer's organizationName SHOULD NOT change, but this specification does not prohibit a manufacturer from changing how its organizationName is stored in the CM. The manufacturer named used for code upgrade is not necessarily the same name used in the DOCSIS Manufacturer Certificate.

The time-varying control values, codeAccessStart and cvcAccessStart, MUST be initialized to a UTCTime compatible with the validity start time of the manufacturer's latest CVC. These time-varying values will be updated periodically under normal operation via manufacturer's CVC's that are received and verified by the cable modem.

Originally, the cable modem will not recognize a co-signing agent.

D.3.3.2 Network Initialization

The method for initiating and obtaining CM code download files is defined in [DOCSIS1]. In support of code verification, the configuration file is used as an authenticated means in which to initialize the code verification process. In the cable modem configuration file, the cable modem receives configuration settings relevant to code upgrade verification. These settings MUST NOT be used until after CMTS has successfully registered the CM.

The configuration file SHOULD always include the most up-to-date CVC applicable for the destination cable modem; but when the configuration file is used to initiate a code upgrade, it MUST include a Code Verification Certificate (CVC) to initialize the cable modem for accepting code files

according to this specification. Regardless of whether a code upgrade is required, a CVC in the configuration file MUST be processed by the cable modem.

A configuration file MAY contain:

- no CVC
- a Manufacturer's CVC only
- a Co-Signer's (MSO) CVC only
- both a Manufacturer's CVC and a Co-Signer's CVC

Before the CM will enable its ability to upgrade code files on the network, it MUST receive a valid CVC in a configuration file and successfully register with the CMTS. In addition, when the cable modem's configuration file does not contain a valid CVC, and its ability to upgrade code files has been disabled, the CM MUST reject any information in a CVC subsequently delivered via SNMP.

When the cable modem's configuration file only contains a valid Manufacturer's CVC, the cable modem will only require a manufacturer signature on the code files. In this case, the CM MUST NOT accept code files that have been co-signed.

When the cable modem's configuration file contains a co-signer's CVC, it is used to initialize the cable modem with a co-signing agent. Once validated, the name of the CVC's subject organizationName will become the code co-signing agent assigned to the cable modem. In order for a CM to subsequently accept a code image, the co-signer in addition to the cable modem manufacturer MUST have signed the code file.

The organization name of the cable modem manufacturer and the manufacturer's time-varying control values MUST always be present in the cable modem. If the cable modem is initialized to accept code co-signed by an additional code-signing agent, the name of the organization and their corresponding time-varying control values MUST be stored and maintained while operational. Space MUST be allocated in the cable modem's memory for the following co-signer's control values:

1. co-signing agent's organizationName
2. co-signer's time-varying control values:
    a) cvcAccessStart
    b) codeAccessStart

The manufacturer's set of these values MUST be stored in the CM's non-volatile memory and not lost when the CM's main power source is removed or during a CM reboot process. When a co-signer is assigned to the CM, the co-signer's set of these values MUST be stored in the CM's memory. The CM MAY retain these values in non-volatile memory that will not be lost when the CM's main power source is removed or during a CM reboot process. However, when assigning a CM a co-signing agent, the CVC is always in the configuration file. Therefore, the CM will always receive the co-signer's control values during the initialization phase and is not required to store the co-signer's time-varying control values when main power is lost or during a reboot process.

D.3.3.2.1 Processing the Configuration File CVC

When a CVC is included in the DOCSIS 1.1 or 2.0 configuration file, the CM MUST verify the CVC before accepting any of the code upgrade settings it contains. At receipt of the CVC in the configuration file, the CM MUST perform the following validation and procedural steps.If any of the following verification checks fail, the CM MUST immediately halt the CVC verification process and log the error if applicable.[27] If the CM configuration file does not include a CVC that validates properly, the CM MUST NOT download upgrade code files whether triggered by the CM configuration file or via an SNMP MIB. In addition, if the CM configuration files does not include a CVC that validates properly, the CM is not required to process CVC's subsequently delivered via an SNMP MIB, and MUST NOT accept information from a CVC subsequently delivered via an SNMP MIB.

At receipt of the CVC in a configuration file, and after the CM has successfully registered with the CMTS, the CM MUST:

1. Verify that the extended key usage extension is in the CVC as defined in section D.3.1.1.2.
2. Check the CVC subject organization name.

   If the CVC is a Manufacturer's CVC (Type 32) then:

   a) IF, the organizationName is identical to the cable modem's manufacturer name, THEN this is the manufacturer's CVC. In this case, the CM MUST verify that the manufacturer's CVC validity start time is greater-than or equal-to the manfacturer's cvcAccessStart value currently held in the CM.

   b) IF, the organizationName is not identical to the cable modem's manufacturer name, THEN this CVC MUST be rejected and the error logged.

   If the CVC is a Co-signer's CVC (Type 33) then:

   a) IF, the organizationName is identical to the cable modem's current code co-signing agent, THEN this is the current co-signer's CVC and the CM MUST verify that the validity start time is greater-than or equal-to the co-signer's cvcAccessStart value currently held in the CM.

   b) IF, the organizationName is not identical to the current code co-signing agent name, THEN after the CVC has been validated (and registration is complete) this subject organization name will become the CM's new code co-signing agent. The CM MUST NOT accept a code file unless it has been signed by the manufacturer, and co-signed by this code co-signing agent.
3. Validate the certificate signature using the DOCSIS root key held by the CM. Verification of the CVC signature will authenticate the source and validate trust in the CVC parameters.
4. Update the CM's current value of cvcAccessStart corresponding to the CVC's subject organizationName (*i.e.*, manufacturer or code co-signing agent) with the validity start time value from the validated CVC. If the validity start time value is greater than the CM's current value of codeAccessStart, update the CM's codeAccessStart value with the validity start time value. The CM SHOULD discard any remnants of the CVC.

27. edited per bpi-n-02011, 07/27/02, ab

D.3.3.2.2 Processing the SNMP CVC

The CM MUST process SNMP delivered CVC's when enabled to upgrade code files; otherwise, all CVC's delivered via SNMP MUST be rejected. When validating the CVC delivered via SNMP, the CM MUST perform the following validation and procedural steps. If any of the following verification checks fail, the CM MUST immediately halt the CVC verification process, log the error if applicable, and remove all remnants of the process to that step.[28]

The CM MUST:

1. Verify that the extended key usage extension is in the CVC as defined as defined is section D.3.1.1.2.
2. Check the CVC subject organization name.
    a) IF, the organizationName is identical to the cable modem's manufacturer name, THEN this is the manufacturer's CVC. In this case, the CM MUST verify that the manufacturer's CVC validity start time is greater-than the manufacturer's cvcAccessStart value currently held in the CM.
    b) IF, the organizationName is identical to the cable modem's current code co-signing agent, THEN this is a current co-signer's CVC and the validity start time MUST be greater-than the co-signer's cvcAccessStart value currently held in the CM.
    c) IF, the organizationName is not identical to cable modem's manufacturer or current code co-signing agent name, THEN the CM MUST immediately reject this CVC.
3. Validate the certificate signature using the DOCSIS root key held by the CM. Verification of the signature will authenticate the certificate and confirm trust in the CVC's validity start time.
4. Update the current value of the subject's cvcAccessStart values with the validated CVC's validity start time value. If the validity start time value is greater than the CM's current value of codeAccessStart, update the CM's codeAccessStart value with the validity start value. All certificate parameters EXCEPT for the validity start time are no longer needed and SHOULD be discarded.

D.3.4 Code Signing Requirements

The following procedures MUST be followed when signing code files.

D.3.4.1 DOCSIS Certificate Authority (CA) Requirements

In addition to the DOCSIS Manufacturer Certificate issued to a manufacturer as described earlier in this document, the DOCSIS Root CA will issue code-signing certificates called Code Verification Certificates (CVCs).

The Code Verification Certificate (CVC) is provided by the DOCSIS CA and signed by the DOCSIS root key (DRK). The CVCs signed by the DOCSIS CA MUST be exactly as specified in D.3.1.1.2 and only used in support of DOCSIS cable modem code signatures. The DOCSIS CA MUST not sign any

28. edited 07/25/02 per bpi-n-02011, ab

CVC unless it is identical to the format specified in that section. Before signing a CVC, the DOCSIS CA MUST verify that the code-signing agent is authentic and is valid code-signing agent.

The DOCSIS CA will be responsible for registering names of authorized code-signing agents. Code-signing agents include the CM manufacturers and MSO's that will co-sign cable modem code images. It is the responsibility of the DOCSIS CA to guarantee that the organization name of every code-signing agent is different. The following guidelines MUST be enforced when assigning organization names for code co-signers:

- The organization name used to identify itself as a code co-signer agent in a CVC MUST be assigned by DOCSIS.
- The name MUST be a printable strong of eight hexadecimal digits that uniquely distinguishes a code-signing agent from all others.
- Each hexadecimal digit in the name MUST be chosen from the character set 0-9 (0x30-0x39) or A-F (0x41-0x46).
- The string consisting of eight 0-digits is not allowed and MUST NOT be used in a CVC.

D.3.4.2 Manufacturing Requirements

To sign their code files, the manufacturer MUST obtain a valid CVC from the DOCSIS CA. All manufacturer code images provided to an MSO for remote upgrade of a CM on a DOCSIS HFC network, MUST be signed according to the requirements defined in this specification.

When signing a code file, a manufacturer MAY choose not to update the PKCS#7 signingTime value in the manufacturer's signing information. This specification requires that the PKCS#7 signingTime value be equal-to or greater-than the CVC's validity start time. If the manufacturer uses a signingTime equal to the CVC's validity start time when signing a series of code files, those code files can be used and re-used. This allows an MSO to use the code file to either upgrade or downgrade the code version for that manufacturer's cable modems. These code files will be valid until a new CVC is generated and received by the cable modem. it is recommended that a manufacturer sign their code files in this manner when DOCSIS and the manufacturer's security policy allows it (See D.4 Security Considerations).

To conserve storage space, the CM MAY internally represent the code co-signing agent's name in an alternate format as long as all information is maintained and the original format can be reproduced; e.g., as a 32-bit nonzero integer, with an integer value of 0 representing the absence of a code-signing agent.[29]

D.3.4.3 MSO Requirements

A DOCSIS MSO will receive software upgrade code files from the manufacturer. Using the DOCSIS root public key, the MSO should validate that the code image is as built by the trusted manufacturer. The MSO can re-verify the code file at any time by repeating the process.

29. edited per bpi-n-02076, 07/30/02, ab

The MSO has the option of co-signing the code image destined for a cable modem on their network. To do this, MSO co-signs the file content according to the PKCS#7 signature standard, and includes their DOCSIS-signed CVC. DOCSIS does not require an MSO to co-sign code files; but when the MSO follows all the rules defined in this specification for preparing a code file, the cable modem MUST accept it.

All code images downloaded to a CM across the DOCSIS HFC network MUST be signed according to the requirements defined in this specification.

D.3.5 Code Verification Requirements

Upgrade code MUST NOT be installed unless the code is found to be trusted according to the verification process described in this specification.

The CM MUST be able to process a PKCS#7 digital signature and a DOCSIS X.509 certificate as defined in this specification. The CM does not have to support the full range of the PKCS#7 and X.509 specifications.

D.3.5.1 Cable Modem Code Verification Steps

When downloading code the CM MUST perform the verification checks presented in this section. If any of the verification checks fail, or if any section of the code file is rejected due to invalid formatting, the CM MUST immediately halt the download process, log the error if applicable, remove all remnants of the process to that step, and continue to operate with its existing code. The verification checks can be made in any order, as long as all of the applicable checks presented in this section are made.

1. The CM MUST validate the manufacturer's signature information by verifying that:
    a) the PKCS#7 signingTime value is equal-to or greater-than the manufacturer's codeAccessStart value currently held in the CM
    b) the PKCS#7 signingTime value is equal-to or greater-than the manufacturer's CVC validity start time
    c) the PKCS#7 signingTime value is less-than or equal-to the manufacturer's CVC validity end time
2. The CM MUST validate the manufacturer's CVC by verifying that:
    a) the CVC subject organizationName is identical to the manufacturer name currently stored in the CM's memory
    b) the CVC validity start time is equal-to or greater-than the manufacturer's cvcAccessStart value currently held in the CM
    c) the extended key usage extension is in the CVC as defined in section D.3.1.1.2

3. The CM MUST validate the certificate signature using the DOCSIS root key held by the CM. Verification of the signature will authenticate the source of the public code verification key (CVK) and confirm trust in the key. Once trust has been established in the manufacturer's CVK, the remaining certificate parameters EXCEPT for the validity start time are no longer needed and SHOULD be discarded.
4. The CM MUST verify the manufacturer's code file signature.
    a) The CM MUST perform a new SHA-1 hash over the SignedContent. If the value of the messageDigest doesn't match the new hash, the CM MUST consider the signature on the code file as invalid.
    b) If the signature does not verify, all components of the code file (including the code image), and any values derived from the verification process MUST be rejected and SHOULD be immediately discarded.
5. If the manufacturer signature verifies and a co-signing agent signature is required:
    a) The CM MUST validate the co-signer's signature information by verifying that:
        i) the co-signer's signature information is included in the code file
        ii) the PKCS#7 signingTime value is equal-to or greater-than the corresponding code-AccessStart value currently held in the CM
        iii) the PKCS#7 signingTime value is equal-to or greater-than the corresponding CVC validity start time
        iv) the PKCS#7 signingTime value is less-than or equal-to the corresponding CVC validity end time
    b) The CM MUST validate the co-signer's CVC, by verifying that:
        i) the CVC subject organizationName is identical to the co-signer's organization name currently stored in the CM's memory
        ii) the CVC validity start time is equal-to or greater-than the cvcAccessStart value currently held in the CM for the corresponding subject organizationName
        iii) the extended key usage extension is in the CVC as defined in section D.3.1.1.2
    c) The CM MUST validate the certificate signature using the DOCSIS root key held by the CM. Verification of the signature will authenticate the source of the co-signer's public code verification key (CVK) and confirm trust in the key. Once trust has been established in the co-signer's CVK, the remaining certificate parameters EXCEPT for the validity start time are no longer needed and SHOULD be discarded.
    d) The CM MUST verify the co-signer's code file signature.
    e) The CM MUST perform a new SHA-1 hash over the SignedContent. If the value of the messageDigest doesn't match the new hash, the CM MUST consider the signature on the code file as invalid.
    f) If the signature does not verify, all components of the code file (including the code image), and any values derived from the verification process MUST be rejected and SHOULD be immediately discarded.

6. If the manufacturer's, and optionally the co-signer's, signature has verified, the code image can be trusted and installation may proceed. Before installing the code image, all other components of the code file and any values derived from the verification process except the PKCS#7 signingTime values and the CVC validity start values SHOULD be immediately discarded.
7. The CM may upgrade its software by installing the code file according to [DOCSIS 1].
8. If the code installation is unsuccessful, the CM MUST reject the PKCS#7 signingTime values and CVC validity start values it just received in the code file. Follow the steps outlined in [DOCSIS 1] for handling this failure condition.
9. When the code installation is successful, the CM MUST update the manufacturer's time-varying controls with the values from the manufacturer's signature information and CVC:
    a) update the current value of codeAcessStart with the PKCS#7 signingTime value
    b) update the current value cvcAccessStart with the CVC validity start value
10. When the code installation is successful, IF the code file was co-signed, the CM MUST update the co-signer's time-varying controls with the values from the co-signer's signature information and CVC:
    a) update the current value of codeAccessStart with the PKCS#7 signingTime value
    b) update the current value of cvcAccessStart with the CVC validity start value

D.3.6 DOCSIS 1.0 Interoperability[30]

DOCSIS 1.1 or 2.0 cable modems MUST verify code upgrades according to this specification even when operating with a DOCSIS 1.0 environment.

DOCSIS 1.0 configuration files intended for DOCSIS 1.1 or 2.0 cable modems MUST support the configuration file requirements that are defined in this specification.

DOCSIS 1.1 or 2.0 cable modems MUST receive DOCSIS 1.1 or 2.0 compliant code files. The upgrade files pass through the DOCSIS 1.0 system untouched, and will not require modification of the DOCSIS 1.0 code file handling requirements.

In a DOCSIS 1.0 environment where DOCSIS 1.1 or 2.0 cable modems are receiving code upgrade files, the SNMP manager SHOULD support the MIBs defined for DOCSIS 1.1 or 2.0 code verification. The availability of this MIB capability is important to the proper operation and security of the DOCSIS 1.1 or 2.0 code upgrade process.

D.3.7 Error Codes

Error codes are defined to reflect the failure states possible during the code verification process. Description and usage guidelines for these error codes can be found in Annex H of [DOCSIS5] or Annex G of [DOCSIS10].[31]

30. Section D.3.6 updated per BPI-N-02098 by RKV on 8/26/02.
31. This paragraph updated per BPI-N-02098 by RKV on 8/26/02.

1. Improper code file controls
    a) CVC subject organizationName for manufacturer does not match the CM's manufacturer name.
    b) CVC subject organizationName for code co-signing agent does not match the CM's current code co-signing agent
    c) the manufacturer's PKCS#7 signingTime value is less-than the codeAccessStart value currently held in the CM
    d) the manufacturer's PKCS#7 validity start time value is less-than the cvcAccessStart value currently held in the CM
    e) the manufacturer's CVC validity start time is less-than the cvcAccessStart value currently held in the CM
    f) the manufacturer's PKCS#7 signingTime value is less-than the CVC validity start time.
    g) missing or improper extended key-usage extension in the manufacturer CVC
    h) the co-signer's PKCS#7 signingTime value is less-than the codeAccessStart value currently held in the CM
    i) The co-signer's PKCS#7 validity start time value is less-than the cvcAccessStart value currently held in the CM.
    j) the co-signer's CVC validity start time is less-than the cvcAccessStart value currently held in the CM
    k) the co-signer's PKCS#7 signingTime value is less-than the CVC validity start time
    l) missing or improper extended key-usage extension in the co-signer's CVC
2. Code file manufacturer CVC validation failure
3. Code file manufacturer CVS validation failure
4. Code file co-signer CVC validation failure
5. Code file co-signer CVS validation failure
6. Improper Configuration File CVC format
    a) Missing or improper key usage attribute
7. Configuration File CVC validation failure
8. Improper SNMP CVC format
    a) CVC subject organizationName for manufacturer does not match the CM's manufacturer name
    b) CVC subject organizationName for code co-signing agent does not match the CM's current code co-signing agent
    c) the CVC validity start time is less-than or equal-to the corresponding subject's cvcAccessStart value currently held in the CM
    d) missing or improper key usage attribute
9. SNMP CVC validation failure

D.4 Security Considerations (Informative)

The protection afforded private keys is a critical factor in maintaining security. Users authorized to sign code, i.e., manufacturers and operators who have been issued code-signing verification certificates (CVCs) by the DOCSIS root CA, must protect their private keys. An attacker with access to the private key of an authorized code-signing user can create, at will, code files that are potentially acceptable to a large number of CMs.

The defense against such an attack is for the operator to revoke the certificate whose associated code-signing private key has been learned by the attacker. To revoke a certificate, the operator must deliver to each affected CM an updated CVC with a validity start time that is newer than that of the certificate(s) being revoked. The new CVC can be delivered via any of the supported mechanisms: configuration file, code file, or SNMP MIB. The new CVC implicitly revokes all certificates whose validity start time is older than that of the new CVC.

To reduce the vulnerability to this sort of attack, it is important that an operator regularly update the CVC in each CM, at a frequency comparable to how often the operator would update a certificate revocation list (CRL) if one were available. Regular update helps manage the time interval during which a compromised code-signing key is useful to an attacker. Regardless of where you are in the CVC update cycle, CVCs should also be updated if it is suspected that a code-signing key has been compromised. To update the CVC, the user needs a DOCSIS-issued CVC whose validity start time is newer than the CVC in the CM. This implies that the DOCSIS root CA must regularly issue new CVCs to all authorized code-signing manufacturers and operators, to make the CVCs available for update. DOCSIS is likely to establish a policy about the schedule for which it issues new CVCs, and operators will likely want to coordinate their update policy with that schedule.

When a CM is attempting to register on the network for the first time or after being off-line for any amount of time, it is important that it receive a trusted CVC as soon as possible. This provides the CM with the opportunity to receive the most up-to-date CVC available and deny access to CVCs that needed to be revoked since the CM last initialization. The first opportunity for the CM to receive a trusted CVC is in its configuration file. If the configuration file does not include a valid CVC, the CM will not request or have the ability to remotely upgrade code files. In addition, the CM will not accept CVCs subsequently delivered via an SNMP MIB.

To mitigate the possibility of a CM receiving a previous code file via a replay attack, the code files include a signing-time value in the PKCS#7 structure that can be used to indicate the time the code image was signed. When the CM receives a code file signing-time that is later than the signing-time it last received, it will update its internal memory with this value. The CM will not accept code files with an earlier signing-time than this internally stored value. To upgrade a CM with a new code file without denying access to past code files, the signer may choose not to update the signing-time. In this manner, multiple code files with the same code signing-time allow an operator to freely downgrade a CMs code image to a past version (that is, until the CVC is updated). This has a number of advantages for the operator, but these advantages should be weighed against the possibilities of a code file replay attack.

Without a reliable mechanism to revert back to a known good version of code, any code-update scheme, including the one in this specification, has the weakness that a single, successful forced update of an invalid code image by a CM may render the CM useless. Even worse, an invalid code image may cause the CM to behave in a malicious way harmful to the network. Such a CM may not be repairable via a remote code update, since the invalid code image may not support the update scheme.

This page intentionally left blank.

Appendix E Upgrading from BPI to BPI+

E.1 Hybrid Cable Modem with BPI+

Some DOCSIS 1.0 CM designs may be capable of supporting BPI+ features via a software upgrade. To facilitate these “DOCSIS 1.0 Hybrid CMs”, [DOCSIS1] or [DOCSIS9] provides the Modem Capabilities Encodings which the Hybrid CM can put in the Registration Request message in order to negotiate its DOCSIS 1.1 or 2.0 features with the CMTS.[32]

A DOCSIS 1.0 Hybrid Cable Modem MAY set Privacy Support Modem Capabilities Setting to 1 (BPI Plus Support) if the CM is fully BPI+ compliant except the following points.

- support of 56-bit DES if the CM supports only 40-bit DES
- support of 1024-bit RSA key if the CM supports 768-bit RSA key
- the permanent, write-once memory for the manufacturer-issued CM certificates
- encryption of the concatenated packets if the Concatenation Support Modem Capabilities Encoding is set to 0 (off)
- encryption of the fragmentation packets if the Fragmentation Support Modem Capabilities Encoding is set to 0 (off)
- encryption of the PHS (Payload Header Suppression) packets if the Payload Header Suppression Support Modem Capabilities Encoding is set to 0 (off)

The Hybrid CM with BPI+ will be interoperable with both the BPI+ CMTS and the BPI CMTS with 56-bit and 40-bit DES. The requirement for the BPI/BPI+ interoperability in addition to the Appendix C is:

a) If a Hybrid CM with BPI+ supports only 40-bit DES and it runs in BPI+ mode, it MUST send the Auth Request message with the Security-Capabilities attribute to specify the 40-bit DES and the CMTS MUST operate with the CM in 40-bit DES mode specified in the section 7.1.

E.2 Upgrading Procedure

The BPI+ features MAY be downloaded into the DOCSIS 1.0 CM by the following procedures.

1. Download the software code image with BPI+ and BPI+ MIB features into the CM using the software downloading function defined by DOCSIS 1.0 Specification. The manufacturer CA certificate signed by the DOCSIS root private key is embedded in this software code image.
2. Set the CM certificate signed by the manufacturer’s private key and the DOCSIS Root CA's public key to the CM using the BPI+ MIB if the CM does not have these information yet. Detail of these BPI+ MIB objects for this operation will be defined by [DOCSIS 8].

Note that the CM can neither run in BPI+ mode nor set Privacy Support Modem Capabilities Setting to 1 (BPI Plus Support) until the CM certificate and the DOCSIS Root CA’s public key are set to the CM.

32. This paragraph updated per BPI-N-02098 and Alex Katsnelson by RKV on 8/26/02.

This page intentionally left blank.

Appendix F References[33]

[DOCSIS1] Data-Over-Cable Service Interface Specifications Radio Frequency Interface Specification SP-RFIv1.1-I09-020830, August 30, 2002.

[DOCSIS2] Baseline Privacy Interface Specification SP-BPI-C01-011119 (SP-BPI) November 19, 2001.

[DOCSIS3] Cable Modem Termination System - Network-Side Interface Specification SP-CMTS-NSII01-960702 (CMTS-NSI) July 2, 1996.

[DOCSIS4] Data-Over-Cable Service Interface Specifications Cable Modem to Customer Premise Equipment Interface Specification SP-CMCI-I08-020830, August 30, 2002.

[DOCSIS5] Data-Over-Cable Service Interface Specifications Operations Support System Interface Specification SP-OSSIv1.1-I08-020830, August 30, 2002.

[DOCSIS6] Data-Over-Cable Service Interface Specifications Cable Modem Telephony Return Interface Specification SP-CMTRI-I01-970804, August 8, 1997.

[DOCSIS8] Stuart M. Green, “Management Information Base for DOCSIS Cable Modems and Cable Modem Termination Systems for Baseline Privacy Plus”, draft-ietf-ipcdn-bpiplus-mib-05.txt, May 8, 2001.

[DOCSIS9] Data-Over-Cable Service Interface Specifications Radio Frequency Interface Specification SP-RFIv2.0-I02-020617, June 17, 2002.

[DOCSIS10] Data-Over-Cable Service Interface Specifications Operations Support System Interface Specification SP-OSSIv2.0-I02-020617, June 17, 2002.

[FIPS-46-2] Federal Information Processing Standard Publications 46-2, Data Encryption Standard (DES), December 30, 1993.

[FIPS-74] Federal Information Processing Standards Publication (FIPS PUB) 74, Guidelines for Implementing and Using the Data Encryption Standard, April 1981.

[FIPS-81] Federal Information Processing Standards Publication (FIPS PUB) 81, DES Modes of Operation, December 1980.

[FIPS-140-1] Federal Information Processing Standards Publication (FIPS PUB) 140-1, Security Requirements for Cryptographic Modules, April 1982.

[FIPS-180-1] Federal Information Processing Standards Publication (FIPS PUB) 180-1, Secure Hash Standard, April 1995.

[FIPS-186] Federal Information Processing Standards Publication (FIPS PUB) 186, Digital Signature Standard, 18 May 1994.

33. Appendix F updated per Alex Katsnelson by RKV on 8/22/02, 8/23/02.

[IEEE1] IEEE Std 802-1990, IEEE Standards for Local and Metropolitan Area Networks: Overview and Architecture, December 1990.

[PKCS#7] RSA Laboratories, PKCS #7: Cryptographic Message Syntax Standard, An RSA Laboratories Technical Note, Version 1.5, Revised November 1, 1993.

[RFC1750] D. Eastlake, S. Crocker, J. Schiller, “Randomness Recommendations for Security”, RFC 1750, December 1994.

[RFC2104] H. Krawczyk, M. Bellare, R. Canetti, “HMAC: Keyed-Hashing for Message Authentication”, RFC 2104, February 1997.

[RFC2202] P. Cheng, R. Glenn, “Test cases for HMAC-MD5 and HMAC-SHA-1”, RFC2202, September 1997.

[RFC3083] “BPI Management Information Base for DOCSIS Compliant Cable Modems and Cable Modem Termination Systems”, RFC3083, March 2001.

[RFC3280] R. Housley, W. Ford, W. Polk, D. Solo, “Internet X.509 Public Key Infrastructure Certificate and CRL Profile”, RFC3280, April 2002.

[RSA] RSA Laboratories, “The Public-Key Cryptography Standards”, RSA Data Security, Inc., Redwood City, CA.

[RSA1] RSA Laboratories, “PKCS #1: RSA Encryption Standard. Version 1.5”, November 1993.

[RSA2] RSA Laboratories, “Some Examples of the PKCS Standards,” RSA Data Security, Inc., Redwood City, CA, November 1, 1993.

[RSA3] RSA Laboratories, “PKCS #1 v2.0: RSA Cryptography Standard”, October 1, 1999.

[SCHNEIER]B. Schneier, Applied Cryptography, Second Edition, John Wiley, New York, 1996.

[SET Book 2]SET, Secure Electronic Transaction Specification Book 2: Programmer’s Guide, Version 1.0, May 31, 1997.

[X.509] ITU-T Recommendation X.509 (1997 E): Information Technology - Open Systems Interconnection - The Directory: Authentication Framework, June 1977.

Appendix G Acknowledgments

CableLabs and the cable industry as a whole are grateful to the following individuals and organizations for their participation and contributions in the development of this specification:

Bill Aiello, *AT&T*
Burcak Beser, *3com*
Jeff Carr, *Broadcom Corporation*
Rusty Cashman, *Correlant Communications*
Lisa Denney, *Broadcom Corporation*
Jonathan Fellows, *General Instrument Corporation*
Stuart Green, *Nortel Networks*
George Hart, *Rogers Cable TV Ltd.*
Sasha Medvinsky, *General Instrument Corporation*
John Pickens, *Com21*
Mike Sabin, *Com21*
Mike St. Johns, *@Home Corporation*
Katherine Unger, *Motorola Corporation*
Wilson Sawyer, *Nortel Networks*
Lior Storfer, *Libit Signal Processing Ltd.*
Mark Sumner, *Motorola Corporation*
Man Wong, *Cisco Systems*
James Yee, *Com21*

CableLabs would especially like to thank Kazuyoshi Ozawa of Toshiba for his unflagging efforts to develop and maintain this specification over many years. His drive and determination to create a clear, consistent, and technically sound document are a model for the industry.

This page intentionally left blank.

Appendix H Revisions

H.1 ECNs included in SP-BPI+-I02-990731

Table H-1. Incorporated ECN Table

ECN	Date Accepted
bpi-n-99022	05/06/99
bpi-n-99023	05/12/99
bpi-n-99024	05/12/99
bpi-n-99026	05/26/99
bpi-n-99027	05/26/99
bpi-n-99028	05/26/99
bpi-n-99038	06/30/99

H.2 ECNs included in SP-BPI+-I03-991105

Table H-2. Incorporated ECN Table

ECN	Date Accepted
bpi-n-99059	08/04/99
bpi-n-99063	08/04/99
bpi-n-99068	10/06/99
bpi-n-99074	09/22/99
bpi-n-99076	09/22/99
bpi-n-99077	09/29/99
bpi-n-99079	09/29/99
bpi-n-99083	10/06/99

H.3 ECNs included in SP-BPI+-I04-000407

Table H-3. Incorporated ECN Table

ECN	Date Accepted
bpi-n-99096	12/15/99
bpi-n-99099	12/01/99
bpi-n-99100	01/05/00
bpi-n-00001	02/16/00
bpi-n-00012	03/15/00
bpi-n-00013	03/15/00

H.4 ECNs included in SP-BPI+-I05-000714

Table H-4. Incorporated ECN Table

ECN	Date Accepted
bpi-n-00030	04/19/00
bpi-n-00045	05/31/00

H.5 ECNs included in SP-BPI+-I06-001215

Table H-5. Incorporated ECN Table

ECN	Date Accepted
bpi-n-00061	07/05/00
bpi-n-00062	07/05/00
bpi-n-00069	08/16/00
bpi-n-00071	08/23/00
bpi-n-00085	10/11/00
bpi-n-00098	11/08/00
bpi-n-00101	11/08/00

H.6 ECNs included in SP-BPI+-I07-010829

Table H-6. Incorporated ECN Table

ECN	Date Accepted
bpi-n-00103	01/03/01
bpi-n-00116	12/06/00
bpi-n-00131	01/03/01
bpi-n-00133	01/24/01
bpi-n-01009	02/14/01
bpi-n-01010	02/14/01
bpi-n-01011	03/07/01
bpi-n-01013	02/14/01
bpi-n-01014	02/14/01
bpi-n-01017	02/28/01
bpi-n-01021	03/14/01
bpi-n-01022	03/14/01
bpi-n-01026	03/28/01
bpi-n-01044	05/02/01
bpi-n-01045	05/02/01
bpi-n-01058	05/23/01

H.7 ECNs included in SP-BPI+-I08-020301

Table H-7. Incorporated ECN Table

ECN	Date Accepted
bpi-n-01055	11/07/01
bpi-n-01073	11/07/01
bpi-n-01094	11/21/01
bpi-n-01101	12/05/01

H.8 ECNs included in SP-BPI+-I09-020830

Table H-8. Incorporated ECN Table[34]

ECN	Date Accepted
bpi-n-02006	02/20/02
bpi-n-02008	03/06/02
bpi-n-02011	02/20/02
bpi-n-02076	05/22/02
bpi-n-02098	05/22/02
bpi-n-02108	06/26/02
bpi-n-02113	06/12/02
bpi-n-02116	06/10/02
bpi-n-02119	07/03/02

34. Beyond these ECNs, RKV made various format changes per Christie Poland on 8/29/30, 8/30/02.

This page intentionally left blank.