

OpenCable™ Specifications

Multi-Stream CableCARD™ Copy Protection System Interface

OC-SP-MCCP-IF-C01-050331

CLOSED

Notice

This document is the result of a cooperative effort undertaken at the direction of Cable Television Laboratories, Inc. for the benefit of the cable industry and its customers. This document may contain references to other documents not owned or controlled by CableLabs. Use and understanding of this document may require access to such other documents. Designing, manufacturing, distributing, using, selling, or servicing products, or providing services, based on this document may require intellectual property licenses from third parties for technology referenced in the document.

Neither CableLabs nor any member company is responsible to any party for any liability of any nature whatsoever resulting from or arising out of use or reliance upon this document, or any document referenced herein. This document is furnished on an "AS IS" basis and neither CableLabs nor its members provides any representation or warranty, express or implied, regarding the accuracy, completeness, or fitness for a particular purpose of this document, or any document referenced herein.

© Copyright 2003–2005 Cable Television Laboratories, Inc. All rights reserved.

Document Status Sheet

Document Control Number:	OC-SP-MCCP-IF-C01-050331			
Document Title:	Multi-Stream CableCARD™ Copy Protection System Interface			
Revision History:	I01 – September 5, 2003 I02 – April 2, 2004 I03 – August 31, 2004 C01 – March 31, 2005			
Date:	March 31, 2005			
Responsible Editor:	Steve Young			
Status:	Work in Progress	Draft	Issued	Closed
Distribution Restrictions:	author only	GL/Member	GL/Member/ Vendor	Public

Key to Document Status Codes:

Work in Progress	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 Members 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.

TRADE MARKS:

CableCARD™, DOCSIS®, eDOCSIS™, PacketCable™, CableHome®, CableOffice™, OpenCable™ and CableLabs® are trademarks of Cable Television Laboratories, Inc.

Contents

1	INTRODUCTION	1
1.1	Scope	1
1.2	References	1
1.2.1	Normative References	1
1.2.2	Informative References	2
1.2.3	Reference Acquisition	2
1.3	Acronyms, Abbreviations, Defined Terms and Symbols	3
1.4	Requirements (Conformance Notation)	6
1.5	Copy Protection System Components	6
1.6	Implementation Outline	6
1.7	Historical Perspective	7
1.8	Related Documents	7
2	SYSTEM OVERVIEW	8
2.1	CableCARD Device and Host Mutual Authentication	8
2.2	ID Reporting and Headend Validation	8
2.2.1	Reporting CableCARD Device and Host Identification Information	8
2.2.2	Headend ID Validation	8
2.3	Calculation of Copy Protection Keys	9
2.4	Copy Control Information	9
2.5	Encryption of Copy Protected Content	9
2.6	CableCARD-Host Messages	9
3	CABLECARD DEVICE AND HOST MUTUAL AUTHENTICATION	10
3.1	Authentication Parameters	10
3.1.1	X.509 Certificates	10
3.1.2	Copy Protection System Parameters	10
3.1.3	Binding Specific Parameters	10
3.2	Initialization	11
3.3	Diffie-Hellman Public Key Calculation	11
3.3.1	Diffie-Hellman Overview	11
3.3.2	Derivation of the Diffie-Hellman Public Keys	12
3.4	Authentication Parameter Exchange and Verification	13
3.5	Shared Binding Key Calculation	13
3.5.1	Diffie-Hellman Shared Secret Key	13
3.5.2	Shared Public Authentication Key	13
3.5.3	Verify Shared Authentication Key	13
3.6	Authentication Failures	14
3.6.1	Host Response Time-out	14

3.6.2	Invalid Certificate	14
3.6.3	Other Authentication Failures	14
3.7	CableCARD Device Operation with Multiple Hosts	15
3.8	Host Operation with Multiple CableCARD Devices	15
3.9	Backward Compatibility	15
3.10	Multi-Stream Mode	15
4	ID REPORTING, VALIDATION, AND AUTHORIZATION	16
4.1	CableCARD Device and Host Identity Reporting	16
4.1.1	Automated ID Reporting	16
4.1.2	Manual ID Reporting	16
4.1.3	Host Request for ID Reporting Screen	17
4.2	Headend Validation	17
4.2.1	ID Registration	17
4.2.2	ID Validation	17
4.3	CA Authorization of Copy Protected Content	18
5	CPS AND CRYPTOGRAPHIC FUNCTIONS	19
5.1	CPS Functions	19
5.2	Cryptographic Functions	19
5.2.1	DFAST	20
5.2.2	Random Integer Generation	20
5.2.3	SHA-1 Secure Hash Algorithm	20
5.2.4	Representation of Large Values As Octets [Informative]	20
5.3	RSA Digital Signatures	21
6	COPY PROTECTION KEY GENERATION	22
6.1	Basic Key Generation Protocol	22
6.2	Copy Protection Key Calculation	22
6.3	CPKey Refresh	23
6.3.1	CPKey Session Timer	23
6.3.2	CPKey Refresh Timer	24
6.3.3	CA Initiated CPKey Refresh	26
7	COPY CONTROL INFORMATION (CCI)	27
7.1	CCI Definition	27
7.1.1	EMI - Digital Copy Control Bits	27
7.1.2	APS - Analog Protection System	27
7.1.3	Host Set CCI Values	28
7.1.4	CIT – Constrained Image Trigger	28
7.2	Associating CCI with a Service	28
7.3	Conveying CCI from Headend to CableCARD Device	28
7.4	Conveying CCI from CableCARD Device to Host	29
7.4.1	CCI Delivery Instances	29

7.4.2	CCI Authenticated Tunnel Protocol	29
8	TRANSPORT ENCRYPTION FROM CABLECARD DEVICE TO HOST	31
8.1	MPEG Scrambling	31
8.1.1	Scrambling Rules	31
8.1.2	Transport Scrambling Control Field.....	32
8.2	Transport Processing	32
8.3	Timing of Scrambling Mode Transitions.....	32
8.4	CP-Scrambling as a Function of CA-Scrambling and EMI Value.....	33
8.5	M-CARD Mode DES_Keys	33
8.5.1	Channel Change	33
8.5.2	ODD-Even Key Synchronization Mode (Informative)	33
9	CABLECARD-HOST MESSAGING PROTOCOLS.....	35
9.1	Message Protocol Overview	35
9.2	CABLECARD-Host Message Parameters.....	36
9.3	Opening a CP Session.....	37
9.3.1	Open_Session_Request() Syntax	38
9.3.2	Open_Session_Response() Syntax	38
9.3.3	Host Capability Evaluation	39
9.4	CableCARD-Host Mutual Authentication Message Protocol	40
9.4.1	Mutual Authentication Messages.....	40
9.4.2	Host AuthKey Verification Request Messages	44
9.5	Copy Protection Key Generation	45
9.6	Host and CableCARD Device Synchronization	47
9.7	CCI Simple Authenticated Tunnel Protocol (SATP) Messages.....	48
APPENDIX A LUHN CHECK DIGIT (NORMATIVE)		53
APPENDIX B APPLYING CPKEY TO DES ENGINE (NORMATIVE).....		54
B.1	Method of Application	54
B.2	Examples of CP Encryption of MPEG DATA in Transport Packets.....	55
APPENDIX C REVISION HISTORY		56

List of Figures

Figure 1 - Diffie-Hellman Key Agreement	12
Figure 2 - Example CP System Failure Notification message	14
Figure 3 - Example ID Reporting Screen	17
Figure 4 - Overall CableCARD Copy Protection	19
Figure 5 - CableCARD CPKey Refresh Session Flow Chart	25
Figure 6 - Host CPKey Refresh Flow Chart	26
Figure 7 - CCI Delivery Sequence	29
Figure 8 - CableCARD-Host Message Protocol Flow	35

List of Tables

Table 1 - System Parameters	10
Table 2 - Length of Device Parameters in the Host Authentication	11
Table 3 - Length of Keys and Parameters Used in the Key Generation	23
Table 4 - CCI Bit Assignments	27
Table 5 - EMI Values and Content Type	27
Table 6 - APS Value Definitions	28
Table 7 - Host Default and Error CCI Value	28
Table 8 - CIT Values and Application	28
Table 9 - MPEG Transport_scrambling_control Values	32
Table 10 - CP-Scrambling Based on CA-Scrambled State and EMI Value	33
Table 11 - Message Reference Sections	36
Table 12 - CP_system_id Values	36
Table 13 - CP System Message Parameters	37
Table 14 - Copy Protection Open Session Information	38
Table 15 - Open_Session_Request() Message Syntax	38
Table 16 - Open_Session_Response() Message Syntax	39
Table 17 - Host CP Support Capability Evaluation Messages	39
Table 18 - CP_open_req() Message Syntax	40
Table 19 - CP_open_cnf() Message Syntax	40
Table 20 - CP_system_id_bitmask Values	40
Table 21 - Host Authentication Messages	41
Table 22 - CP_data_req in the Host Authentication Request Message	42
Table 23 - CP_data_cnf in the Host Authentication Response Message	43
Table 24 - Host Authentication Key Verification Messages	44
Table 25 - CP_data_req in the Authentication Key Verification Request Message	44
Table 26 - CP_data_cnf in the Authentication Key Verification Response Message	45
Table 27 - CP_data in the CPKey Generation Messages	45
Table 28 - CP_data_req() Message Syntax In the Key Generation Messages	46
Table 29 - CP_data_cnf() Message Syntax In the Key Generation Messages	47
Table 30 - Host CableCARD Device and Synchronization Messages	47
Table 31 - CP_sync_req() Message Syntax	47
Table 32 - CP_sync_cnf() Message Syntax	48
Table 33 - Status_field Value	48
Table 34 - CCI Simple Authentication Tunnel Protocol Messages	48
Table 35 - CP_data_req() Message Syntax in SATP Key Generation	49
Table 36 - CP_data_cnf() Message Syntax in CCI SATP Key Generation	50
Table 37 - CP_data_req() Message Syntax in CCI SATP Transmission	51
Table 38 - CP_data_cnf() Message Syntax in CCI SATP Transmission	52

1 INTRODUCTION

1.1 Scope

In digital cable systems, high value movies and video programs (“content”) are protected by a conditional access scrambling system. A properly authorized Multi-Stream CableCARD (M-CARD) security module removes the scrambling and, based on the Content Control Information from the Headend, may rescramble the content before delivering it to consumer receivers and set-top terminals (“Host devices”) across the CableCARD-Host interface defined in OpenCable Multi-Stream CableCARD Interface Specification [11].

This standard defines the characteristics and normative specifications for the system that prevents unrestricted copying of such high value content as it crosses the CableCARD-Host Interface. This interface supports the delivery of up to six independent transport streams across the CableCARD-Host Interface. This specification describes how copy protection of multiple programs on multiple transport streams is achieved.

Content that is delivered unscrambled over cable systems is not subject to this standard. Indeed, this standard would not provide any protection against unrestricted copying of such content. Any unscrambled content output by the Host on the CableCARD interface will not benefit by scrambling upon its subsequent output from the CableCARD device on that same interface.

This standard provides methods for authenticating CableCARD and Host devices, for binding CableCARD device to Host including Diffie-Hellman key exchange, for copy protection key generation, for rescrumbling copy protected content to resist unauthorized access (after the CableCARD device employs the conditional access system to descramble it), and then descrambling by the Host, and for transmission and authentication of Copy Control Information. It also provides for revocation of Host devices that are determined to be fraudulent or non-compliant.

This standard requires the use of the patented DFAST technology (U.S. Patent 4,860,353 and related know-how) that is available under license from CableLabs. Please refer to section 1.2.3 for contact information for such a license.

1.2 References

1.2.1 Normative References

The following standards contain provisions that, through reference in this text, constitute normative provisions of this specification. At the time of publication, the editions indicated are current. All standards are subject to revision, and parties to agreements based on this specification are encouraged to investigate the possibility of applying for the most recent editions of the standards listed in this section.

- [1] SCTE 28: ANSI/SCTE 28 2003: “Host-POD Interface Standard”
- [2] EIA 679-B, Part B: “National Renewable Security Standard”, March 2000
- [3] ITU-T Recommendation X.509, “Information Technology – Open Systems Interconnection – The Directory: Public-key and Attribute Certificate Frameworks”, March 2000
- [4] FIPS PUB 46-3: “Data Encryption Standard (DES)”, October 25, 1999
- [5] FIPS PUB 81: “DES Modes of Operation”, December 21, 1980
- [6] FIPS PUB 140-2: “Security Requirements for Cryptographic Modules”, May 25, 2001
- [7] FIPS PUB 180-1: “Secure Hash Standard”, April 17, 1995

- [8] FIPS PUB 186-2, “Digital Signature Standard” Federal Information Processing Standards Publications (FIPS PUB), January 27, 2000
- [9] RSA1, “PKCS #1: RSA Encryption Standard”, Version 1.5, November 1993
- [10] MPEG: ISO/IEC 13818-1 (2000) Information Technology - Generic coding of moving pictures and associated audio information: Systems
- [11] OC-SP-MC-IF-C01-050331, OpenCable Multi-Stream CableCARD Interface Specification, March 31, 2005
- [12] OC-SP-SEC-I05-040831, OpenCable System Security Specification, August 31, 2004
- [13] OC-SP-CCCP-IF-C01-050331, OpenCable CableCARD Copy Protection System, March 31, 2005
- [14] SCTE 41 2003: “POD Copy Protection System”

1.2.2 Informative References

The following references contain information that is useful in understanding of this specification:

- [15] RSA3, “PKCS #10 V1.7: Certification Request Syntax Standard”, May 2000

1.2.3 Reference Acquisition

CableLabs Specifications, DFAST Technology, PHILA, and PHICA: Cable Television Laboratories, Inc.

Cable Television Laboratories, Inc., 858 Coal Creek Circle, Louisville, CO 80027; Telephone: 303-661-9100; Facsimile: 303-661-9199; E-mail: opencable@cablelabs.com; URL: www.cablelabs.com

EIA Standards: Electronic Industries Association

Global Engineering Documents, World Headquarters, 15 Inverness Way East, Englewood, CO 80112-5776; Telephone 800-854-7179; Facsimile: 303-397-2740; E-mail: global@ihs.com; URL: <http://global.ihs.com>

IEEE Standards: Institute of Electrical and Electronic Engineers

Institute of Electrical and Electronic Engineers, 445 Hose Lane, Piscataway, NJ 08855-1331; E-mail: customer.service@ieee.org; URL: <http://standards.ieee.org/index.html>

ISO: International Standards Organization

Global Engineering Documents, World Headquarters, 15 Inverness Way East, Englewood, CO 80112-5776; Telephone: 800-854-7179; E-mail: global@ihs.com; URL: <http://global.ihs.com>

ITU-T: International Telecommunications Union – Telecom Standardization

International Telecommunications Union, Geneva, Switzerland. URL: <http://www.itu.int/publications/index.html>

FIPS Publications: Federal Information Processing Standards Publications

National Technical Information Service (NTIS), U. S. Department of Commerce, 5285 Port Royal Road, Springfield, VA 22161; Telephone: 1-800-553-NTIS (6847) or 703-605-6000; FAX: 703-321-8547; E-mail orders: orders@ntis.fedworld.gov; URL: <http://www.itl.nist.gov/fipspubs/>

RSA Security

RSA Security, Inc, 174 Middlesex Turnpike, Bedford, MA 01730; Telephone: 781-515-5000; FAX: 781-515-5010; URL: <<http://www.rsasecurity.com/rsalabs/pkcs>>

SCTE Standards: Society of Cable Telecommunications Engineers

Society of Cable Telecommunications Engineers, 140 Philips Road, Exton, PA 19341; Telephone: 610-363-6888; Facsimile: 610-363-5898; E-mail: standards@scte.org; URL: <<http://www.scte.org/standards/standardsavailable.html>>

1.3 Acronyms, Abbreviations, Defined Terms and Symbols

APDU	Application Protocol Data Unit: a command, query, and reply message exchange protocol between CableCARD device and Host
APS	Analog Protection System for copy control of analog output video
AuthKey	Authentication Key, calculated by both the CableCARD device and Host as part of the Host authentication process
Authentication	A procedure for the Host and CableCARD device to securely confirm that the other is a authentic device for CableCARD-CP binding. Also: a means to securely confirm that a message originated in a trusted source.
CA, CA System	Conditional Access, Conditional Access System – secures delivery of cable services to the CableCARD device
CA-only	The CableCARD device mode of CA-descrambling EMI=0 content and returning it to the Host CP-unscrambled
Cable	The Cable Television industry, services, systems, or equipment
CableCARD (CARD)	CableCARD device, also referred to as “Point of Deployment” (POD), is a detachable device distributed by cable providers, that connects to the home receiver.
CableCARD Certificate	The unique X.509 certificate issued to each CableCARD device and used for CableCARD authentication. Parameter name: CARD_DevCert.
CableCARD-CP	CableCARD copy protection, as specified in this document
CableCARD CPS	The CableCARD Copy Protection System, as specified in this document
CARD_ID	The CableCARD device’s unique identification number
CCI	Copy Control Information
CIT	Constrained Image Trigger. A CCI bit that controls image constraint on component analog outputs
CP	Copy Protection
CPKey	The Copy Protection Key derived between the CableCARD device and Host, and used by the CableCARD device to CP-scramble protected content sent to the Host
CP System	The Copy Protection System described in this specification

CRL	Certificate Revocation List: the means of reporting bad Host_IDs to cable headends
DES	Data Encryption Standard
DES-ECB	Data Encryption Standard – Electronic Code Book
DFAST	Dynamic Feedback Arrangement Scrambling Technique, a component of the encryption algorithm
DH	Diffie-Hellman, a public key agreement protocol based on the intractability of taking discrete logarithms over the integer field.
DSG	DOCSIS Set-top Gateway, a method of using DOCSIS protocols to support an out-of-band communication path.
EMI	“Encryption Mode Indicator” As used in this document the meaning of this acronym is "Copy Control" Mode Indicator for digital outputs. The acronym EMI is used by the DTLA and is retained here for consistency.
EMM	Entitlement Management Message
Encrypted	Data modified to prevent unauthorized access (compare with "scrambled")
FAT	Forward Application Transport, the 6 MHz digital channels from headend to home and between CableCARD device and Host
Headend	The cable operator’s facility which acts as the source of cable signals, services, and conditional access control.
Host	The consumer device used to access and navigate cable content. Typically a digital TV or set-top DTV receiver.
Host Certificate	The unique X.509 certificate issued to each Host device and used for Host authentication. Parameter name: Host_DevCert
Host_ID	The Host device’s unique identification number
lsb	Least Significant Bit, of a specified binary value
LTSID	Local Transport Stream ID
M-CARD	Multi-Stream CableCARD device
MMI	Man Machine Interface
MPEG	The ISO/IEC 13818 specifications and ISO/IEC 13818-1 in particular
msb	Most Significant Bit, of a specified binary value
Nonce	A random value generated fresh for each use and included in some Host-CableCARD exchanges to make each exchange unique
Pass-through	The CableCARD device mode of passing CA-scrambled back to the Host unchanged, leaving it unusable by the Host

PHI	POD-Host Interface as specified in SCTE 28
PHICA	POD-Host Interface Certificate Authority, root X.509 certificate administrator for X.509 certificates on the PHI. Identified under PHILA.
PHILA	POD-Host Interface Licensing Agreement, covers the DFAST technology and specifies the PHI Certificate Authority - PHICA
Point of Deployment (POD)	A POD, also referred to as a CableCARD device, is a detachable device distributed by cable providers, that connects to the home receiver.
RDC	Return Data Channel: a communication channel on the coaxial cable that delivers the main cable service but running “upstream” from home to the headend
Report-back	The action or process of reporting information from the CableCARD device or Host back to the headend
Rescramble	The CableCARD mode of CA-descrambling and CP-scrambling content
RSA algorithm	An RSA Security defined commercial public key cryptographic algorithm
S-CARD	Single Stream CableCARD. The PHI specified in SCTE 28 2003 and SCTE 41 2003. Devices compliant with this specification will be capable of operating in backward compatible single-stream mode compliant with those earlier specifications.
SATP	Simple Authenticated Tunnel Protocol
Scrambled	Content modified to prevent unauthorized access (compare with "encrypted")
SHA-1	Secure Hash Algorithm, a cryptographic compression function
SHALL	"SHALL" denotes a mandatory provision of this standard
Should, May	"Should" denotes a provision that is recommended but not mandatory. "May" denotes a feature whose presence does not preclude compliance and may or may not be present at the option of the implementer.
SPDU	Session Protocol Data Unit (SPDU)
SSK	A shared secret system parameter used by both CableCARD device (SSK_P) and Host (SSK_H) to authenticate the exchange of Diffie-Hellman public key parameters.
Validation	The process of reporting the Host_ID to the system operator, checking it against a revocation list, reporting the validated Host_ID back to the CableCARD device, and the CableCARD device confirming it matches the stored Host_ID.
X.509	The ITU-T Recommendation X.509 standard
XCA	X.509 certificate authority
 	Binary concatenation operator symbol, the “bar character”, means combine the bit strings of two binary parameters, e.g., for A = 111 and B = 000, A B = 111000
C^D	Exponents are displayed in superscript. C ^D means raise the first value, C, to the exponential power of the second, superscripted, value D. For C = 2 and D = 3, C ^D = 2 ³ = 8.

- ⊕ The exclusive-OR operator symbol, XOR, means the bitwise binary operation of comparing each bit starting with the lsb of each value. For each such bit pair the result is true, 1, if one or the other but not both bits is true, 1. The smaller value is padded with leading, msb, bits such that the result of the XOR operation has the same number of bits as the larger input value.

1.4 Requirements (Conformance Notation)

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

- “SHALL/MUST” These words or the adjective “REQUIRED” means that the item is an absolute requirement of this specification.
- “SHALL NOT/MUST NOT” This phrase means that the item is an absolute prohibition of this specification.
- “SHOULD” This word or the adjective “SHOULD” 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 “MAY” 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.

1.5 Copy Protection System Components

This copy protection system (CP System) includes:

- the Host (or cable content navigation device)
- the CableCARD device
- the cable headend (which revokes selected services from compromised Hosts)
- the PHICA which issues device IDs and generates X.509 manufacturer certificates.

1.6 Implementation Outline

The CableCARD CP System is based on:

- the CableCARD device and Host authenticating each other,
- sending IDs to the headend and receiving them back as validated IDs,
- calculating shared encryption keys,
- CableCARD encryption of specific copy protected content for delivery to the Host along with associated usage permissions, and
- Host decryption and protection of that content according to those permissions.

1.7 Historical Perspective

This specification has its origins in EIA-679, the National Renewable Security Standard [2], which was initially adopted in September 1998. Part B of that standard has the physical size, shape and connector of the computer industry PCMCIA card.

That standard did not take into account the requirements of the movie industry to protect against the unrestricted copying of digital video movies and programs.

Further extensions and modifications of EIA-679 led to the adoption of EIA-679-B in March 2000. EIA-679-B permits the use of copy protection techniques but does not select any single approach.

A specific approach was embodied in DVS/213 and proposed to SCTE DVS in June 1999. Extensive revisions were developed by a cable industry group, and submitted as DVS/301 in January 2000. Continued work and revision proceeded during the first half of 2000, leading to substantial changes that were embodied in DVS/301r1, r2 and finally DVS/301r3 which was successfully balloted and adopted as ANSI/SCTE 41 [13]. This document updates that work to current industry practice per OC-SP-MC-IF [11] and adds the capability to copy protect simultaneous delivery of multiple content streams across the CableCARD-Host interface.

1.8 Related Documents

This document is intended to supplement the functionality of the CableCARD (POD) interface described in SCTE 28 [1] and [11] which define how copy protection fits into the overall CableCARD interface functionality.

2 SYSTEM OVERVIEW

2.1 CableCARD Device and Host Mutual Authentication

The CableCARD device and Host authenticate each other by a process of exchanged messages, calculations using stored secrets, and confirmations that the results meet specific criteria.

The CableCARD device and Host each:

- calculate a Diffie-Hellman public key from stored secrets and a generated random integer,
- sign their DH public key with their X.509 private key (that matches the public key embedded in their X.509 device certificate),
- send their list of X.509 Certificate, DH public key, and signature of that key to the other device,
- verify the signatures of the other device's X.509 certificates and DH public key,
- calculate a shared secret key, DHKey, that is unique to each binding,
- calculate and exchange a long term authentication key, "AuthKey",
- confirm that the received AuthKey matches the internally calculated AuthKey.

When the steps above are completed the CableCARD device and Host share DHKey and AuthKey values and the binding is "authenticated."

2.2 ID Reporting and Headend Validation

The CableCARD device and Host identification numbers, typically CARD_ID and Host_ID, must be reported to the cable operator to validate the device pair and allow reception of copy-protected content. The IDs are compared to a list of devices whose copy protection security is compromised.

2.2.1 Reporting CableCARD Device and Host Identification Information

The CableCARD device extracts the CARD_ID and Host_ID from the authenticated X.509 device certificates and sends them to the headend. Two means are employed to report these IDs. In order of preference:

- Two-Way: If the cable system has an active RDC (legacy or DSG) and the Host has an RDC transmitter, the CableCARD device MAY use them to report the IDs to the headend.
- Manual: If an automated means is not available, the CableCARD device will display the ID information on the subscriber's TV screen with a reporting telephone number and a request for the subscriber to call the operator to manually report the ID information.

2.2.2 Headend ID Validation

The cable operator records the reported IDs and their binding as a copy protection pair. If the devices are authorized to receive copy-protected content, the CA System will send a validation message to the CableCARD device with the validated IDs. After receiving and authenticating this message the CableCARD device checks that the received IDs are the same as its current binding and if so enables delivery of copy protected content to the bound Host.

2.3 Calculation of Copy Protection Keys

The CableCARD device and Host each derive the Copy Protection Key (CPKey) based on random integers exchanged for this purpose, and the binding specific secret DHKey and public AuthKey. The resulting CPKey is unique to the particular CableCARD-Host pair and to the key session. Content scrambled with this key by the CableCARD device will be useful only to its bound Host

The CPKey is changed/refreshed by calculating a new key based on new random integers generated by each device. The CableCARD device initiates calculation of a new CPKey after its initial binding to a valid Host, periodically by a refresh clock, at every power-up, and on command by the CA System.

CPKey calculation relies upon a set of cryptographic techniques including the SHA-1 and DFAST algorithms.

2.4 Copy Control Information

Copy control information (CCI) is passed from CableCARD device to Host across the data channel to inform the Host device of the level of copy protection required. CCI is sent in the clear to the Host device, but the integrity of the information is authenticated with a simple protocol.

The one-byte CCI field contains information that the Host uses to control copying of content. Two EMI bits control copying on Host digital outputs (that support EMI), two APS bits control copying on analog outputs, one bit as a Constrained Image Trigger, and three bits are reserved.

2.5 Encryption of Copy Protected Content

All copy protected content will arrive at the Host device as a data stream encrypted by the service providers CA System. This CA-encrypted stream passes from the Host to the CableCARD device while still protected by the CA System. When fully bound to an authenticated and validated Host and authorized by the CA System the CableCARD device removes the CA-encryption.

The CableCARD device uses encryption techniques to scramble copy-protected content transmitted across its interface with the Host. The CableCARD device descrambles the content it has descrambled under the conditional access system. CP-scrambling uses DES in ECB mode and the computed Copy Protection Key before sending it across the interface to the Host.

The CableCARD device passes content in one of four modes determined by CA System scrambling mode and EMI values as detailed in Table 10.

- Clear: no change of CA-unscrambled and EMI=0 content which remains 'in-the-clear'
- CA-only: descrambles CA-scrambled content marked EMI=0 for output 'in-the-clear'
- Rescramble: CA-descrambles and CP-scrambles content marked EMI>0
- Pass-through: no change of CA-scrambled content (leaving it unrecognizable to the Host)

When selected, data packets of all selected and copy protected content from multiple simultaneous programs are protected in the same manner with the same CPKey.

2.6 CableCARD-Host Messages

Messages between the CableCARD device and Host on the data channel and extended channel do not require protection and are exchanged without encryption.

3 CABLECARD DEVICE AND HOST MUTUAL AUTHENTICATION

3.1 Authentication Parameters

Three types of parameters are employed in the mutual authentication of CableCARD device and Host devices.

3.1.1 X.509 Certificates

The CableCARD device and Host each contain an X.509 version 3 device certificate and a matching private key used for digital signatures. The certificate includes a unique ID and the public key provided to other devices to validate the digital signatures. Each device also has the Manufacturer CA certificate that was used to sign the device certificate, and the PHICA root certificate that is used to sign the Manufacturer CA certificate.

3.1.2 Copy Protection System Parameters

Table 1 defines system parameter length and source:

Table 1 - System Parameters

Parameter	Size (bits)	Source of Parameter
CARD_ID (part of device certificate)	64 bits	PHICA and manufacturer
Host_ID (part of device certificate)	40 bits	PHICA and manufacturer
Diffie-Hellman prime (n)	1024 bits	PHICA
Diffie-Hellman base (g)	1024 bits	PHICA
RSA public signing key exponent	40 bits	PHICA

The CableCARD device manufacturer SHALL set the 24 most significant bits of the 64-bit CARD_ID, bits numbered 63 through 40, to zero. The CableCARD device and Host manufacturer SHALL set bits 39 through 30 to their PHICA assigned manufacturer number and bits 29 to 0 to a unit number that is unique to that manufacturer number. Only unit numbers less than decimal 1,000,000,000 (one billion) SHALL be assigned.

3.1.3 Binding Specific Parameters

The following binding specific parameters are calculated from fixed and random values for each CableCARD-Host binding.

DH_pubKey_C The CableCARD Diffie-Hellman public key. Derived and unique for each CableCARD-Host binding.

DH_pubKey_H The Host Diffie-Hellman public key. Derived and unique for each CableCARD-Host binding.

DHKey The Diffie-Hellman Shared Private Key

AuthKey_C The authentication key derived by the CableCARD device, before verifying that it equals AuthKey_H.

AuthKey_H The authentication key derived by the Host, before verifying that it equals AuthKey_C.

AuthKey	The shared public authentication key.
x, y	Diffie-Hellman private values for CableCARD device and Host. Random integers generated uniquely for each binding.

Table 2 - Length of Device Parameters in the Host Authentication

Parameter	Size (bits)
Diffie-Hellman Private Values (x,y)	160
Diffie-Hellman Public Keys (DH_pubKey _H , DH_pubKey _C)	1024
Diffie-Hellman Shared Private Key (DHKey)	1024
Authentication Key (AuthKey)	160

3.2 Initialization

CableCARD-Host Initialization occurs on each power-up or CableCARD device insertion as follows:

1. The Host SHALL report copy protection as a resource during the profile inquiry process. Upon such report, the CableCARD device SHALL initiate mutual authentication with the Host and SHALL prohibit CA decryption of all CA-scrambled content until the authentication process is complete.
2. The CableCARD device SHALL open a session to the copy protection resource. See Section 9.3.
3. The CableCARD device SHALL evaluate the Host's support for the CableCARD-CP system. See Section 9.3.3.
4. If the CableCARD device contains a stored, non-zero, AuthKey in its non-volatile memory, it SHALL request the Host's AuthKey. If not, the CableCARD device SHALL record the non-authenticated and non-validated states in nonvolatile memory and proceed with Diffie-Hellman public key calculation.
5. The Host SHALL respond with its AuthKey, if available. If it is not available, then it SHALL respond with a value of zero.
6. The CableCARD device SHALL compare the Host's AuthKey with its stored AuthKey.
 - a. If they are identical, authentication is complete, as restored from a previous authentication. The CableCARD device SHALL store AuthKey in non-volatile memory, enable CA-decryption of MPEG programs with EMI=00, and proceed with ID reporting. See Section 4.
 - b. If the Host's AuthKey does not match the CableCARD device stored AuthKey, the CableCARD device SHALL set AuthKey, Validated_CARD_ID, and Validated_Host_ID to zero, record them in non-volatile memory, and proceed with Diffie-Hellman key exchange.

3.3 Diffie-Hellman Public Key Calculation

3.3.1 Diffie-Hellman Overview

The Diffie-Hellman Public Key Agreement procedure provides a method for CableCARD device and Host to compute a long-term shared secret, DHKey, that is used in authentication. The Diffie-Hellman protocol provides the system with a cryptographic property known as "perfect forward secrecy". Computing the private exponents from the public values is computationally infeasible. Figure 1 illustrates the two-step Diffie-Hellman operations conducted between the CableCARD device and Host.

3.3.2 Derivation of the Diffie-Hellman Public Keys

The CableCARD device and Host derive the Diffie-Hellman public keys and their signatures as follows:

1. The CableCARD device SHALL generate a random private value, x , where $1 \leq x \leq n - 2$.
2. The Host SHALL generate a random private value, y , where $1 \leq y \leq n - 2$.
3. The CableCARD device SHALL compute its DH public key as: $DH_pubKey_P = g^x \text{ mod } n$.
4. The Host SHALL compute its DH public key as: $DH_pubkey_H = g^y \text{ mod } n$.
5. The CableCARD device SHALL derive $SIGN_P$ by signing DH_pubKey_P with its X.509 private key.
6. The Host SHALL derive $SIGN_H$ by signing DH_pubKey_H with its X.509 private key.

Note: steps 1 and 2 differ slightly from SCTE 41 [13].

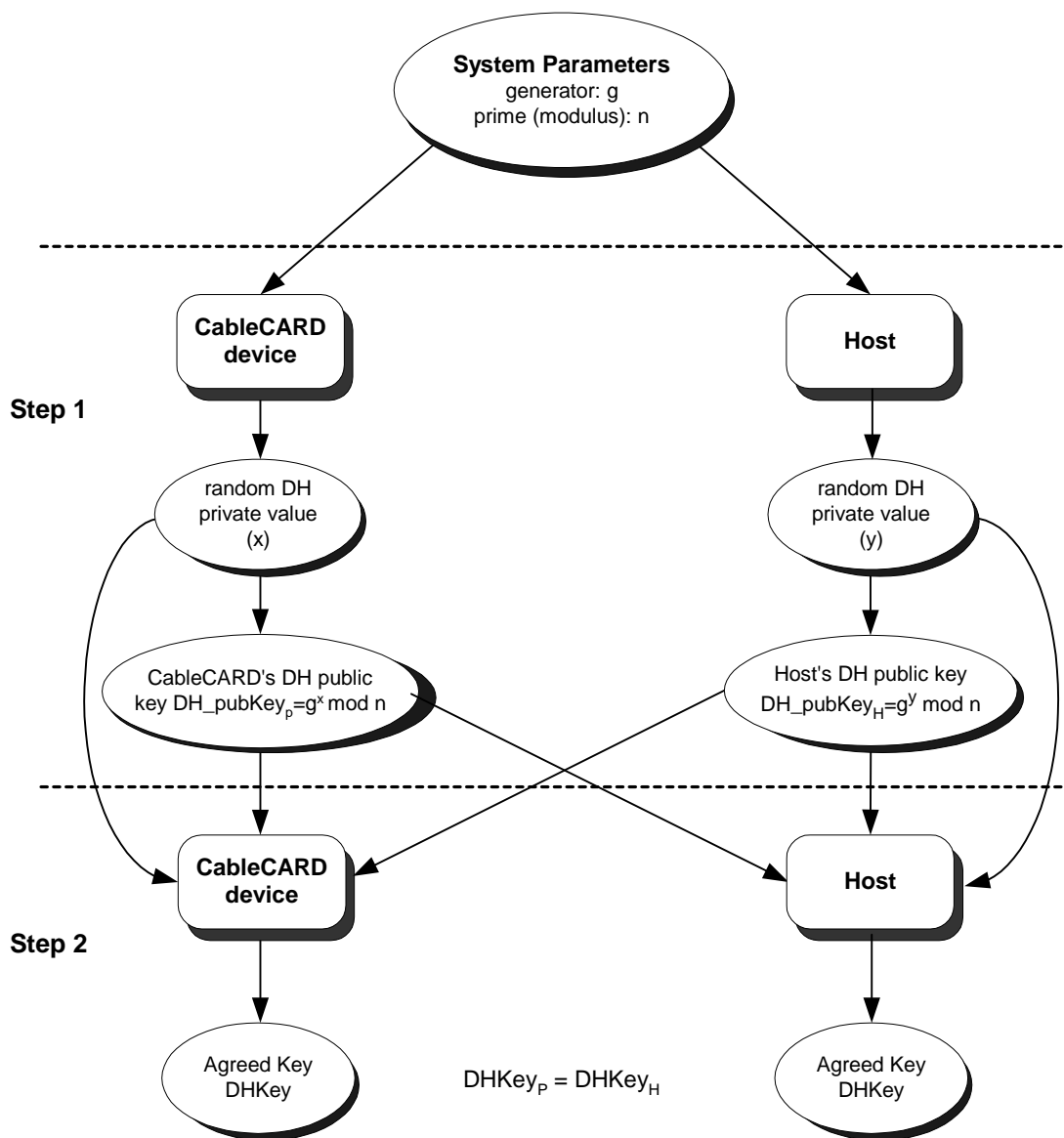


Figure 1 - Diffie-Hellman Key Agreement

3.4 Authentication Parameter Exchange and Verification

The CableCARD device and Host exchange data and each verify the other's X.509 certificates and signature:

1. The CableCARD device SHALL send its certificate data (CARD_DevCert and CARD_ManCert), DH_pubKey_P, and SIGN_P to the Host with a request for the Host's authentication parameters.
2. The Host SHALL reply with its certificate data (Host_DevCert and Host_ManCert), DH_pubKey_H and SIGN_H.
3. The CableCARD device SHALL verify Host_DevCert, Host_ManCert, and SIGN_H and extract the Host_ID from the Host certificate.
4. The Host SHALL verify CARD_DevCert, CARD_ManCert, and SIGN_P and extract the CARD_ID from the CableCARD device certificate.

3.5 Shared Binding Key Calculation

3.5.1 Diffie-Hellman Shared Secret Key

The CableCARD device and Host SHALL each compute the DH shared secret key, DHKey, using the other's DH public key, their own DH private value, and the DH prime system parameter, n , as follows:

1. The CableCARD device derives the 1024 bit shared key $DHKey = (DH_pubKey_H)^x \bmod n$; and
2. The Host derives the 1024 bit shared key $DHKey = (DH_pubKey_C)^y \bmod n$;

The CableCARD device and Host SHALL store DHKey in non-volatile memory for CPKey generation and refresh.

Even though both the CableCARD device and Host are making computations using different private values (x , y), both calculations result in the same shared secret key value: DHKey

$$(DH_pubKey_H)^x \bmod n = (g^y \bmod n)^x \bmod n = g^{yx} \bmod n = DHKey$$

$$(DH_pubKey_C)^y \bmod n = (g^x \bmod n)^y \bmod n = g^{xy} \bmod n = DHKey$$

3.5.2 Shared Public Authentication Key

The CableCARD device and Host each compute the shared public key, AuthKey, from the shared secret DHKey, CARD_ID, and Host_ID, as follows:

1. The CableCARD device computes its authentication key by applying the SHA-1 function:

$$AuthKey_C = \text{SHA-1} [DHKey | Host_ID | CARD_ID]$$

2. The Host computes its authentication key by applying the SHA-1 function:

$$AuthKey_H = \text{SHA-1} [DHKey | Host_ID | CARD_ID]$$

The Host SHALL store AuthKey_H in non-volatile memory for CPKey generation and refresh.

3.5.3 Verify Shared Authentication Key

The CableCARD device SHALL request and the Host SHALL provide AuthKey_H. The CableCARD device SHALL compare AuthKey_H to AuthKey_C. If they are identical the CableCARD device SHALL store AuthKey in non-volatile memory. This completes CableCARD-Host mutual authentication and the Card SHALL initiate CPKey generation and enable CA-decryption of MPEG programs with zero EMI.

If the binding is not validated, the CableCARD device SHALL initiate ID reporting or resume ID reporting attempts.

3.6 Authentication Failures

If the Host is in an off-state or any non-video-viewing state, it SHALL deny any MMI dialog open request. When the Host is in a video viewing state, it SHALL grant the open MMI dialog request.

3.6.1 Host Response Time-out

If the Host fails to respond to any APDU within 5 seconds the CableCARD device SHALL set the IIR flag. See [1].

3.6.2 Invalid Certificate

In the event that the CableCARD device supplies an invalid certificate to the Host, the Host SHALL display a message informing the user, for example:

**CableCARD Device Certificate
Invalid**

In the event that the Host supplies an invalid certificate to the CableCARD device, the CableCARD device SHALL request the copy protection message screen and display a message informing the user, for example:

Host Certificate Invalid

3.6.3 Other Authentication Failures

If any other part of mutual authentication procedure described above fails, including signature or AuthKey verification, the CableCARD device SHALL take the following steps:

1. Notify the headend by automated means (RDC) if possible,
2. Open a session to the Host's MMI resource and MMI dialog (if not already open),
3. Display a message to the subscriber similar to that shown below in Figure 2.

**There was a technical problem during the
authorization process.**

**This product may have some component
failure or may not be designed to be fully
compatible with digital cable television
services. Please contact the manufacturer or
the retailer.**

Figure 2 - Example CP System Failure Notification message

Following display at initial failure, the CP system failure notification message SHALL be displayed only if authentication has failed and 1) the message is selected through a user interface menu, or 2) the user tunes to a scrambled channel protected by the CA System.

3.7 CableCARD Device Operation with Multiple Hosts

Each CableCARD device SHALL bind to exactly one Host at a time. No CableCARD device SHALL store two or more sets of Authentication Keys or other Host-specific information. A given CableCARD device can be removed from a Host and inserted into a different Host at any time. The re-authentication procedure will indicate a mismatch in authentication keys, and the CableCARD device SHALL initiate the binding procedure, including full Host Certificate verification. If this CableCARD device is later returned to the previous Host it SHALL again initiate the binding procedure, as it has authentication information only on the last Host to which it was bound.

3.8 Host Operation with Multiple CableCARD Devices

Hosts may operate with multiple CableCARD devices. The Host SHALL perform all aspects of this specification independently for each CableCARD device that it supports. This includes that the Host SHALL have a unique X.509 Device Certificate and its associated private key for each supported CableCARD device interface. Each certificate SHALL have a unique Host_ID. The Host SHALL generate, exchange and calculate, separate and independent DH private and public values, DHKey, AuthKey, nonces, and CPKeys for each supported CableCARD device. Note: future versions of this specification may define other ways for a Host to support multiple CableCARD devices.

3.9 Backward Compatibility

The CableCARD-Host interface supports two modes of operation:

1. the multi-stream mode interface, which supports multiple transport streams across the interface, and
2. a backward compatible mode, which is the copy protection operation as defined in the Host-POD Interface Standard [1].

When the M-CARD device is operating in backward compatible mode, it SHALL support the functionality as defined in the OpenCable CableCARD Copy Protection Specification [13]. It SHALL support the copy protection resource identifier and APDUs as defined in [1].

3.10 Multi-Stream Mode

When the M-CARD device is operating in multi-stream mode, the copy protection system operation SHALL be as defined in this document.

In multi-stream mode, there is a unique DES_Key generated for each MPEG program being protected by the copy protection system.

4 ID REPORTING, VALIDATION, AND AUTHORIZATION

The Host_ID and CARD_ID must be reported to the service provider before the CableCARD device will provide copy protected content to the Host, that is content with non-zero EMI. The retailer may perform the reporting service for the subscriber.

Following authentication, the CableCARD device SHALL check the reporting and validation status in non-volatile memory. If validation is complete, the CableCARD device SHALL proceed with CPKey refresh.

4.1 CableCARD Device and Host Identity Reporting

In cable systems with RDC functionality, where the Host, cable plant, and headend all support compatible connections, the Host_ID and CARD_ID may be sent to the headend in an authenticated CA System message.

For one-way cable systems, unidirectional Hosts, or any system without an automatic report-back mechanism, the CableCARD and Host IDs must be reported manually, typically by the subscriber.

The CableCARD device SHALL report identification information, including CableCARD-ID and Host_ID, to the headend by one of the following automated or manual means. The CableCARD device SHALL store CARD_ID and Host_ID in non-volatile memory so they can be compared with the validated IDs received back from the headend. The CableCARD device SHALL store the status of reporting attempts in non-volatile memory to prevent unnecessary retransmissions.

4.1.1 Automated ID Reporting

The CableCARD device SHALL send the CARD_ID and Host_ID to the cable headend as a private CA message via the RDC or telco modem.

4.1.2 Manual ID Reporting

The CableCARD device SHALL display the “ID Reporting Screen” to the subscriber via the Host MMI resource. The ID Reporting Screen SHALL include the CARD_ID, Host_ID, a reporting telephone number and any other information required to identify the CableCARD device and Host to the CA System.

If the Host is in an off state or any non-video viewing state, it SHALL deny the MMI dialog open request. When the Host is in a video viewing state, it SHALL grant the open MMI dialog request. The Host SHALL display the message and confirm to the CableCARD device that the message has been displayed to the customer.

The CableCARD device SHALL display the CARD_ID and Host_ID as 13-digit decimal numbers with display sequence M-MMU-UUU-UUU-UUL^{1,2}, where:

- M-MM is the decimal representation of the 10-bit PHICA assigned manufacturer number,
- U-UUU-UUU-UU is the decimal representation of the 30-bit manufacturer assigned unit number, and
- L is a Luhn check digit calculated over the preceding 12 digits, see Appendix A.

The ID Reporting Screen SHALL be displayed only if:

¹ Note: It is the sequence of the defined values that is specified. The use of dashes as the delimiter is shown with an example placement to be consistent with the examples used elsewhere in this specification and SCTE-41.

² See the OpenCable System Security Specification [12] for details on use of binary and decimal values in the POD_ID and Host_ID.

- the message is selected through a user menu system,
- the user selects a program with CP active (non-zero EMI) before the Host is validated, or
- the CableCARD device initiates the message display, e.g., at the request of the CA System.

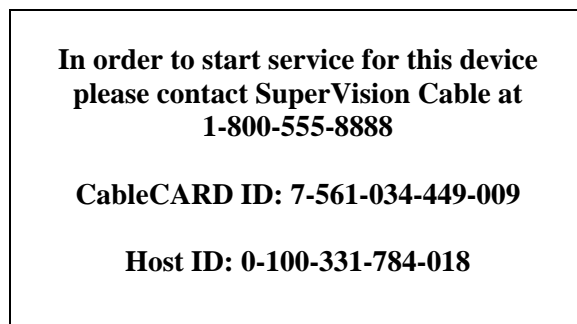


Figure 3 - Example ID Reporting Screen³

4.1.3 Host Request for ID Reporting Screen

If the Host requests the ID Reporting Screen and authentication is complete, the CableCARD device SHALL display the same information and format defined above for Manual ID Reporting. If authentication is not complete, the CableCARD device SHALL display “Information not available.”

In order to support a Host request to display the ID Reporting Information Screen, the CableCARD device SHALL support the “Copy Protection” with `application_type = 0x01` as specified in [11]. The ID Reporting information application SHALL be defined strictly as display of the ID Reporting information screen as defined in this section above. The CableCARD device SHALL only support one (1) ID Reporting information application and therefore only one application with `application_type = 0x01`.

4.2 Headend Validation

4.2.1 ID Registration

The CA System records the binding of the `Host_ID` and `CARD_ID`. If the headend CA System receives a new revocation list, it SHALL examine all previously reported `Host_IDs` and if there are any matches, it SHALL notify the cable operator.

4.2.2 ID Validation

The CA System compares the `Host_ID` and `CARD_ID` to the ID component of any X.509 certificates on its current certificate revocation list. If not found, the CA System sends `Validated_CARD_ID = CARD_ID` and `Validated_Host_ID = Host_ID` to the CableCARD device in a private authenticated CA System ID validation message.

The ID validation message may be sent to the CableCARD device substantially later in time than when the CableCARD device and Host IDs are reported to the headend.

The CableCARD device SHALL authenticate the ID validation message and confirm the validated ID values match its current stored `CARD_ID` and `Host_ID`. If they match, validation is complete: the CableCARD device SHALL enable CA decryption of authorized content with EMI values of 01, 10, and 11 and proceed with CPKey generation.

³ The text portion of this screen is shown as an example only; there is no implied requirement to duplicate the exact wording or formatting shown. All three numeric fields MUST be included as defined above.

If they do not match, the CableCARD device SHALL continue to limit its CA decryption to content with EMI value 00.

4.3 CA Authorization of Copy Protected Content

The service provider authorizes the CableCARD device for services, programs and content through its conditional access system. The CA System will deny authorization for specific protected content to any CableCARD device or Host whose CP security is considered insufficient for reception of that content.

The headend always has an opportunity to revoke content or services using CA System EMMs. CRLs are used in the headend only. The CA System headend can receive new CRLs, look up new revoked IDs, and revoke selected services from any compromised device.

5 CPS AND CRYPTOGRAPHIC FUNCTIONS

5.1 CPS Functions

The CableCARD device SHALL comply with a CA System request to initiate full copy protection reinitialization, as if the CableCARD device were inserted into its Host for the first time. When this occurs, the CableCARD device SHALL disable all CA-decryption, clear its stored AuthKey and initiate mutual authentication as if it had never before been registered or authenticated with that specific Host. For one-way cable systems or Uni-Directional Hosts this will require the consumer to call the operator to report the Host and CableCARD IDs for validation.

If a CableCARD device reports a failure of the copy protection system to the headend CA System, the headend CA System will notify the cable operator.

5.2 Cryptographic Functions

The basic key negotiation process for CableCARD copy protection is shown in Figure 4 below.

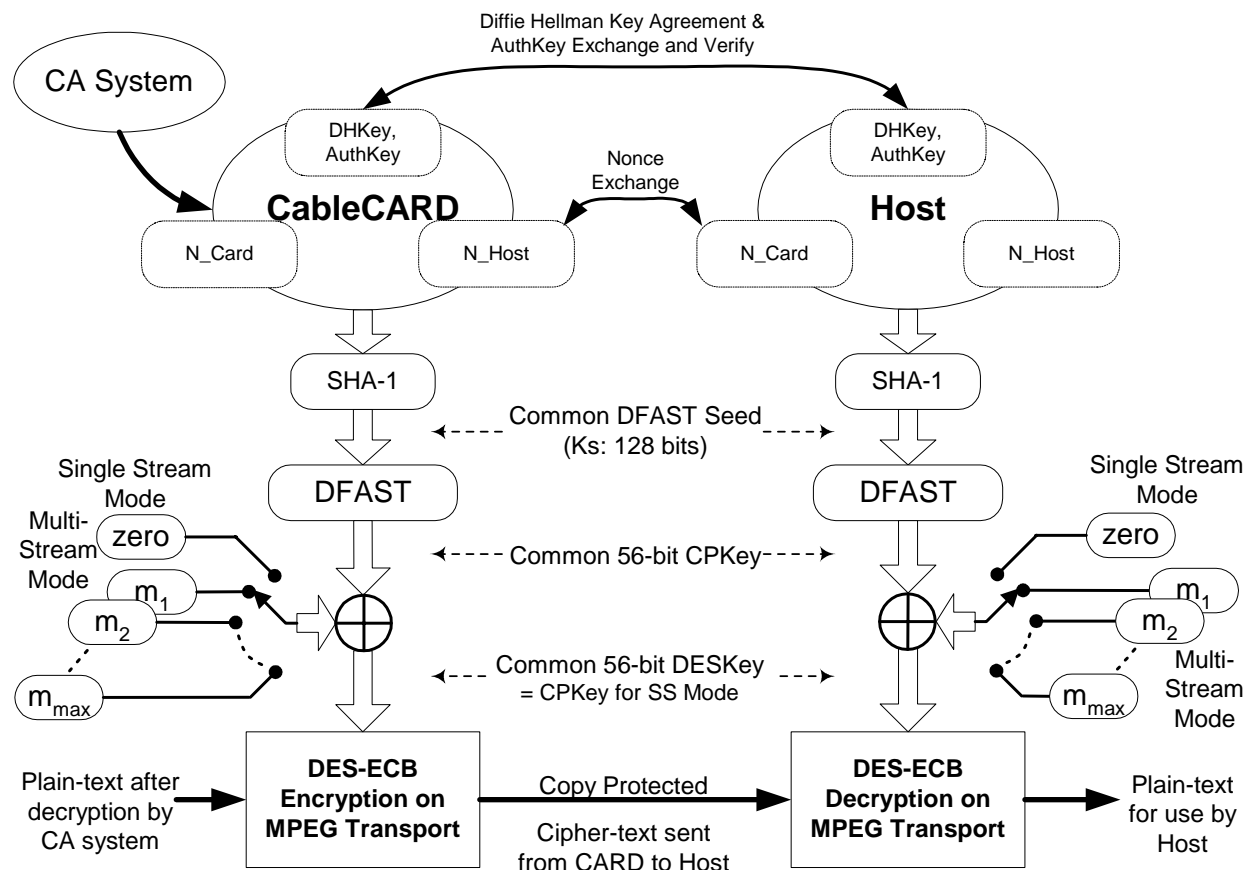


Figure 4 - Overall CableCARD Copy Protection

5.2.1 DFAST

DFAST, the Dynamic Feedback Arrangement Scrambling Technique, is used in the generation of unique cryptographic keys to seed the DES-ECB encryption and decryption functions as shown in Figure 4. Detailed information on DFAST design and implementation is presented in the document “DFAST Implementation Description” obtained from the PHICA.

The DFAST function accepts a 128-bit input value (K_s) and generates 56 bits of output (CPKey). This output from the DFAST function is used as the DES ECB key for copy protection content scrambling and descrambling.

5.2.2 Random Integer Generation

Random integers are required for use as the DH private keys (x and y), the nonces used in CPKey generation, and the CCI transfer nonces. The CableCARD device and Host SHALL each implement a physical or a pseudorandom (software) integer generator. Pseudorandom generators SHALL be compliant with the SHA-1 based algorithm described in [8], Appendix 3, Section 3.3, and include a unit-unique secret seed. Physical random integer generators SHALL comply with [6], Section 4.7.1 test for randomness.

5.2.3 SHA-1 Secure Hash Algorithm

The CableCARD Copy Protection specification employs the RSA signature algorithm with SHA-1 for all X.509 digital certificates. The CableCARD Copy Protection specification uses F4 (65537 decimal, 010001 Hex) as the public exponent for its signing operation. F4 SHALL be padded with most significant zeros to create a 40-bit exponent. The Device Root PHICA will employ a modulus length of 2048 bits for signing the Manufacturer XCA certificates it issues. Manufacturer XCAs SHALL employ signature key modulus length of 2048 bits.

The following functions and operations use the SHA-1 algorithm:

- Host Certificate Signature Verification: the signature algorithm is based on the RSA digital signature scheme defined in [7], which uses the SHA-1 primitive.
- CableCARD Certificate Signature Verification: the signature algorithm is based on the RSA digital signature scheme defined in [7], which uses the SHA-1 primitive.
- Authentication Key generation as described in Section 3.5.2.
- Copy Protection Key generation, as described in Section 6.

5.2.4 Representation of Large Values As Octets [Informative]

To represent large parameter values, like the 1024-bit modulus, as a series of octets (bytes) the most significant bit (msb) of the first octet should represent the msb of the value, the least significant bit (lsb) of the first octet the eighth msb of the value, continuing until the lsb of the value becomes the lsb of the last octet. In other words, the first octet in the series has the most significance in the integer and the last octet has the least significance.

A large parameter z of length $k \cdot 8$ bits should be converted into an octet block PV of length k , where PV_1, \dots, PV_k are the octets of PV from first to last, such that:

$$z = \sum_{i=1}^k 2^{8(k-i)} PV_i$$

5.3 RSA Digital Signatures

RSA digital signatures SHALL be computed using block type 01 as specified in PKCS #1 version 1.5 [9].

6 COPY PROTECTION KEY GENERATION

The CableCARD device SHALL generate and refresh the CPKey at the following times:

- After completion of the validation process;
- Periodically at a rate set by max_key_session_period;
- At every power cycle;
- When initiated by the CA System; and
- At every hard reset.

Highly randomized variables are used as new random integers (“nonces”). These nonces along with IDs are exchanged between the CableCARD device and Host interface. A common Copy Protection Key between the CableCARD device and Host is derived from these newly exchanged random integers, the Authentication Key (AuthKey_c or AuthKey_H) and the 1024 bit shared secret Diffie-Hellman key (DHKey). The derived common Copy Protection Key (CPKey) is then used to encrypt/decrypt MPEG data sent from the CableCARD device to the Host in backward compatible S-CARD mode.

In multi-stream mode the CableCARD device and Host will derive a unique DES key, DES_Key[m], for each copy-protected program, where ‘m’ = (LTSID | program_number).

6.1 Basic Key Generation Protocol

The CableCARD device and Host SHALL perform the following procedure to generate the CPKey:

- The CableCARD device checks for a previously derived AuthKey stored in non-volatile memory. If such an AuthKey_c is not present, then follow the whole authentication process as detailed in Section 3.2.
- The CableCARD device checks validation status in non-volatile memory. If validation is complete, then continue to the next step, otherwise return to the validation procedure, Section 4.
- The CableCARD device generates its 64 bit random integer (N_CARD).
- The CableCARD device sends this N_CARD and its ID (CARD_ID) in the clear to the Host.
- The Host generates its 64 bits random integer (N_Host).
- The Host sends N_Host and its Host_ID in the clear to the CableCARD device.
- The CableCARD device checks if the received Host_ID is equal to the previously stored ID. If they are the same, CableCARD device SHALL proceed with the key generation process; otherwise, the CableCARD device SHALL CA decrypt only services with EMI value of 00.
- The CableCARD device computes the Copy Protection Key based on long-term keys and newly exchanged random integer using the SHA-1 hash function and the DFAST algorithm, as described in the following section.
- The Host computes the Copy Protection Key also based on long-term keys and newly exchanged random integer using the SHA-1 hash function and the DFAST algorithm, as described in the following section.

6.2 Copy Protection Key Calculation

The CableCARD device and Host SHALL each calculate the DFAST seed value, Ks, as:

$$Ks = \text{SHA-1} [\text{AuthKey} | \text{DHKey} | \text{N_Host} | \text{N_CARD}]_{\text{msb128}}$$

Truncating the 160-bit SHA-1 output to its 128 msb, left-most bits, generates a seed, Ks, with the proper 128-bit length for the input to the DFAST engine. The CableCARD device and Host SHALL apply the DFAST function to Ks to produce Copy Protection Key, CPKey:

$$\text{CPKey} = \text{DFAST} [\text{Ks}]$$

DFAST details are specified in a separate document; contact the PHICA. Table 3 defines the size of keys, as well as the parameters used to derive them.

Table 3 - Length of Keys and Parameters Used in the Key Generation

Parameter	Size (bits)	Description
Nonces (N_Host, N_CARD)	64 bits each	Random integers unique to each calculation of CPKey.
DFAST Seed (Ks)	128 bits	The 128 most significant bits of the SHA-1 output, where the SHA-1 input is AuthKey, DHKey, and N_Host and N_CARD.
Copy Protection Key (CPKey)	56 bits	DFAST output, final copy protection encryption and decryption key

6.3 CPKey Refresh

The CPKey SHALL refresh periodically as initiated by the CableCARD device. The CA System will set the refresh period with a parameter, max_key_session_period, transmitted to the CableCARD device by the CA System with maximum security.

The CableCARD device SHALL initiate each CPKey refresh cycle and start a key refresh timer. The CableCARD device SHALL stop scrambling the selected program during the synchronization of keys. It SHALL start to encrypt again on the earlier of 1) successful completion of the authenticated CPKey refresh cycle, or 2) transmitting unencrypted data for one second. The CCI SHALL NOT be changed during the period of the key refresh timer.

Each CPKey refresh SHALL recalculate the content key using a new pair of nonces (N_Host, N_CARD) exchanged between the CableCARD device and Host.

The CP keys SHALL be refreshed sequentially, one at a time, until all keys have been refreshed.

6.3.1 CPKey Session Timer

The CPKey session period is the length of time the CableCARD device and Host use a given CPKey. The CA System SHALL set the value of the parameter max_key_session_period. The CableCARD device SHALL implement a CPKey Session Timer and reset it each time a new CPKey is generated. This timer is not dependent on the program selected by the Host.

If this timer reaches the value of the max_key_session_period, the CableCARD device SHALL initiate a CPKey refresh. The max_key_session_period is a 16-bit value with a resolution of 10 seconds (one decasecond). If the value of max_key_session_period is zero the maximum CPKey session period is unlimited. The Host is not aware of max_key_session_period.

The same key session period SHALL apply to all CPKeys on the interface. It is a global parameter for all CPKeys.

6.3.2 CPKey Refresh Timer

The CableCARD device SHALL start a Key Refresh Timer when the CableCARD device sends its nonce to the Host in the CP_data_req() message. When the Host receives the CP_data_req(), the Host generates its nonce and sends it to the CableCARD device in the CP_data_cnf() message. The Host SHALL reply with a CP_data_cnf() message within one second of receiving a CP_data_req() message.

When the Host issues the CP_data_cnf() the CableCARD device and Host SHALL start the calculation of the Copy Protection Key. The CableCARD device and Host SHALL each calculate CPKey within eight seconds. When the Key Refresh Timer reaches nine seconds the CableCARD device SHALL send the CP_sync_req() to the Host. This timing ensures that both the CableCARD device and Host have a minimum of eight seconds to complete key calculation. The CableCARD CP_sync_req() message indicates that the CableCARD device has completed calculation of CPKey. The Host SHALL issue the CP_sync_cnf() message when it has received the CP_sync_req() message and has completed calculation of the Host Copy Protection Key.

In a single CPKey operation, when the CableCARD device issues the CP_sync_req() message, the CableCARD device SHALL turn off scrambling of the MPEG content output and set the MPEG transport_scrambling_control_field to 00. The Host receives cleartext packets and will recognize these packets as unencrypted according to MPEG rules. When the Key Refresh Timer reaches ten seconds, the CableCARD device SHALL immediately return to scrambling of MPEG packets. If the key refresh has not completed when the Key Refresh Timer reaches ten seconds, the CableCARD device SHALL disable CA decryption of all copy protected content (including other copy protected services on the interface) until a full CP_Key refresh is completed.

In dual CPKey operation all protected content SHALL be scrambled throughout the CPKey generation and change process. No one-second clear period SHALL occur. This is the primary objective of the dual key capability.

Figure 5 illustrates the CableCARD flow during a Key Refresh cycle. Figure 6 illustrates the Host flow during a Key Refresh cycle.

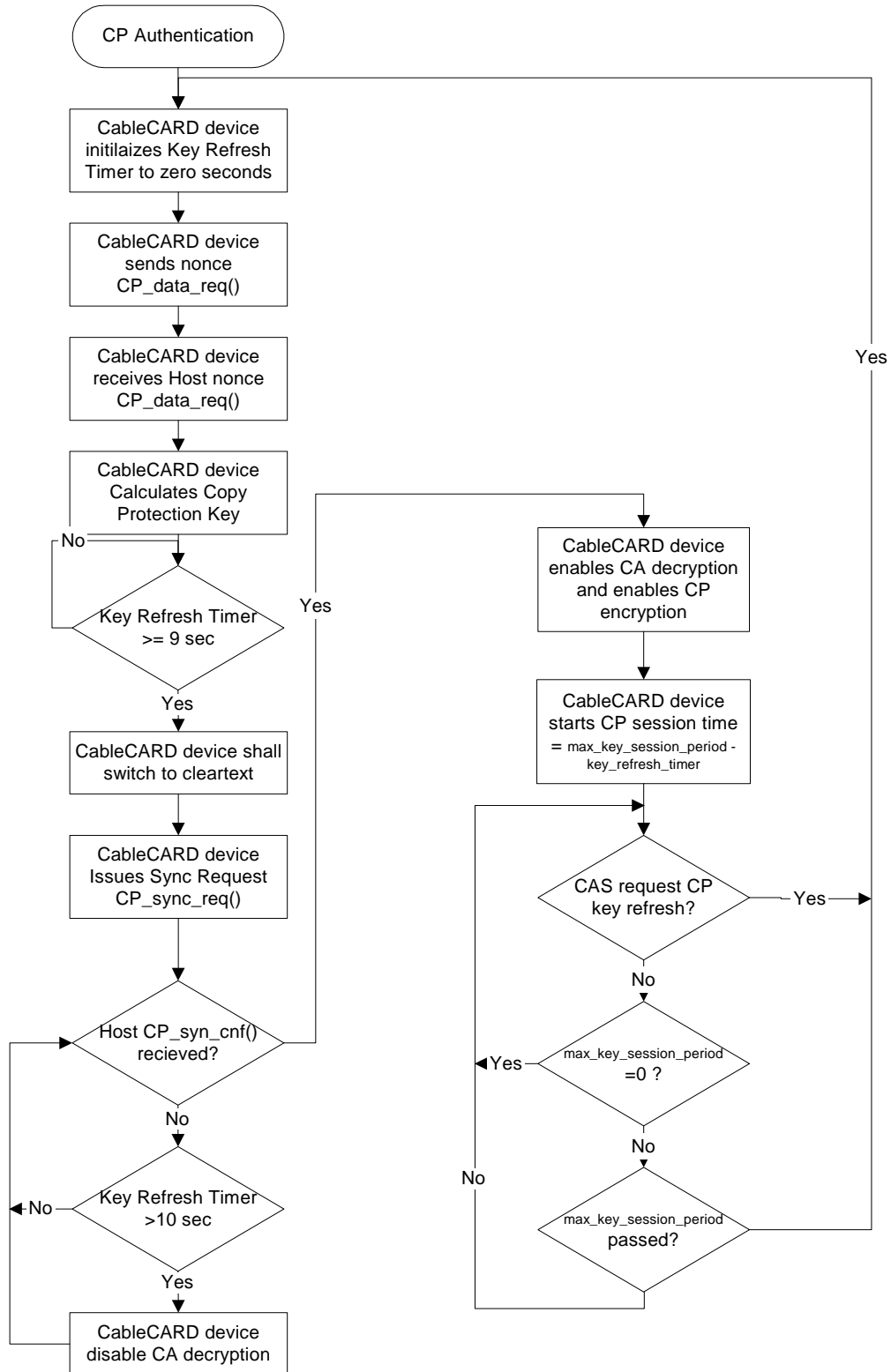


Figure 5 - CableCARD CPKey Refresh Session Flow Chart

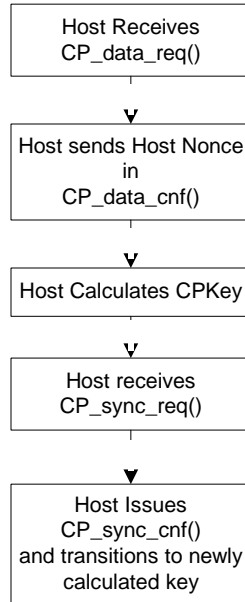


Figure 6 - Host CPKey Refresh Flow Chart

6.3.3 CA Initiated CPKey Refresh

The CableCARD device SHALL be capable of initiating a key refresh at the command of the CA System. This key refresh command SHALL occur regardless of any other conditions, excepting that a key refresh is occurring at that time.

7 COPY CONTROL INFORMATION (CCI)

The content provider and the content distributor determine the CCI value for each program. The CA System delivers the CCI for each program securely to the CableCARD device. The CableCARD device passes CCI to the Host through a secure authentication protocol, run once for each copy protected program being decrypted. The Host uses the CCI to control copy creation, analog output copy control encoding, and to set copy control parameters on Host outputs for a particular program. CCI is time sensitive. The Host SHALL retain the temporal association of CCI with program content to within 2 seconds. The Host SHALL control output of content according to the originally associated CCI value.

7.1 CCI Definition

CCI is a single byte, 8-bit, field conveyed from CableCARD device to Host. Five of the eight bits are defined. The remaining three are reserved. The reserved bits SHALL be set to zero by the CableCARD device as shown in Table 4. The Host SHALL use the reserved bit values received from the CableCARD device only for execution of the Authenticated Tunnel Protocol described below. The Host SHALL ignore the reserved bit values thereafter.

Table 4 - CCI Bit Assignments

CCI Bits #	7	6	5	4	3	2	1	0
CableCARD device sets to	0	0	0	CIT	APS1	APS0	EMI1	EMI0
Host interprets as	rsvd	rsvd	rsvd	CIT	APS1	APS0	EMI1	EMI0

7.1.1 EMI - Digital Copy Control Bits

The two lsb of the CCI byte are the EMI bits. They SHALL control copy permissions for digital copies. The EMI bits SHALL be supplied to any Host digital output ports for control of copies made from those outputs. The EMI bits are defined in Table 5.

Table 5 - EMI Values and Content Type

EMI Value	Digital Copy Permission	Content Type
00	Copying not restricted	Not "High Value"
01	No further copying is permitted	High Value
10	One generation copy is permitted	High Value
11	Copying is prohibited	High Value

7.1.2 APS - Analog Protection System

Bits 3 and 2 of CCI as shown in Table 4 are the APS bits 1 and 0 respectively. The Host SHALL use the APS bits to control copy protection encoding of analog composite outputs as described in Table 6.

Table 6 - APS Value Definitions

APS	Description
00	Copy Protection Encoding Off
01	AGC Process On, Split Burst Off
10	AGC Process On, 2 Line Split Burst On
11	AGC Process On, 4 Line Split Burst On

7.1.3 Host Set CCI Values

During periods when the CCI for a program is not yet known to the Host, e.g., immediately after a channel change, the Host shall set the Default CCI Value as defined in Table 7, and shall start a 10-second timer. If the Host has not yet successfully completed the CCI delivery protocol when the timer reaches ten (10) seconds the Host shall change CCI to the Error CCI Value.

Table 7 - Host Default and Error CCI Value

CCI Bits #	7	6	5	4	3	2	1	0
CCI Bit Assignment	rsvd	rsvd	rsvd	CIT	APS1	APS0	EMI1	EMI0
Default CCI Value	0	0	0	0	0	0	1	1
Error CCI Value	0	0	0	1	0	0	1	1

7.1.4 CIT – Constrained Image Trigger

The Host shall control Image Constraint of high definition analog component outputs according to the value of CIT as shown in Table 8.

Table 8 - CIT Values and Application

CIT Value	Image Constraint Application
0	No Image Constraint asserted
1	Image Constraint required

7.2 Associating CCI with a Service

The CA System SHALL securely associate unique CCI with a specific MPEG Program and LTSID. The MPEG Program Number zero SHALL NOT be used for content covered by this specification.

7.3 Conveying CCI from Headend to CableCARD Device

The CA System will provide a private secure delivery means (e.g., an ECM) to transfer CCI from the headend to the CableCARD device. This delivery means SHALL preserve the association between CCI and the specific MPEG Program and LTSID.

7.4 Conveying CCI from CableCARD Device to Host

Delivery of CCI from CableCARD device to Host is authenticated via the exchange of messages as shown in Figure 7. The messages are based on a SHA-1 function performed on the CCI, CPKey, MPEG program number, and LTSID.

The sequence is repeated for each program currently being decrypted by the CableCARD device. Each CCI exchange should be completed before the next one begins.

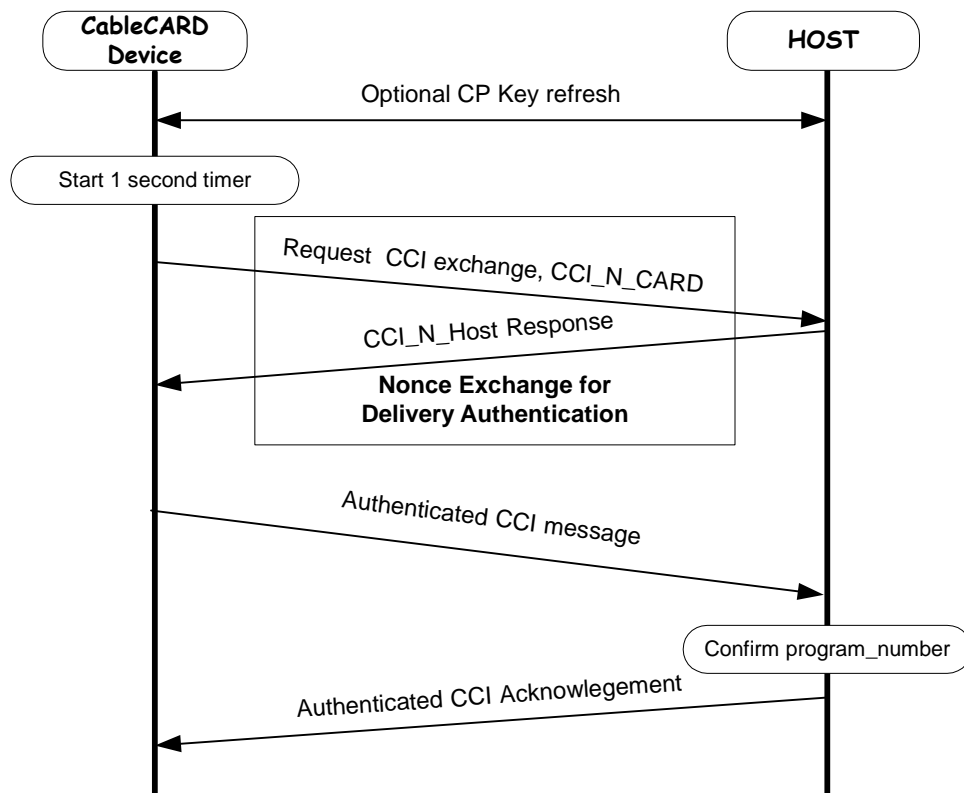


Figure 7 - CCI Delivery Sequence

7.4.1 CCI Delivery Instances

The CableCARD device SHALL send CCI to the Host only after the CableCARD device and Host have successfully completed Authentication and ID Validation, and negotiated a shared CPKey. The CableCARD device SHALL initiate CCI transfer for the effected program to the Host immediately after:

- the Host requests a new MPEG program, or
- the MPEG program number changes on a tuned 'channel', or
- the CCI bits change for any program that the CableCARD device is descrambling, or
- the MPEG packet ID values change for any program that the CableCARD device is descrambling.

7.4.2 CCI Authenticated Tunnel Protocol

The "authenticated tunnel protocol" is a means of verifying delivery of valid CCI from CableCARD device to Host. The CableCARD device and Host SHALL jointly execute the steps below once for each transfer of CCI. There SHALL be a transfer of CCI for every MPEG service requested by the Host device. Any failure of the steps

described below SHALL result in a failed CCI delivery. Until a successful CCI delivery has been completed, the CableCARD device SHALL disable CA-descrambling of protected content and the Host SHALL default to maximal protection of all CP-scrambled content.

For every unique combination of LTSID and program_number:

- Step 1. The CableCARD device SHALL generate a new random integer CCI_N_CARD and starts a 1-second time-out.
- Step 2. The CableCARD device SHALL send CCI_N_CARD, program_number, LTSID, and a request for CCI_N_Host.
- Step 3. The Host SHALL generate a new random integer CCI_N_Host.
- Step 4. The Host SHALL reply with CCI_N_Host and program_number and LTSID (received in step 2 above).
- Step 5. The CableCARD device SHALL calculate two values: CCI_auth to authenticate CCI delivery, and CCI_ack to authenticate Host acknowledgment of receipt, as:
$$\text{CCI_auth} = \text{SHA-1}(\text{CCI} | \text{CPKey} | \text{CCI_N_CARD} | \text{CCI_N_Host} | \text{program_number} | \text{LTSID})$$
$$\text{CCI_ack} = \text{SHA-1}(\text{CPKey} | \text{CCI} | \text{CCI_N_CARD} | \text{CCI_N_Host} | \text{program_number} | \text{LTSID})$$
- Step 6. The CableCARD device SHALL transmit CCI_auth, CCI, program_number and LTSID to the Host.
- Step 7. The Host SHALL calculate CCI_auth using the received CCI value and compare it with the CCI_auth value received from the CableCARD. If they do not match the Host SHALL set the Error CCI Value and return a zero value CCI_ack to the CableCARD to indicate the failure.
- Step 8. The Host SHALL begin controlling its outputs based on valid CCI within one second.
- Step 9. The Host calculates CCI_ack and sends it to the CableCARD.
- Step 10. The CableCARD device compares the received CCI_ack with the value calculated in step 5 above. If they match, the CableCARD device SHALL begin delivery of the associated program as indicated by the CCI. If they do not match, the CableCARD device SHALL NOT CA descramble the associated program until its CCI delivery is successfully acknowledged.

If the steps above are not completed before the one-second time-out expires the CableCARD device SHALL disable CA-descrambling of copy protected content for that LTSID/program_number pair until the CCI delivery protocol completes successfully.

8 TRANSPORT ENCRYPTION FROM CABLECARD DEVICE TO HOST

All copy-protected content is protected against unauthorized access in the form of MPEG Transport Streams as they pass from CableCARD device to Host by DES-based data encryption.

8.1 MPEG Scrambling

The CableCARD device processes content flowing from input to output in one or four modes:

- Clear: no change of CA-unscrambled and EMI=00 content which remains 'in-the-clear'
- CA-only: descrambles CA-scrambled content marked EMI=00 for output 'in-the-clear'
- Rescramble: CA-descrambles and CP-scrambles content marked EMI>0
- Pass-through: no change of CA-scrambled content (leaving it useless to the Host)

The CP-scrambling mode is set as shown in Table 10.

8.1.1 Scrambling Rules

- DES ECB SHALL be used to scramble copy protected MPEG programs in the CableCARD device and to descramble them in the Host. Any residual blocks less than 64 bits in size SHALL be left in the clear.
- The CableCARD device SHALL encrypt only the payload portion of MPEG transport stream packets. M-CARD packet preheaders [11], MPEG packet headers and adaptation headers SHALL NOT be encrypted.
- The MPEG scrambling bits output from the CableCARD device SHALL be set as described in Table 9.
- CA-scrambled but unauthorized services and CA-scrambled and authorized but unselected services SHALL pass through CableCARD device unaltered, and are therefore useless to the Host.
- CP-scrambling SHALL only be applied to selected MPEG programs for which EMI is non-zero
- The CableCARD device SHALL CP-scramble only authorized and selected programs. The CableCARD device SHALL immediately switch from rescramble mode to pass-through mode when the active program is deauthorized by the CA System.
- No data SHALL be double scrambled with both CA and CP-scrambling.

MPEG content which is delivered with EMI equal to zero, no copying restrictions specified, for example free access off-air broadcast content, SHALL be delivered CP-unscrambled from the CableCARD device to the Host. Such content may or may not be CA-scrambled during delivery from the headend to the CableCARD device.

8.1.2 Transport Scrambling Control Field

The transport_scrambling_control field of the MPEG transport packet provides control information for key changes. The transport_scrambling_control field bit values for single CPKey and dual CPKey modes SHALL be defined as in Table 9.

Table 9 - MPEG Transport_scrambling_control Values

Bit Values	For Single CPKey Mode	For Optional Dual CPKey Mode
00	No scrambling of TS packet payload	No scrambling of TS packet payload
01	Reserved	Reserved
10	Reserved	TS packet scrambled using EVEN key
11	Transport packet scrambled	TS packet scrambled using ODD key

8.2 Transport Processing

MPEG packet scrambling parity may take on the values 0 or 1 without limitation on CP-scrambled output from the CableCARD device. The CableCARD device SHALL set the MPEG packet scrambling bits as defined in section Table 9.

CP-scrambling mode changes (i.e., transitions from "CP-scrambling ON" to "CP-scrambling OFF/Clear") SHALL be handled so as to minimize any period of time where copy protected content is sent unscrambled from CableCARD device to Host. If the status of CA-scrambling (e.g. due to a change in CableCARD device entitlement or a change in the scrambling mode of the MPEG stream input to the CableCARD device), then the CableCARD device SHALL alter the mode of its input CA-descrambling prior to altering its mode of output CP-scrambling.

The transport processing is applied to all transport streams across the interface.

8.3 Timing of Scrambling Mode Transitions

CP-scrambling mode changes from "CP-scrambling OFF" to "CP-scrambling ON" SHALL be accomplished quickly and in no case more than 1.5 seconds after the event that causes the mode change, e.g., an EMI change from 0 to non-zero. All MPEG packets of the relevant program SHALL be CP-scrambling as soon as possible following a EMI change from 0 to a protected value of 1, 2, or 3. A change from EMI >0 to EMI=00 SHALL result in scrambling going inactive within 1.5 seconds.

CA-decryption may be effected by receipt of encryption management or control messages or by changes in the CA-encryption mode of the content data. The CableCARD device SHALL continue to comply with all CP-scrambling requirements while responding to any such messages or mode changes.

8.4 CP-Scrambling as a Function of CA-Scrambling and EMI Value

The CableCARD device SHALL apply copy protection scrambling of content flowing to the Host as shown in Table 10.

Table 10 - CP-Scrambling Based on CA-Scrambled State and EMI Value

CA Scrambling State	EMI Value	CableCARD Scrambles Output	Comments
Unscrambled	00	No	
Unscrambled	01, 10, or 11	No	Undesired*
Scrambled	00	No	
Scrambled	01, 10, or 11	Yes	

* Cable operators SHOULD CA-Scramble all programs with non-zero EMI. Only CA-Scrambled programs will be protected from unauthorized copying.

8.5 M-CARD Mode DES_Keys

The CableCARD device SHALL encrypt each copy protected program m , where

$$m = \text{LTSID} | \text{program_number}$$

using an individual $\text{DES_Key}[m]$. $\text{DES_Key}[m]$ is defined as follows:

$$\text{DES_Key}[m] = \text{CPKey} \oplus m$$

where m has been padded with 32 left-most zero bits to match the length of CPKey.

Each instance of $\text{DES_Key}[m]$ SHALL be available for and in use by the DES-ECB cryptographic element for encryption in the CableCARD device and decryption in the Host immediately upon transmission of CP_sync_cnf following CPKey generation or refresh.

Any changes in program_number or LTSID SHALL be immediately reflected in the value of DES_Key used to encrypt and decrypt the affected MPEG stream packets.

8.5.1 Channel Change

When a channel change occurs, the Host SHALL set CCI value to the Default CCI Value defined in Section 7.1.3. The Host SHALL immediately begin using the value of CCI when it is received from the CableCARD device. Channel change SHALL NOT cause a key refresh to occur.

8.5.2 ODD-Even Key Synchronization Mode (Informative)

Optionally, a CableCARD device or Host may implement key refresh using a system of EVEN and ODD CPKeys. In that case the above described system of going into the clear for one second is not needed, and is instead replaced with a one second period before a new key is written into one of the EVEN or ODD key registers. Such a two key system is not fully specified at this time, but will be fully specified in a future release.

The presence of two CPKey registers and selection logic for EVEN and ODD CPKeys based on MPEG-TS header `transport_scrambling_control` bits, is optional in both CableCARD device and Host. If implemented the `transport_scrambling_control` bits will select the EVEN or ODD key.

It is highly recommended that CableCARD device and Host manufacturers build silicon that is capable of holding both an EVEN and an ODD CPKey, and is capable of properly selecting the correct EVEN or ODD key based on the `transport_scrambling_control` bits. It is further recommended that all CableCARD device and Host devices include a firmware download means to fully support ODD/EVEN CPKey refresh when it is fully defined, e.g., for APDUs, protocols flows, etc.

9 CABLECARD-HOST MESSAGING PROTOCOLS

9.1 Message Protocol Overview

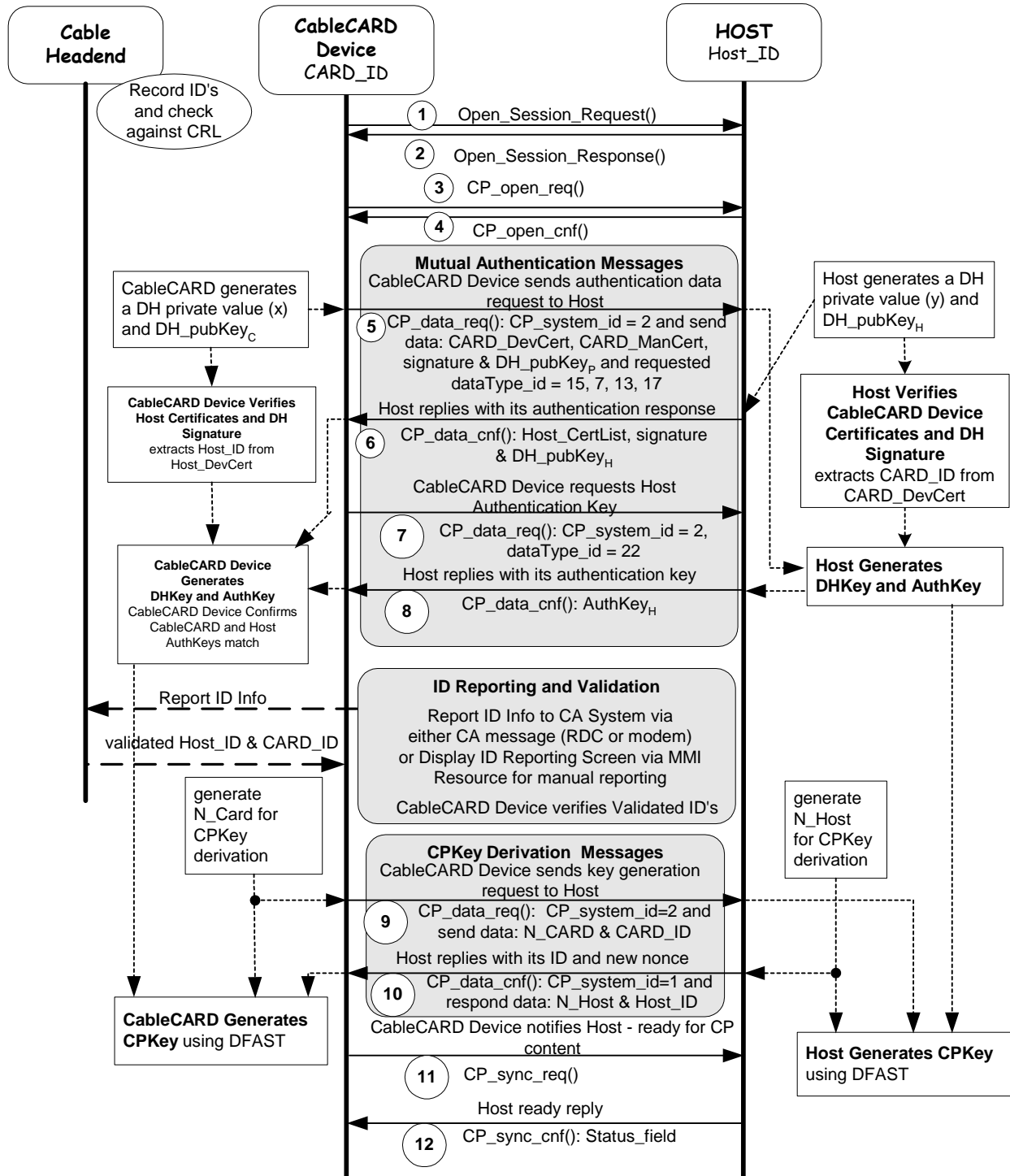


Figure 8 - CableCARD-Host Message Protocol Flow

Table 11 gives an overview of the CP System message flow starting with initial CableCARD-Host binding. After the CableCARD device and Host binding is authenticated or validated only parts of the illustrated sequence will be repeated, as described in Sections 3 and 4, for example, following power-ups or for CPKey refresh.

Table 11 - Message Reference Sections

#	Message Name	Protocol Layer / Tag Value (hex)	Reference Section	Purpose
1	Open_Session_Request	SPDU / 91	Section 9.3.1	Open CP session
2	Open_Session_Response	SPDU / 92	Section 9.3.2	
3	CP_open_req	APDU / 9F9000	Section 9.3.3	Evaluate Host
4	CP_open_cnf	APDU / 9F9001		
5	CP_data_req	APDU / 9F9002	Section 9.4.1	CableCARD device & Host authentication data
6	CP_data_cnf	APDU / 9F9003		
7	CP_data_req	APDU / 9F9002	Section 9.4.2	Authentication Key verification
8	CP_data_cnf	APDU / 9F9003		
9	CP_data_req	APDU / 9F9002	Section 9.5	CPKey derivation
10	CP_data_cnf	APDU / 9F9003		
11	CP_sync_req	APDU / 9F9004	Section 9.6	CableCARD device & Host Synchronization
12	CP_sync_cnf	APDU / 9F9005		

9.2 CABLECARD-Host Message Parameters

Table 12 - CP_system_id Values

CP_system_id	ID Value (Binary)
No compatible CP system supported	XXX0 0000
System 1	XXX0 0001
System 2 (The CableCARD-CP System)	XXX0 0010
Systems 3 to 30	XXX0 0011 to XXX1 1110
System 31	XXX1 1111
Message is Encrypted	1XXX XXXX
Message is Not Encrypted	0XXX XXXX

Table 13 - CP System Message Parameters

Datatype_id	id value	Length (Bytes)
Reserved (Manufacturer ID not used)	1	
Reserved	2	
Reserved	3	
Reserved	4	
Host_ID	5	5
CARD_ID	6	8
Host_ManCert (Host Manufacturer XCA Certificate)	7	2048
CARD_ManCert (CableCARD device Manufacturer XCA Certificate)	8	2048
Reserved	9	
Reserved	10	
N_Host (Host's challenge to CableCARD device)	11	8
N_CARD (CableCARD device's challenge to Host)	12	8
DH_pubKey _H (Host DH public key)	13	128
DH_pubKey _P (CableCARD DH public key)	14	128
Host_DevCert (Host Device Certificate Data)	15	2048
CARD_DevCert (CableCARD Device Certificate Data)	16	2048
SIGN _H (the signature of Host DH public key)	17	128
SIGN _P (the signature of CableCARD DH public key)	18	128
CCI_N_Host	19	8
Reserved	20	
Reserved	21	
AuthKey _H (Host Authentication Key before verification)	22	20
Reserved	23	
CCI_N_CARD	24	8
CCI_data	25	1
Program_Number	26	2
CCI_auth	27	20
CCI_ack	28	20
LTSID (Local Transport Stream ID), assigned by the Host	29	1

9.3 Opening a CP Session

The CableCARD device SHALL request a session to be opened to a resource on its transport connection. Since Host provides resources it replies directly with a session number in its open session response.

Two objects defined at Session Protocol Data Unit (SPDU) layer, `open_session_request()` and `open_session_response()` are used here. Detailed SPDU data structure and other SPDU objects are defined in [11].

Table 14 - Copy Protection Open Session Information

SPDU Tag / Object	Tag Value (Hex)	Action	Direction
Open_Session_Request()	91	The CableCARD device requests a session of the Copy Protection resource to be opened.	CableCARD → Host
Open_Session_Response()	92	The Host responds with a session status. If opened, a session number is assigned. The session number SHALL then be used for all subsequent exchanges of messages (APDUs) between CableCARD device and Host.	CableCARD ← Host

9.3.1 Open_Session_Request() Syntax

Opening a copy protection request uses an object defined in the Session Layer protocol. The CableCARD device issues this SPDU object to request opening a copy protection session between the CableCARD device and Host.

Table 15 - Open_Session_Request() Message Syntax

Message Syntax	bits	bytes	Description
open_session_request () {			
Open_session_request_tag	8	1	Has the value of 91 (hex)
Length_field()	8	1	length_field () is defined in [2] section 7. Since resource_identifier() followed only has 32 bits, length_field() SHALL have the following values set: size_indicator = 0, length_value = 4
Resource_identifier()	32	4	Resource_identifier () is defined in [2] section 8.2.2.
			Resource_identifier() {
			resource_id_type
			if (resource_id_type != 3) {
			resource_class
			resource_type
			resource_version
			}
			else {
			private_resource_definer
			private_resource_identity
			}
}			}

As specified in section 7.2.6.1 of [2]: the resource_identifier SHALL match in both class and type of resource that the Host has in its list of available resources. Copy protection resource coding is listed in the OpenCable Multi-Stream CableCARD Interface Specification [11].

If the version field of the supplied resource identifier is zero, then the Host SHALL use the current version in its list. If the version number in the request is less than or equal to the current version number in the Host's list then the current version is used. If the requested version number is higher than the version in the Host's list, the Host SHALL refuse the request with the appropriate return code as defined in [1].

9.3.2 Open_Session_Response() Syntax

The Host issues this object to the CableCARD device to allocate a session number or to tell the CableCARD device that its request could not be met.

Table 16 - Open_Session_Response() Message Syntax

Message Syntax	bits	bytes	Description
open_session_response () {			
Open_session_response_tag	8	1	Has the value of 92 (hex)
Length_field()	8	1	length_field () is defined in [2], section 7. Since session_status (1 byte), resource_identifier() (4 bytes), and session_nb (2 bytes) are followed, length_field() SHALL have the following values set: size_indicator = 0, length_value = 7
Session_status	8	1	Session status values listed in [2].
Resource_identifier()	32	4	Resource_identifier () is defined Table 14.
Session_nb	16	2	The Host allocates session number for the requested session. Value 0 is reserved. The session_nb SHALL be used for all subsequent exchanges of APDUs between the CableCARD device and Host until session is closed. When the requested session could not be opened (session_status != 0), the session_nb has no meaning.
}			

9.3.3 Host Capability Evaluation

The CableCARD device SHALL check the Host's ability to support the CP System when the CableCARD device is powered-on and before starting the Key Exchange process.

The CableCARD device SHALL use CP_open_req() and the Host SHALL use CP_open_cnf().

Table 17 - Host CP Support Capability Evaluation Messages

APDU Tag / Object	Tag Value (Hex)	Action	Direction
CP_open_req()	9F9000	CableCARD device queries which copy protection system is supported by Host.	CableCARD → Host
CP_open_cnf()	9F9001	Host replies to CableCARD device	CableCARD ← Host

9.3.3.1 CP_open_req() Syntax

This APDU object is issued by the CableCARD device to query the Host's ability to support various copy protection systems.

Table 18 - CP_open_req() Message Syntax

Message Syntax	bits	bytes	Description
CP_open_req () { CP_open_req_tag Length_field() }	24 8	3 1	Has the value of 9F9000 (hex) length_field () is defined in [2], section 7. Since there is no other field followed, length_field() SHALL have the following values set: size_indicator = 0, length_value = 0

9.3.3.2 CP_open_cnf() Syntax

This object is issued by the Host to the CableCARD device.

Table 19 - CP_open_cnf() Message Syntax

Message Syntax	bits	bytes	Description
CP_open_cnf () { CP_open_cnf_tag Length_field() CP_system_id_bitmask }	24 8	3 1	Has the value of 9F9001 (hex) length_field () is defined in [2], section 7. The length_field() SHALL have the following values set: size_indicator = 0, length_value = 4 Values are list in Table 20.

Table 20 - CP_system_id_bitmask Values

CP_system_id_bitmask	Bit Number	Description
System 1	0	reserved
System 2	1	CableCARD CP System
System 3	2	reserved
System 4	3	reserved
System 5	4	reserved

For an example, if bit number 0, 1 and 3 are set to 1, it means that Host has the capability of supporting System 1, System 2, and System 4.

9.4 CableCARD-Host Mutual Authentication Message Protocol

9.4.1 Mutual Authentication Messages

Two objects, CP_data_req() and CP_data_cnf(), as defined at Application Protocol Data Unit (APDU) layer are used to exchange the authentication messages.

Table 21 - Host Authentication Messages

APDU Tag / Object	Tag Value (Hex)	Action	Direction
CP_data_req()	9F9002	CableCARD device sends its authentication data to Host	CableCARD → Host
CP_data_cnf()	9F9003	Host replies to CableCARD device with its authentication data	CableCARD ← Host

9.4.1.1 CP_data_req() Syntax in Host Authentication Request Message

The CableCARD device issues this APDU to send its authentication data to the Host. CARD_DevCert, CARD_ManCert, DH_pubKey_c and SIGN_p are included in this message.

Table 22 - CP_data_req in the Host Authentication Request Message

Message Syntax	bits	bytes	Description
CP_data_req () {			
CP_data_req_tag	24	3	Has the value of 9F9002 (hex)
Length_field()	24	3	Defined by and with values set to: Size_indicator = 1 (1 bit, bslbf); Length_field_size = 2 (7 bits, uimbsf); Length_value_byte[0] = 17 (8 bits, bslbf); (most significant byte) Length_value_byte[1] = 19 (8 bits, bslbf); (least significant byte) (message size following above is 4371 bytes
CP_system_id	8	1	CP_system_id = 2 (CableCARD CPS)
Send_datatype_nbr	8	1	Send_datatype_nbr SHALL have the value of 4
For(i=0; i<Send_datatype_nbr; i++) {	(96)	(12)	
Datatype_ID	8	1	When i = 0, Datatype_ID has the value of 16 (CARD_DevCert);
	8	1	When i = 1, Datatype_ID has the value of 8 (CARD_ManCert);
	8	1	When i = 21, Datatype_ID has the value of 14 (DH_pubKey _P);
	8	1	When i = 3, Datatype_ID has the value of 18 (SIGN _P);
Datatype_length	16	2	When i = 0, Datatype_length has the value of 2048;
	16	2	When i = 1, Datatype_length has the value of 2048;
	16	2	When i = 2, Datatype_length has the value of 128;
	16	2	When i = 3, Datatype_length has the value of 128;
For (j=0; j<Datatype_length; j++) {		(4352)	
Data_type	16384	2048	When i = 0, Data_type = CARD_DevCert;
	16384	2048	When i = 1, Data_type = CARD_ManCert;
	1024	128	When i = 2, Data_type = DH_pubKey _P ;
	1024	128	When i = 3, Data_type = SIGN _P ;
}			
}			
Request_datatype_nbr	8	1	Request_datatype_nbr SHALL have the value of 4.
For(i=0; i<Request_datatype_nbr; i++) {	(32)	(4)	
Datatype_ID	8	1	When i = 0, Datatype_ID has the value of 15 (Host_DevCert)
	8	1	When i = 1, Datatype_ID has the value of 7 (Host_ManCert)
	8	1	When i = 2, Datatype_ID has the value of 13 (DH_pubKey _H).
	8	1	When i = 3, Datatype_ID has the value of 17 (SIGN _H).
}			
}			

9.4.1.2 CP_data_cnf() Syntax in Host Authentication Response Message

The Host issues this APDU respond to the CableCARD device with Host_DevCert, Host_ManCert, DH_pubKey_H and SIGN_H.

Table 23 - CP_data_cnf in the Host Authentication Response Message

Message Syntax	bits	bytes	Description
CP_data_cnf () {			
CP_data_cnf_tag	24	3	Has the value of 9F9003 (hex)
Length_field()	24	3	Defined by and with values set to: Size_indicator = 1 (1 bit, bslbf); Length_field_size = 2 (7 bits, uimsbf) Length_value_byte[0] = 17 (8 bits, bslbf) (most significant byte) Length_value_byte[1] = 14 (8 bits, bslbf) (least significant byte) (message size = 4366 bytes after length bytes)
CP_system_id	8	1	CP_system_id = 2 (CableCARD CPS)
Send_datatype_nbr	8	1	Send_datatype_nbr SHALL have the value of 4
For(i=0; i<Send_datatype_nbr; i++) {	(96)	(12)	
Datatype_ID	8	1	When i = 0, Datatype_ID= 15 (Host_DevCert);
	8	1	When i = 1, Datatype_ID has the value 7 (Host_ManCert);
	8	1	When i = 2, Datatype_ID has the value 13 (DH_pubKey _H);
	8	1	When i = 3, Datatype_ID has the value 17 (SIGN _H);
Datatype_length	16	2	When i = 0, Datatype_length has the value of 2048
	16	2	When i = 1, Datatype_length has the value of 2048
	16	2	When i = 2, Datatype_length has the value of 128
	16	2	When i = 3, Datatype_length has the value of 128
For (j=0; j<Datatype_length; j++) {	34816	(4352)	
Data_type	16384	2048	When i = 0, Data_type = Host_DevCert;
	16384	2048	When i = 1, Data_type = Host_ManCert;
	1024	128	When i = 2, Data_type = DH_pubKey _H ;
	1024	128	When i = 3, Data_type = SIGN _H
}			
}			
}			

9.4.2 Host AuthKey Verification Request Messages

Two objects, CP_data_req() and CP_data_cnf(), as defined at Application Protocol Data Unit (APDU) layer, are used for the CableCARD device to obtain the authentication key from the Host.

Table 24 - Host Authentication Key Verification Messages

APDU Tag / Object	Tag Value (Hex)	Action	Direction
CP_data_req()	9F9002	CableCARD device requests Host authentication key	CableCARD → Host
CP_data_cnf()	9F9003	Host replies to CableCARD device with AuthKey _H	CableCARD ← Host

9.4.2.1 CP_data_req() In the AuthKey Verification Request Message

The CableCARD device issues this APDU to request the Host's authentication key.

Table 25 - CP_data_req in the Authentication Key Verification Request Message

Message Syntax	bits	bytes	Description
CP_data_req () { CP_data_req_tag length_field() CP_system_id Send_datatype_nbr Request_datatype_nbr For(i=0; i<Request_datatype_nbr; i++) { Datatype_ID } }	24 8 8 8 8 (8) 8	3 1 1 1 1 (1) 1	Has the value of 9F9002 (hex) Length_field () is defined [2], section 7. The length_field() in this message SHALL have the following values set: size_indicator = 0, length_value = 4 CP_system_id = 2 Send_datatype_nbr SHALL have the value of 0. Request_datatype_nbr SHALL have the value of 1. Datatype_ID has the value of 22 (AuthKey _H , see Table 13).

9.4.2.2 CP_data_cnf() In the Authentication Key Verification Response Message

The Host issues this APDU to send its authentication key (AuthKey_H) to the CableCARD device.

Table 26 - CP_data_cnf in the Authentication Key Verification Response Message

Message Syntax	bits	bytes	Description
CP_data_cnf () {			
CP_data_cnf_tag	24	3	Has the value of 9F9003 (hex)
length_field()	8	1	length_field () is defined in [2], section 7. The length_field() in this message SHALL have the following values set: size_indicator = 0, length_value = 25
CP_system_id	8	1	CP_system_id = 2
Send_datatype_nbr	8	1	Send_datatype_nbr SHALL have the value of 1.
For(i=0; i<Send_datatype_nbr;	(16)	(2)	
i++) {			
Datatype_ID	8	1	Datatype_ID has the value of 22 (AuthKey _H , see Table 13)
Datatype_length	16	2	Datatype_length has the value of 20 (see Table 13)
For (j=0; j<Datatype_length;			
j++)			
{			
Data_type	160	20	Data_type = AuthKey _H (see Table 13)
}			
}			
}			

9.5 Copy Protection Key Generation

The CableCARD device sends a CPKey generate request to the Host with the CARD_ID and N_CARD. Upon receipt of this request, the Host SHALL generate N_Host and send it to the CableCARD device along with the Host_ID. Two APDUs are used here: CP_data_req() and CP_data_cnf()

Table 27 - CP_data in the CPKey Generation Messages

APDU Tag / Object	Tag Value (Hex)	Action	Direction
CP_data_req()	9F9002	CableCARD device requests the generation of a new transmission key. This message contains CARD_ID and N_CARD (CP_system_id = 2, send datatype_id = 6, 12, and receive datatype_id = 5, 11).	CableCARD → Host
CP_data_cnf()	9F9003	Host replies to the CableCARD device with Host_ID and N_Host.	CableCARD ← Host

Table 28 - CP_data_req() Message Syntax In the Key Generation Messages

Message Syntax	bits	bytes	Description
CP_data_req () {			
CP_data_req_tag	24	3	Has the value of 9F9002 (hex)
Length_field()	8	1	length_field () is defined in [2], section 7. size_indicator = 0, length_value = 27
CP_system_id	8	1	Values are listed in Table 12: CP_system_id = 2
Send_datatype_nbr	8	1	Send_datatype_nbr SHALL have the value of 2.
For(i=0; i<Send_datatype_nbr; i++) {	(48)	(2*3)	
Datatype_ID	8	1	When i = 0, Datatype_id = 6 (CARD_ID)
	8	1	When i = 1, Datatype_id = 12 (N_CARD)
Datatype_length	16	2	When i = 0, Datatype_length = 0x0008
	16	2	When i = 1, Datatype_length = 0x0008
For (j=0; j<Datatype_length; j++)	(128)	(16)	
{			
Data_type	64	8	When i = 0, Data_type = CARD_ID
	64	8	When i = 1, Data_type = N_CARD;
}			
Request_datatype_nbr	8	1	Request_datatype_nbr SHALL have the value of 2.
For(i=0; i<Request_datatype_nbr; i++)	(16)	(2*1)	
{			
Datatype_id	8	1	When i = 0, Datatype_id = 5 (Host_ID)
	8	1	When i = 1, Datatype_id = 11 (N_Host)
}			
}			

Table 29 - CP_data_cnf() Message Syntax In the Key Generation Messages

Message Syntax	bits	bytes	Description
CP_data_cnf () { CP_data_cnf_tag Length_field() CP_system_id Send_datatype_nbr For(i=0; I<Send_datatype_nbr; i++) { Datatype_id Datatype_length (j=0; j<Datatype_length; j++) { Data_type } } }	24 8 8 8 (48) 8 8 16 16 (104) 40 64	3 1 1 1 (2*3) 1 1 2 2 (13) 5 8	Has the value of 9F9003 (hex) length_field () is defined in [2], section 7. The length_field() SHALL have the following values set: size_indicator = 0, length_value = 21 Values are listed in Table 12 Send_datatype_nbr SHALL have the value of 2. When i = 0, Datatype_id= 5 (Host_ID) When i = 1, Datatype_id=11 (N_Host) When i = 0, Datatype_length = 0x0005 When i = 1, Datatype_length = 0x0008 When i = 0, Data_type = Host_ID When i = 1, Data_type = N_Host

9.6 Host and CableCARD Device Synchronization

The CableCARD device notifies the Host of its intention to start CP encryption of Protected MPEG programs with CP_sync_req(). The Host replies when it is ready with CP_sync_cnf().

Table 30 - Host CableCARD Device and Synchronization Messages

APDU Tag / Object	Tag Value (Hex)	Action	Direction
CP_sync_req()	9F9004	The CableCARD device notifies the Host when it is ready to start to transmit the CP data.	CableCARD → Host
CP_sync_cnf()	9F9005	Host replies to the CableCARD device to confirm Host readiness.	CableCARD ← Host

Table 31 - CP_sync_req() Message Syntax

Message Syntax	Bits	bytes	Description
CP_sync_req () { CP_sync_req_tag Length_field() }	24 8	3 1	Has the value of 9F9004 (hex) length_field () is defined in [2], section 7. The length_field() SHALL have the following values set: size_indicator = 0, length_value = 0

Table 32 - CP_sync_cnf() Message Syntax

Message Syntax	bits	bytes	Description
CP_sync_cnf () {			
CP_sync_req_tag	24	3	Has the value of 9F9005 (hex)
Length_field()	8	1	length_field () is defined in [2], section 7. The length_field() SHALL have the following values set: size_indicator = 0, length_value = 1
Status_field	8	1	Values are listed in Table 33
}			

Status_field SHALL return the status of the CP_sync_req(). If the Host is ready to receive the incoming stream, then Status_field SHALL be set to 0x00. Otherwise, it SHALL be set as indicated in Table 33.

Table 33 - Status_field Value

Status_field	Value
OK	00
Error – No CP support	01
Error – Host Busy	02
Reserved	03-FF

9.7 CCI Simple Authenticated Tunnel Protocol (SATP) Messages

The simple authenticated tunnel protocol uses two pairs of request-confirm messages. The CableCARD device generates a nonce and sends it to the Host. The Host generates a nonce and sends it to the CableCARD device. The CableCARD device calculates a fingerprint using the CCI value, program number, LTSID, and each nonce, and sends it to the Host appended to the CCI value. Finally, the Host sends a reply message without a data payload.

Table 34 - CCI Simple Authentication Tunnel Protocol Messages

APDU Tag / Object	Tag Value (Hex)	Action	Direction
CP_data_req	9F9002	CableCARD device requests the generation of a new 8 byte random integer. The message contains the random nonce generated by the CableCARD device (CCI_N_CARD) and the program number (program_number), the same one found in the CA_pmt_req() message and the LTSID. (CP_system_id = 2, send datatype_id = 24, 26, 29 and request datatype_id = 19, 26, 29).	CableCARD → Host
CP_data_cnf	9F9003	Host replies to CableCARD device with the requested data types. The response contains the random nonce generated by the Host (CCI_N_Host), the program number (program_number), and the LTSID. (CP_system_id = 2, send datatype_id = 19, 26, 29).	CableCARD ← Host
CP_data_req	9F9002	CableCARD device sends the CCI payload (CCI_data), the program number (program_number), the calculated message digest (CCI_auth and the LTSID. (CP_system_id = 2, send datatype_id = 25, 26, 27, request datatype_id=26, 28, 29).	CableCARD → Host
CP_data_cnf	9F9003	Host replies to CableCARD device with CCI_ack. (CP_system_id = 2, send datatype_id=26, 28)	CableCARD ← Host

Table 35 - CP_data_req() Message Syntax in SATP Key Generation

Message Syntax	bits	bytes	Description
CP_data_req(){			
CP_data_req_tag	24	3	Has the value of 0x9F9002.
length_field()	8	1	Has the value of 0x1A. size_indicator = 0, length_value =26
CP_system_id	8	1	Has the value of 2. Values are listed in Table 12.
Send_datatype_nbr	8	1	Has the value of 3.
for(i=0; i<Send_datatype_nbr;			
i++)			
{			
Datatype_id	8	1	i = 0, Datatype_id has the value of 24 (CCI_N_CARD).
	8	1	i = 1, Datatype_id has the value of 26 (program_number).
	8	1	i = 2, Datatype_id has the value of 29 (LTSID).
Datatype_length	16	2	i = 0, Datatype_length has the value of 0x0008.
	16	2	i = 1, Datatype_length has the value of 0x0002.
	16	2	i = 2, Datatype_length has the value of 0x0001.
for (j=0; j<Datatype_length;			
j++)			
{			
Data_type	64	8	When i = 0, Data_type = CCI_N_CARD.
	16	2	When i = 1, Data_type = program_number.
	8	1	When i = 2, Data_type = LTSID.
}			
}			
Request_datatype_nbr	8	1	Has the value of 3.
for(i=0;			
i<Request_datatype_nbr; i++)			
{			
Datatype_id	8	1	When i=0, Datatype_id has the value 19 (CCI_N_Host).
	8	1	When i=1, Datatype_id has the value 26 (program_number).
	8	1	When i=2, Datatype_id has the value 29 (LTSID).
}			
}			
}			

Table 36 - CP_data_cnf() Message Syntax in CCI SATP Key Generation

Message Syntax	bits	bytes	Description
CP_data_cnf(){			
CP_data_cnf_tag	24	3	Has the value of 0x9F9003.
length_field()	8	1	Has the value of 0x16. size_indicator = 0, length_value = 22
CP_system_id	8	1	Has the value of 2. Values are listed in Table 12.
Send_datatype_nbr	8	1	Has the value of 3.
for(i=0; i<Send_datatype_nbr;			
i++)			
{			
Datatype_id	8	1	i = 0, Datatype_id has the value of 19 (CCI_N_Host).
	8	1	i = 1, Datatype_id has the value of 26 (program_number).
	8	1	i = 1, Datatype_id has the value of 29 (LTSID).
Datatype_length	16	2	i = 0, Datatype_length has the value of 0x0008.
	16	2	i = 1, Datatype_length has the value of 0x0002.
	16	2	i = 1, Datatype_length has the value of 0x0001.
for (j=0;			
j<Datatype_length; j++)			
{			
Data_type	64	8	When i = 0, Data_type = CCI_N_Host.
	16	2	When i = 1, Data_type = program_number.
	2	1	When i = 2, Data_type = LTSID.
}			
}			
}			
}			

Table 37 - CP_data_req() Message Syntax in CCI SATP Transmission

Message Syntax	bits	bytes	Description
CP_data_req(){			
CP_data_req_tag	24	3	Has the value of 0x9F9002.
length_field()	8	1	Has the value of 0x2A. size_indicator = 0, length_value = 42
CP_system_id	8	1	Has the value of 2. Values are listed in Table 12.
Send_datatype_nbr	8	1	Has the value of 4.
for(i=0; i<Send_datatype_nbr;			
i++)			
{			
Datatype_id	8	1	i = 0, Datatype_id has the value of 25 (CCI_data).
	8	1	i = 1, Datatype_id has the value of 26 (program_number).
	8	1	i = 2, Datatype_id has the value of 27 (CCI_auth).
	8	1	i = 3, Datatype_id has the value of 29 (LTSID).
Datatype_length	16	2	i = 0, Datatype_length has the value of 0x0001
	16	2	i = 1, Datatype_length has the value of 0x0002
	16	2	i = 2, Datatype_length has the value of 0x0014
	16	2	i = 3, Datatype_length has the value of 0x0001
for (j=0; j<Datatype_length;			
j++)			
{			
Data_type	8	1	When i = 0, Data_type = CCI_data.
	16	2	When i = 1, Data_type = program_number.
	160	20	When i = 2, Data_type = CCI_auth.
	8	1	When i = 3, Data_type = LTSID.
}			
}			
Request_datatype_nbr	8	1	Has the value of 3.
for(i=0;			
i<Request_datatype_nbr; i++)			
{			
Datatype_id	8	1	When i=0, Datatype_id has the value 28 (CCI_ack).
Datatype_id	8	1	When i=1, Datatype_id has the value 26 (program_number).
	8	1	When i=2, Datatype_id has the value 29 (LTSID).
}			
}			
}			

Table 38 - CP_data_cnf() Message Syntax in CCI SATP Transmission

Message Syntax	bits	bytes	Description
CP_data_cnf(){			
CP_data_cnf_tag	24	3	Has the value of 0x9F9003.
length_field()	8	1	Has the value of 0x22, size_indicator = 0, length_value = 34
CP_system_id	8	1	Has the value of 3. Values are listed in Table 12.
Send_datatype_nbr	8	1	Has the value of 3.
for(i=0; i<Send_datatype_nbr;			
i++)			
{			
Datatype_id	8	1	i = 0, Datatype_id has the value of 28 (CCI_ack).
	8	1	i = 1, Datatype_id has the value of 26 (program_number).
	8	1	i = 2, Datatype_id has the value of 29 (LTSID).
Datatype_length	16	2	i = 0, Datatype_length has the value of 0x0014.
	16	2	i = 1, Datatype_length has the value of 0x0002.
	16	2	i = 2, Datatype_length has the value of 0x0001.
for (j=0;			
j<Datatype_length; j++)			
{			
Data_type	160	20	When i = 0, Data_type = CCI_ack.
	16	2	When i = 1, Data_type = program_number.
	8	1	When i = 2, Data_type = LTSID.
}			
}			
}			
}			

Appendix A Luhn Check Digit (Normative)

The Luhn check digit is appended to each ID.

The Luhn check digit is calculated using the following algorithm.*

1. Convert the value into decimal format based on the CARD and Host ID fields.
2. Double the value of alternate digits beginning with the first right hand digit (least significant digit) and moving left.
3. Add the individual digits comprising the products obtained in step 2 to each of the unaffected digits in the original number.
4. Subtract the total obtained in step 3 from the next higher number ending in 0. This is equivalent to calculating the “tens complement” of the low order digit of the total. If the total obtained in step 3 is a number ending in 0, then the check digit is 0.

Example:

For the 40-bit Host_ID 0x01 2997 2A1F (hexadecimal):

1. Convert the 10 msb to the 3-digit manufacturer number 8.
2. Convert the 30 lsb to the 9-digit decimal unit number 697,772,799.
3. Concatenate these into the 12-digit value 008,697,772,799.
4. Separate this decimal number into odd and even digits starting from the right (least significant digit):

digit #: $^{12}_{11}10987654321$
 'odd' digits: 0, 6, 7, 7, 7, 9
 'even' digits: 0, 8, 9, 7, 2, 9

5. Multiply each 'odd' digit by 2:

0, 6, 7, 7, 7, 9 → 0, 12, 14, 14, 14, 18

6. Add each individual digit of the products above to the 'even' digits :

$[0 + 1 + 2 + 1 + 4 + 1 + 4 + 1 + 4 + 1 + 8] + [0 + 8 + 9 + 7 + 2 + 9] = 62$

7. Subtract the least significant digit of this sum from 10 to form the check digit:

$10 - 2 = 8$

8. Append this digit to the right of the decimal ID number for display to subscribers in the ID reporting screen:

0,086,977,727,998

(which may be displayed on screen as 0-086-977-727-998; note that the placement of the dashes is simply an example)

*Further information was available at <http://staff.sem.el.fi/~kribe/document/luhn.htm> as of 29 August 2003.

Appendix B Applying CPKey to DES Engine (Normative)

B.1 Method of Application

The cryptographic key is applied to many DES engines as a 64-bit value as described in [4] and [5]. The CableCARD-CP specification [13] defines generation of a 56-bit integer CPKey. The 64-bit key is generated from CPKey by adding a parity bit to each 7-bit block.

Starting with CPKey in a 56-bit format:

CPKey = $K_1 K_2 K_3 \dots K_{56}$ Where K_1 represents the most significant bit of CPKey.

By adding parity bits after each 7 bits of CPKey we get the 64-bit key:

$K_{64bit} = K_1 K_2 \dots K_7 P_1 K_8 \dots K_{14} P_2 \dots \dots K_{50} \dots K_{56} P_8$

where P_i SHALL be either 0 or 1 so that each octet has odd parity (i.e., there is an odd number of "1" bits).

For example, for an original value of CPKey:

CPKey = 0123456789abcd₍₁₆₎
 = 0000 0001 0010 0011 0100 0101 0110 0111 1000 1001 1010 1011 1100 1101₍₂₎

Break it into eight 7-bit blocks:

CPKey = 0000000 1001000 1101000 1010110 0111100 0100110 1010111 1001101₍₂₎

Add the parity bits as the last bit of each octet to get the 64-bit key:

$K_{64bit} = 00000001 10010001 11010000 10101101 01111001 01001100 10101110 10011011$ ₍₂₎
 = 0191d0ad794cae9b₍₁₆₎

B.2 Examples of CP Encryption of MPEG DATA in Transport Packets

This section shows examples of packets before and after DES encryption by the copy protection system. The encryption key used here is **0123456789ABCDEF**₍₁₆₎ in 64-bit format (or 00451338957377₍₁₆₎ in 56-bit format), which is shown in [5] as an example. The lines “C:” and “E:” for each example show the transport packet data before and after CP encryption respectively (cleartext and encrypted).

Example 1: A null packet.

```
C: 47 1f ff 10 ff ff ff ff ff ff ff ff ff ff ff ff ...
E: 47 1f ff 10 ff ff ff ff ff ff ff ff ff ff ff ff ...
```

CP encryption leaves the packets that don't belong to a copy protected program unchanged.

Example 2: A packet without adaptation field that belongs to a copy protected program.

```
C: 47 10 22 1c d4 75 09 40 c3 61 ec 26 1a 30 cf 1c c6 e1 d0 d1 ...
E: 47 10 22 dc 03 f9 77 f6 89 01 4a 9f 09 f0 ef bc 85 58 9f 9f ...
```

DES encryption starts right after the packet header. **transport_scrambling_control** field is changed from **00** to **11** (4th byte: **1c** to **dc**). This field could be either 10 (even key) or 11 (odd key) when seamless key refresh mechanism is introduced as per ECN-00075. Each 8-byte block in the packet payload is encrypted with DES ECB mode.

Example 3: A packet with adaptation field that belongs to a copy protected program.

```
C: 47 00 50 32 02 00 ff 88 f5 32 3e ac 87 eb 10 ...
... c3 d6 88 f7 32 32 ac af eb e0 78 41 11 (end of packet)
E: 47 00 50 f2 02 00 ff bb 5a ec 14 56 8b 66 b4 ...
... 80 50 cf cd ad 7e d1 de eb e0 78 41 11 (end of packet)
```

DES encryption starts after the adaptation field, which takes 3 bytes in this example (1 byte for **adaptation_field_length** and 2 bytes for the body). The payload is encrypted the same way except for the short block (5 bytes) at the end, which remains clear. **transport_scrambling_control** field is changed as described in Example 2 above.

Appendix C Revision History

OC-SP-MCCP-IF-I03-040831 contains modifications from the following ECNs.

ECN	Date Accepted	Summary
MCCP-IF-04.0583-2	4/9/04	Add CIT to MCCP
MCCP-IF-04.0652-1	8/13/04	ID Reporting Screen Reconciliation

OC-SP-MCCP-IF-I02-040402 contains modifications from the following ECNs.

ECN	Date Accepted	Summary
MCCP-IF-03.0528-1	2/4/04	Section 6.1 of the specification has a logical error which does not match the operation defined earlier in the specification nor similar operation defined in SCTE 41 2003

OC-SP-MCCP-IF-C01-050331 contains modifications from the following ECNs.

ECN	Date Accepted	Summary
MCCP-IF-04.0674-2	8/23/04	Require Host Authentication before any CA-decryption