

# **OpenCable™ Specifications**

## **CableCARD™ Copy Protection 2.0 Specification**

**OC-SP-CCCP2.0-I12-120531**

**ISSUED**

### **Notice**

This OpenCable specification 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 this 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, noninfringement, or fitness for a particular purpose of this document, or any document referenced herein.

© Cable Television Laboratories, Inc. 2004-2012

## Document Status Sheet

<b>Document Control Number:</b>	OC-SP-CCCP2.0-I12-120531			
<b>Document Title:</b>	CableCARD™ Copy Protection 2.0 Specification			
<b>Revision History:</b>	I01—March 31, 2005 I02—July 8, 2005 I03—June 22, 2006 I04—August 3, 2006 I05—January 5, 2007 I06—March 23, 2007 I07—June 15, 2007 I08—November 9, 2007 I09—May 8, 2009 I10—September 4, 2009 I11—May 12, 2011 I12—May 31, 2012			
<b>Date:</b>	May 31, 2012			
<b>Status:</b>	<del>Work in Progress</del>	<del>Draft</del>	<del>Issued</del>	<del>Closed</del>
<b>Distribution Restrictions:</b>	<del>author only</del>	<del>CL/Member</del>	<del>CL/Member/ Vendor</del>	<del>Public</del>

### Key to Document Status Codes:

<b>Work in Progress</b>	An incomplete document, designed to guide discussion and generate feedback, that may include several alternative requirements for consideration.
<b>Draft</b>	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.
<b>Issued</b>	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.
<b>Closed</b>	A static document, reviewed, tested, validated, and closed to further engineering change requests to the specification through CableLabs.

### Trademarks:

CableCARD™, CableHome®, CableLabs®, CableNET®, CableOffice™, CablePC™, DCAS™, DOCSIS®, DPoE™, EBIF™, eDOCSIS™, EuroDOCSIS™, EuroPacketCable™, Go2Broadband<sup>SM</sup>, M-Card™, M-CMTS™, OCAP™, OpenCable™, PacketCable™, PCMM™, PeerConnect™, and tru2way® are marks of Cable Television Laboratories, Inc. All other marks are the property of their respective owners.

# Contents

<b>1</b>	<b>SCOPE.....</b>	<b>1</b>
1.1	Introduction and Overview .....	1
1.2	Purpose of Document.....	1
1.3	Organization of Document.....	2
1.4	Historical Perspective .....	2
1.5	Requirements .....	3
<b>2</b>	<b>REFERENCES .....</b>	<b>4</b>
2.1	Normative References.....	4
2.2	Informative References .....	4
2.3	Reference Acquisition.....	5
2.3.1	<i>OpenCable Bundle Requirements.....</i>	<i>5</i>
2.3.2	<i>Other References.....</i>	<i>5</i>
<b>3</b>	<b>ACRONYMS, ABBREVIATIONS, DEFINED TERMS AND SYMBOLS.....</b>	<b>7</b>
<b>4</b>	<b>SYSTEM OVERVIEW .....</b>	<b>12</b>
4.1	Card and Host Mutual Authentication .....	12
4.2	ID Reporting and Headend Validation.....	12
4.2.1	<i>Reporting Card and Host Identification Information .....</i>	<i>12</i>
4.2.2	<i>Headend ID Validation.....</i>	<i>13</i>
4.3	Calculation of Copy Protection Keys.....	13
4.4	Copy Control Information.....	13
4.5	Encryption of Copy Protected Content .....	13
4.6	Card-Host Messages .....	14
4.7	Debug Modes Prohibited .....	14
<b>5</b>	<b>CARD AND HOST MUTUAL AUTHENTICATION.....</b>	<b>15</b>
5.1	Authentication Parameters .....	15
5.1.1	<i>X.509 Certificates .....</i>	<i>15</i>
5.1.2	<i>Copy Protection System Parameters.....</i>	<i>15</i>
5.1.3	<i>Binding Specific Parameters .....</i>	<i>15</i>
5.2	Copy Protection Initialization.....	16
5.2.1	<i>Binding Reinitialization.....</i>	<i>17</i>
5.3	Diffie-Hellman Public Key Calculation .....	17
5.3.1	<i>Diffie-Hellman Overview .....</i>	<i>17</i>
5.3.2	<i>Derivation of the Diffie-Hellman Public Keys.....</i>	<i>17</i>
5.3.3	<i>Authentication Parameter Exchange and Verification .....</i>	<i>18</i>
5.3.4	<i>Diffie-Hellman Shared Secret Key.....</i>	<i>19</i>
5.4	Shared Authentication Key .....	19
5.5	Completion of Authentication.....	19
5.6	Authentication Failures .....	19
5.6.1	<i>Host Response Time-out .....</i>	<i>19</i>
5.6.2	<i>Invalid Certificate .....</i>	<i>19</i>
5.6.3	<i>Other Authentication Failures .....</i>	<i>20</i>
5.7	Card Operation with Multiple Hosts .....	20
<b>6</b>	<b>ID REPORTING, VALIDATION, AND AUTHORIZATION .....</b>	<b>21</b>
6.1	Card and Host Identity Reporting .....	21
6.1.1	<i>Automated ID Reporting.....</i>	<i>21</i>

6.1.2	<i>Manual ID Reporting</i> .....	21
6.1.3	<i>Host Request for ID Reporting Screen</i> .....	22
6.2	Headend Validation .....	22
6.2.1	<i>ID Registration</i> .....	22
6.2.2	<i>ID Validation</i> .....	22
6.3	CA Authorization of Copy Protected Content .....	23
<b>7</b>	<b>CRYPTOGRAPHIC FUNCTIONS</b> .....	<b>24</b>
7.1	DFAST.....	25
7.2	Random Integer Generation .....	25
7.3	SHA-1 Secure Hash Algorithm.....	25
7.4	Representation of Large Values as Octets [Informative] .....	26
7.5	RSA Digital Signatures .....	26
<b>8</b>	<b>COPY PROTECTION KEY GENERATION</b> .....	<b>27</b>
8.1	Basic Key Generation Protocol.....	27
8.2	Copy Protection Key (CPKey) Calculation .....	28
8.3	CP Session Timer.....	29
8.4	CPKey Generation Flow .....	29
8.5	CA Initiated CPKey Refresh.....	31
<b>9</b>	<b>COPY CONTROL INFORMATION (CCI)</b> .....	<b>32</b>
9.1	CCI Definition .....	32
9.1.1	<i>EMI - Digital Copy Control Bits</i> .....	32
9.1.2	<i>APS - Analog Protection System</i> .....	32
9.1.3	<i>CIT – Constrained Image Trigger</i> .....	33
9.1.4	<i>RCT – Redistribution Control Trigger</i> .....	33
9.2	Associating CCI with a Service .....	34
9.3	Conveying CCI from Headend to Card.....	34
9.4	Conveying CCI from Card to Host .....	34
9.4.1	<i>CCI Delivery Instances</i> .....	34
9.4.2	<i>CCI Delivery Protocol</i> .....	35
9.4.3	<i>Host Application of CCI</i> .....	37
<b>10</b>	<b>TRANSPORT ENCRYPTION FROM CARD TO HOST</b> .....	<b>38</b>
10.1	CP-Encryption as a Function of CA-Scrambling and CCI Value.....	38
10.2	CP-Encryption Rules .....	38
10.2.1	<i>Transport Scrambling Control Field</i> .....	39
10.3	Timing of Encryption Mode Transitions.....	39
10.4	DESKeys for Single and Multi-stream Modes.....	39
10.4.1	<i>DESKey for Single Stream Mode</i> .....	39
10.4.2	<i>DESKey for Multi-Stream Mode</i> .....	39
<b>11</b>	<b>CARD-HOST MESSAGING PROTOCOLS</b> .....	<b>41</b>
11.1	Message Protocol Overview .....	41
11.2	Card-Host Message Parameters .....	42
11.3	Opening a CP Session.....	44
11.3.1	<i>Host CP Support Capability Evaluation</i> .....	45
11.4	Card-Host Mutual Authentication Message Protocol.....	46
11.4.1	<i>Mutual Authentication Data Exchange Messages</i> .....	46
11.4.2	<i>AuthKey Verification Messages</i> .....	48
11.5	Copy Protection Key Generation Protocol.....	49
11.5.1	<i>Copy Protection Key Data Exchange Messages</i> .....	49
11.5.2	<i>Card CPKey Generation Message</i> .....	50
11.5.3	<i>Host CPKey Generation Message</i> .....	50

11.6	Card-Host CPKey Synchronization Protocol .....	51
11.6.1	CPKey Sync Message.....	51
11.6.2	Card CPKey Ready Message.....	51
11.6.3	Host CPKey Ready Message.....	51
11.7	CCI Delivery Protocol .....	52
11.7.1	CCI Delivery Message .....	52
11.7.2	Card CCI Challenge Message .....	52
11.7.3	Host CCI Response Message .....	53
11.7.4	CCI Delivery Message .....	54
11.7.5	CCI Acknowledgement Message.....	55
11.8	Card Validation Status .....	56
<b>ANNEX A</b>	<b>LUHN CHECK DIGIT (NORMATIVE).....</b>	<b>58</b>
<b>ANNEX B</b>	<b>APPLYING CPKEY TO DES ENGINE (NORMATIVE).....</b>	<b>59</b>
B.1	Method of Application.....	59
B.2	Examples of S-Mode CP Encryption of MPEG DATA in Transport Packets .....	60
B.3	M-Mode Transport Packet Encryption with Triple DES .....	60
B.4	Examples of CP Encryption of MPEG DATA in Transport Packets .....	65
<b>APPENDIX I</b>	<b>REVISION HISTORY .....</b>	<b>67</b>

## List of Figures

Figure 5.3-1	- Diffie-Hellman Key Agreement.....	18
Figure 5.6-1	- Example Card Authentication Failure Notification Message.....	20
Figure 5.6-2	- Example Host Authentication Failure Notification Message .....	20
Figure 5.6-3	- Example CP System Failure Notification Message.....	20
Figure 6.1-1	- Example ID Reporting Screen .....	22
Figure 7-1	- Encryption Key Generation.....	24
Figure 7-2	- S-Mode Encryption .....	24
Figure 7-3	- M-Mode Encryption.....	25
Figure 8.4-1	- Card CPKey Session Flow Chart for S-Mode.....	30
Figure 8.4-2	- Host CPKey Session Flow Chart .....	31
Figure 9.4-1	- CCI Delivery Sequence.....	34
Figure 9.4-2	- Two Examples of CCI Transfer During CPKey Refresh .....	37
Figure 11.1-1	- Card-Host Message Protocol Flow .....	41
Figure B.3-1	- 3DES Encryption and Decryption, ECB mode .....	61
Figure B.3-2	- 2-key Triple DES, EDE-121 mode, Encryption.....	61
Figure B.3-3	- 2-key Triple DES, EDE-121 mode, Decryption.....	62
Figure B.3-4	- MPEG Transport Packet Scrambling, Case 1.....	63
Figure B.3-5	- MPEG Transport Packet Descrambling, Case 1.....	63
Figure B.3-6	- MPEG Transport Packet Scrambling, Case 2.....	64
Figure B.3-7	- MPEG Transport Packet Descrambling, Case 2.....	64

## List of Tables

Table 5.1-1 - System Parameters .....	15
Table 5.1-2 - Length of Parameters in the Host Authentication.....	16
Table 8.2-1 - Length of Keys and Parameters Used in the Key Generation .....	28
Table 9.1-1 - CCI Bit Assignments.....	32
Table 9.1-2 - EMI Values and Copy Permissions.....	32
Table 9.1-3 - APS Value Definitions.....	33
Table 9.1-4 - CIT Value Definitions.....	33
Table 9.1-5 - RCT Value Definitions .....	33
Table 9.1-6 - Host DTCP Outputs .....	33
Table 10.1-1 - CP-Encryption Based on CA-Scrambling and CCI Value .....	38
Table 10.2-1 - MPEG transport_scrambling_control Values .....	39
Table 11.1-1 - Message Reference Sections .....	42
Table 11.2-1 - CP_system_id Values.....	42
Table 11.2-2 - CP System Message Parameters.....	43
Table 11.3-1 - Copy Protection Open Session Information.....	44
Table 11.3-2 - CableCARD Copy Protection Resource.....	44
Table 11.3-3 - CableCARD Copy Protection Resource - Host Optional .....	44
Table 11.3-4 - Host CP Support Capability Evaluation Messages.....	45
Table 11.3-5 - Card's CP Support Request Message Syntax.....	45
Table 11.3-6 - Host's CP Support Confirm Message Syntax.....	45
Table 11.3-7 - CP_system_id_bitmask Values .....	45
Table 11.4-1 - Authentication Data Exchange Messages.....	46
Table 11.4-2 - Card Authentication Data Message Syntax .....	47
Table 11.4-3 - Host Authentication Data Message Syntax .....	48
Table 11.4-4 - AuthKey Verification Messages.....	48
Table 11.4-5 - Card Request for Host AuthKey Message Syntax.....	49
Table 11.4-6 - Host Reply with AuthKey Message Syntax.....	49
Table 11.5-1 - CPKey Generation Messages .....	49
Table 11.5-2 - Card CPKey Generation Message Syntax.....	50
Table 11.5-3 - Host's CPKey Generation Message Syntax .....	50
Table 11.6-1 - Card-Host CPKey Synchronization Messages .....	51
Table 11.6-2 - Card CPKey Ready Message Syntax .....	51
Table 11.6-3 - Host CPKey Ready Message Syntax.....	51
Table 11.6-4 - Host Status_field Value.....	51
Table 11.7-1 - CCI Delivery Protocol Messages .....	52
Table 11.7-2 - Card's CCI Challenge Message Syntax.....	53
Table 11.7-3 - Host's CCI Response Message Syntax.....	54
Table 11.7-4 - CCI Delivery Message Syntax .....	55
Table 11.7-5 - CCI Acknowledgement Message Syntax .....	56
Table 11.8-1 - Card Validation Status Messages.....	56
Table 11.8-2 - Host Validation Status Request Message Syntax (type 4 ver 3).....	56

Table 11.8-3 - Card Validation Status Reply Message Syntax (type 4 ver 3).....	57
Table 11.8-4 - Card Validation Status_field Value.....	57

This page intentionally left blank.



# 1 SCOPE

## 1.1 Introduction and Overview

In digital cable systems the system operator protects selected content against unauthorized access with a conditional access scrambling system. A properly authorized CableCARD security module (Card) removes the conditional access scrambling and, based on the Copy Control Information (CCI) from the Headend, may rescamble the content before delivering it to consumer receivers and set-top terminals (Hosts) across the Card-Host Interface defined in OpenCable CableCARD 2.0 Interface Specification [CCIF].

This document defines the characteristics and normative specifications for the system that prevents unauthorized copying of certain content as it crosses from Card to Host on the Card-Host Interface. This interface supports the delivery of up to six independent transport streams across the Card-Host Interface. This specification describes how copy protection is achieved on the interface in two distinct modes:

- 1) Single Stream Mode (S-Mode) for use with or between Cards and Hosts capable of supporting only one MPEG program in one transport stream on the Transport Stream Interface as described in [SCTE 28] and the S-Mode specified in [CCIF].
- 2) Multi-Stream Mode (M-Mode) for use between a Card and Host both implementing the Multistream Mode interface as specified by the M-Mode in [CCIF].

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

This specification provides methods for:

- a) Authenticating Cards and Hosts
- b) Binding the Card and Host including Diffie-Hellman key exchange
- c) Copy protection key generation
- d) Rescrambling copy protected content (after the Card has descrambled it using a conditional access system)
- e) Descrambling by the Host
- f) Authenticated delivery of Copy Control Information to the Host
- g) Recognition and deauthorization of Hosts that are determined to be fraudulent or non-compliant

This specification 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 2.3 for contact information for such a license.

## 1.2 Purpose of Document

This document specifies the means of protecting designated content as it passes from the Card back to the Host after CA-descrambling in the Card. Copy Protection protects this content against unauthorized use and retransmission. This document specifies the public aspects of the copy protection system. The security of any protection must also rely upon secrets and methods provided by a "trust entity" designated by the content provider, herein referred to as "CHICA."

## 1.3 Organization of Document

This document lays out the specification of the copy protection system in the following sequence:

- a) System Overview – describing the overall systems operation.
- b) Card and Host Mutual Authentication - which defines the means for each device to confirm the other holds trusted source secrets and certificates and from which they derive new shared secrets.
- c) ID Reporting, Validation, and Authorization – defines the method of reporting device IDs to the systems operator, validation of those IDs, and authorization to descramble protected MPEG programs.
- d) Cryptographic Functions – defines how the cryptographic methods are applied.
- e) Copy Protection Key Generation – defines the means to generate and refresh a shared key for copy protection encryption in the Card and decryption in Host.
- f) Copy Control Information – defines the means for the Card to inform the Host of content usage permissions.
- g) Transport Encryption from Card to Host – defines how selected content is protected as it passes from the Card back to the Host.
- h) Card-Host Messaging Protocols – defines the detailed protocol elements for communication of copy protection messages on the Command Channel [CCIF].

## 1.4 Historical Perspective

This specification has its origins in the National Renewable Security Standard, 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 TV shows.

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 2001 revised to SCTE 41 2003 and later to ANSI/SCTE 41 2004. This document updates that work to current industry practice and adds the capability to copy protect simultaneous delivery of multiple content streams across the CableCARD interface.

## 1.5 Requirements

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

“SHALL”	This word means that the item is an absolute requirement of this specification.
“SHALL NOT”	This phrase means that the item is an absolute prohibition of this specification.
“SHOULD”	This word 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 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.

## 2 REFERENCES

### 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.

All references are subject to revision, and parties to agreement based on this specification are encouraged to investigate the possibility of applying the most recent editions of the documents listed below.

References are either specific (identified by date of publication, edition number, version number, etc.) or non-specific:

- For a specific reference, subsequent revisions do not apply.
- For a non-specific, non-Bundle reference, the latest version applies.
- For non-specific CableLabs references that are part of the [OC-BUNDLE], the versions mandated in a particular Bundle apply.

[CCIF]	CableCARD Interface 2.0 Specification, OC-SP-CCIF2.0, Cable Television Laboratories, Inc. Referenced in [OC-BUNDLE].
[FIPS 46-3]	FIPS PUB 46-3: Data Encryption Standard (DES), October 25, 1999.
[FIPS 81]	FIPS PUB 81: DES Modes of Operation, December 21, 1980.
[FIPS 140-2]	FIPS PUB 140-2: Security Requirements for Cryptographic Modules, May 25, 2001.
[FIPS 180-2]	FIPS PUB 180-2: Secure Hash Standard, August 2000.
[FIPS 186-2]	FIPS PUB 186-2, Digital Signature Standard, Federal Information Processing Standards Publications (FIPS PUB), January 27, 2000.
[MPEG]	ISO/IEC 13818-1 (2000): Information Technology - Generic coding of moving pictures and associated audio information: Systems.
[OC-BUNDLE]	OpenCable Bundle Requirements, OC-SP-BUNDLE. See Section 2.3.1 to acquire this specification.
[RSA1]	PKCS #1 v2.1: RSA Cryptography Standard, June 14, 2002.
[SCTE 41]	ANSI/SCTE 41 2004, POD Copy Protection System.
[SCTE 28]	ANSI/SCTE 28 2004, HOST-POD Interface Standard.
[X.509]	ITU-T Recommendation X.509: Information Technology – Open Systems Interconnection – The Directory: Public-key and Attribute Certificate Frameworks, March 2000.

### 2.2 Informative References

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

[HOST]	OpenCable Host Device 2.1 Core Functional Requirements, OC-SP-HOST2.1, Cable Television Laboratories, Inc. Referenced in [OC-BUNDLE].
--------	---

- [RSA3] PKCS #10 V1.7: Certification Request Syntax Standard, May 2000.
- [SEC] OpenCable System Security Specification, OC-SP-SEC, Cable Television Laboratories, Inc. Referenced in [OC-BUNDLE].

## 2.3 Reference Acquisition

### 2.3.1 OpenCable Bundle Requirements

The OpenCable Bundle Requirements specification [OC-BUNDLE] indicates the set of CableLabs specifications required for the implementation of the OpenCable Bundle. The version number of [OC-BUNDLE] corresponds to the release number of the OpenCable Bundle that it describes. One or more versions of [OC-BUNDLE] reference this specification. Current and past versions of [OC-BUNDLE] may be obtained from CableLabs at <http://www.cablelabs.com/opencable/specifications>.

### 2.3.2 Other References

***CableLabs Specifications, DFAST Technology, CHILA, and CHICA: Cable Television Laboratories, Inc.***

Cable Television Laboratories, Inc., 858 Coal Creek Circle, Louisville, CO 80027; Telephone: +1-303-661-9100; Facsimile: +1-303-661-9199; E-mail: [opencable@cablelabs.com](mailto:opencable@cablelabs.com); URL: [www.cablelabs.com](http://www.cablelabs.com)

***EIA Standards: Electronic Industries Association***

Global Engineering Documents, World Headquarters, 15 Inverness Way East, Englewood, CO 80112-5776; Telephone +1-800-854-7179; Facsimile: +1-303-397-2740; E-mail: [global@iht.com](mailto:global@iht.com); URL: <http://global.iht.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](mailto: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: +1-800-854-7179; E-mail: [global@iht.com](mailto:global@iht.com); URL: <http://global.iht.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 +1-703-605-6000; FAX: +1-703-321-8547; E-mail orders: [orders@ntis.fedworld.gov](mailto:orders@ntis.fedworld.gov); URL: <http://www.itl.nist.gov/fipspubs/>

***NIST Publications: National Institute of Standards and Technology***

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

***RSA Security***

RSA Security, Inc, 174 Middlesex Turnpike, Bedford, MA 01730; Telephone: +1-781-515-5000; FAX: +1-781-515-5010; 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: +1-610-363-6888; Facsimile: +1-610-363-5898; E-mail: [standards@scte.org](mailto:standards@scte.org) ;  
URL: <http://www.scte.org/standards/standardsavailable.html>

### 3 ACRONYMS, ABBREVIATIONS, DEFINED TERMS AND SYMBOLS

<b>APDU</b>	Application Protocol Data Unit: a command, query, and reply message exchange protocol between Card and Host.
<b>APS</b>	Analog Protection System, a two bit CCI-A value setting copy control modes for analog outputs.
<b>AuthKey</b>	Authentication Key, calculated by both the Card and Host as part of the Host authentication process.
<b>Authenticated</b>	Employing or resulting from the use of Authentication: trusted.
<b>Authentication</b>	A means to securely confirm that a message or device is worthy of trust. Specifically a procedure for the Card and Host to securely confirm that the other is an authentic device for CableCARD-CP binding.
<b>Binding</b>	The process of Card-Host mutual authentication and headend validation. Binding completes, and the Card-Host pair is fully bound, when both authentication and validation are successfully completed.
<b>Bound</b>	The Card-Host pair is bound together by A) the mutual authentication process of generating a shared secret DHKey and a public AuthKey, and B) validation of their ID's by the CA System. The Card-Host pair is fully bound when both steps have completed successfully.
<b>CA, CA System</b>	Conditional Access, Conditional Access System – secures delivery of cable services to the Card.
<b>CA-only</b>	The Card mode of CA-descrambling content and returning it to the Host in-the-clear.
<b>Cable</b>	The Cable Television industry, services, systems, or equipment.
<b>CableCARD-CP</b>	CableCARD copy protection, as specified in this document.
<b>Card</b>	A PCMCIA card distributed by cable providers and inserted into a Host to enable reception of premium services without a separate cable receiver, also called CableCARD Device and “Point of Deployment” (POD) module.
<b>Card_ID</b>	The Card's unique identification number.
<b>CCI</b>	Copy Control Information.
<b>CCI-A</b>	The subset of CCI bits, APS and CIT, controlling Host analog outputs.
<b>CCI-D</b>	The subset of CCI bits, EMI and RCT, controlling Host digital outputs.
<b>CHICA</b>	CableCARD-Host Interface Certificate Authority, root X.509 certificate administrator for X.509 certificates on the Card-Host interface identified under CHILA; formerly PHICA.
<b>CHILA</b>	CableCARD-Host Interface Licensing Agreement, formerly identified as PHILA, covers the DFAST technology and specifies the Certificate Authority –CHICA.

<b>CIT</b>	Constrained Image Trigger, a single bit CCI-A value controlling use of Image Constraint on high definition analog outputs.
<b>Constrained Image</b>	The visual equivalent of not more than 520,00 pixels per frame (e.g., an image with resolution of 540 vertical pixels by 960 horizontal pixels for a 16:9 aspect ratio). A Constrained Image can be output or display using video processing techniques such as line doubling or sharpening to improve the perceived quality of the image.
<b>CP</b>	Copy Protection.
<b>CPKey</b>	Copy Protection Key. A shared secret key derived between the Card and Host, and used to derive DESKeys.
<b>CP System</b>	The copy protection system described in this specification.
<b>CRL</b>	Certificate Revocation List: the means of reporting bad Card and Host IDs to cable headends.
<b>DES</b>	Data Encryption Standard.
<b>DESKey<sub>m</sub></b>	The final key entered into the DES-ECB processor to encrypt or decrypt MPEG program 'm'.
<b>Device Certificate</b>	The XCA certificate unique to each Card or Host device, Card_DevCert and Host_DevCert.
<b>DFAST</b>	Dynamic Feedback Arrangement Scrambling Technique, a component of the encryption algorithm.
<b>DH</b>	Diffie-Hellman, a public key agreement protocol based on the intractability of taking discrete logarithms over the integer field.
<b>DSG</b>	DOCSIS Set-top Gateway, a method of using DOCSIS protocols to report IDs to the headend.
<b>DTCP</b>	Digital Transmission Content Protection; see Host specification.
<b>ECB</b>	Electronic Code Book.
<b>ECM-PID</b>	Entitlement Control Message – Packet Identifier. This 13-bit field value indicates the PID of the Transport Stream packets that contain the ECM information associated with a specific MPEG program request by the Host by the CA_system_ID reported by the Card (ECM_stream). The Card and Host associate one and only one ECM-PID value with each copy-protected MPEG program; see Section 10.4.2.
<b>EMI</b>	Encryption Mode Indicator, a two bit CCI-D value setting copy protection modes for digital outputs.
<b>Encrypted</b>	Data modified by a CableCARD device to prevent unauthorized access (compare with "scrambled").
<b>EPN Mode</b>	Encryption Plus Nonassertion Mode, a redistribution protection mode of DTCP that does not limit copies. Note: In DTCP, EPN Mode is signaled with an EPN bit value of zero.



<b>ES</b>	Elementary Stream – a component of an MPEG-TS identified with a unique PID.
<b>Headend</b>	The cable operator’s facility, which acts as the source of cable signals, services, and conditional access control.
<b>Host</b>	The consumer device used to access and navigate cable content. Typically a digital TV or set-top DTV receiver.
<b>Host_ID</b>	The Host’s unique identification number found in the Host Device Certificate.
<b>Image Constraint</b>	See Constrained Image.
<b>In-the-clear</b>	Without encryption or scrambling by either the CA or CP system.
<b>lsb</b>	Least Significant Bit, of a specified binary value.
<b>LTSID</b>	Local Transport Stream ID, assigned by the Host in M-Mode.
<b>Manufacturer Certificate</b>	The XCA certificate that CHICA issues to the manufacturer of Device Certificates. It is used to issue Device Certificates and to verify them during Binding.
<b>MMI</b>	Man Machine Interface.
<b>MPEG</b>	The ISO/IEC 13818 specifications and ISO/IEC 13818-1 in particular.
<b>MPEG program</b>	A stream of data identified within an MPEG transport stream by an MPEG program number. It may be a stream of indefinite duration, unrelated to the content it conveys. Typically includes multiple PES (program elementary streams) and multiple PIDs (packet ID). MPEG programs often contain data representing content, e.g., a “movie” or “TV program” but the phrase “MPEG program” refers to a data structure not the content. For M-Mode, an MPEG program is uniquely identified by its LTSID and MPEG program number.
<b>msb</b>	Most Significant Bit, of a specified binary value.
<b>M-Mode</b>	Multi-Stream Mode as defined in [CCIF].
<b>Nonce</b>	A short-term use random value generated fresh for each Card-Host protocol sequence to improve security by making each instance unique.
<b>Non-zero</b>	An adjective meaning a value other than zero, or a set of values, one or more not zero.
<b>OpenCable Bundle</b>	The OpenCable Bundle defines a set of specifications required to build a specific version of an OpenCable device. See [OC-BUNDLE].
<b>Pass-through</b>	The Card processing mode of passing a CA-scrambled content back to the Host unchanged, leaving it unusable by the Host.
<b>PES</b>	Packetized Elementary Stream – an ES carried in a PES protocol as defined in [MPEG].
<b>PID</b>	Packet Identifier – the 13-bit MPEG transport stream packet header ID.

<b>POD module</b>	Synonymous with “POD”, “point of deployment module” and Card. A detachable device distributed by cable providers and inserted into a Host connector to enable reception of encrypted services.
<b>RCT</b>	The Redistribution Control Trigger is a CCI-D bit activating protection on digital outputs.
<b>RDC</b>	Return Data Channel: a communication channel on the coaxial cable that delivers the main cable service but running “upstream” from home to the headend.
<b>Rescramble</b>	The Card processing mode of CA-descrambling and CP-scrambling selected content.
<b>Root Certificate</b>	The root XCA certificate used by CHICA to issue Manufacturer Certificates and by the Card and Host to validate each other’s Manufacturer Certificate.
<b>RSA algorithm</b>	An RSA Security defined commercial public key cryptographic algorithm.
<b>Scrambled</b>	Content modified by a CA system to prevent unauthorized access (compare with "encrypted").
<b>SHA-1</b>	Secure Hash Algorithm, a cryptographic compression function, see [FIPS 180-2].
<b>SPDU</b>	Session Protocol Data Unit (SPDU).
<b>S-Mode</b>	Single Stream Mode as defined in [CCIF].
<b>T</b>	Time period variable set to different values in S-Mode and M-Mode to allow more time for the more complex CPKey generation. Also effects CCI delivery delay timing.
<b>Validation</b>	The process of reporting Card_ID and Host_ID to the system operator, recording those values, checking them against a revocation list, reporting the validated ID’s back to the Card, and the Card confirming the message was sent by the CA-system and that the ID’s match the values the Card reported.
<b>X.509</b>	ITU-T Recommendation X.509.
<b>XCA</b>	X.509 certificate authority.
<b>Zero</b>	An adjective meaning a value of zero or a set of values all equal to zero.
<b> </b>	Binary concatenation operator symbol, the “bar character”, means combine the bit strings of two binary parameters, e.g., for A = 111b and B = 000b, A B = 111000b.
<b>C<sup>D</sup></b>	Exponents are displayed in superscript. C <sup>D</sup> means raise the first value, C, to the exponential power of the second, superscripted, value D. For C = 2 and D = 3, C <sup>D</sup> = 2 <sup>3</sup> = 8.
<b>⊕</b>	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, 1b, if one or the other but not both bits is true, 1b. 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.
<b>0x</b>	number prefix indicating that the following number is an integer expressed in hexadecimal format. Space characters are inserted each 4 characters to improve readability.

- b** number suffix indicating that the preceding number is an integer expressed in binary format. Space characters are inserted each 4 characters to improve readability.

## 4 SYSTEM OVERVIEW

This copy protection system defines protection of content delivered to the Host as CA-scrambled MPEG programs, passed to the Card for CA-descrambling, and returned to the Host across the CableCARD Interface. The Host will apply content usage restrictions as indicated by CCI (Copy Control Information). The Card will deliver a CCI value to the Host for all programs it CA-descrambles. The Card will apply CP-encryption to some programs for protection as they return to the Host.

### 4.1 Card and Host Mutual Authentication

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

The Card will not CA-descramble any content until it has successfully completed authentication of the Host.

To complete the authentication process, the Card and Host use the following steps:

- a) Calculate a Diffie-Hellman (DH) public key from stored secrets and a generated random integer.
- b) Sign their DH public key with their X.509 private key (that corresponds to the public key embedded in their X.509 Device Certificate).
- c) Send their Device Certificate, Manufacturer Certificate, DH public key, and signature of the DH public key to the other device.
- d) Verify the other device's signature on the DH public key and validate its certificate chain.
- e) Calculate a shared secret key, DHKey, that is unique to each binding.
- f) Calculate and exchange a long term authentication key, AuthKey.
- g) In addition, the Card confirms that the AuthKey requested from the Host matches its internally calculated AuthKey.

When the steps above are finished, the Card and Host share identical DHKey and AuthKey values and the binding authentication is complete.

The Card MAY CA-descramble zero CCI-D content for authenticated Hosts.

### 4.2 ID Reporting and Headend Validation

The Card SHALL NOT CA-descramble any non-zero CCI-D content until Headend Validation is completed.

The Card 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.

#### 4.2.1 Reporting Card and Host Identification Information

The Card extracts the Card\_ID from its Device Certificate and the Host\_ID from the authenticated Host Device Certificate for reporting purposes. Two means are employed to report these IDs. In order of preference:

**Automated:** If the Card has an active electronic means to report IDs to the headend, such as an active cable plant RDC and a Host RDC transmitter, or via a DOCSIS cable modem, the Card may use it to report the IDs to the headend.

**Manual:** If an automated means is not available, the Card 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.

#### 4.2.2 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 Card with the validated IDs.

The Card SHALL enable CA-descrambling of all content after both: a) receiving and authenticating this message, and b) confirming validated IDs are the same as its current binding.

### 4.3 Calculation of Copy Protection Keys

The Card and Host each derive an identical 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 Card-Host pair and to the key session. Content encrypted with this key by the Card can only be decrypted by its bound Host.

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

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

In Single Stream Mode (S-Mode) the Card and Host use the shared CPKey directly as DESKey<sub>0</sub> to initialize a single DES process. In Multi-Stream Mode (M-Mode), the Card and Host generate a unique DESKey<sub>m</sub> for each current MPEG program to initialize a triple-DES process.

### 4.4 Copy Control Information

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 triggers image constraint on analog outputs, one bit triggers redistribution control, and two bits are reserved.

Copy control information (CCI) is sent from the Card to inform the Host of the level of copy protection required. CCI is sent in the clear to the Host, but the integrity of the information is authenticated with the simple protocol described in Section 9.4.2.

### 4.5 Encryption of Copy Protected Content

Copy-protected content will arrive at the Host in a data stream scrambled by the CA System. The selected stream passes from the Host to the Card while still protected by CA-scrambling.

After the Host is authenticated, the Card may CA-descramble zero CCI-D content.

When fully bound to an authenticated and validated Host, the Card may also CA-descramble non-zero CCI-D content. The Card may CP-encrypt content to protect it on return to the Host across the Card-Host interface.

In Single stream mode (S-Mode), the CP-encryption uses single DES in ECB mode and the computed Copy Protection Key before sending it across the interface to the Host. In Multi-stream mode (M-Mode), the CP encryption is triple-DES (ABA) in ECB mode.

The Card receives content from the Host, processes it in one of the following four modes, and returns it to the Host:

<b>Clear mode</b>	No change of an in-the-clear MPEG program, which returns to the Host in-the-clear
<b>Pass-through mode</b>	No change of a CA-scrambled MPEG program, returning it to the Host in an unusable state
<b>CA-only mode</b>	CA-descrambles an MPEG program and returns it to the Host in-the-clear
<b>Rescramble mode</b>	CA-descrambles and CP-encrypts an MPEG program for protected return to the Host

The mode employed depends upon:

- CA-authorization of the Card (the Card must be CA-authorized for CA-only and Rescramble modes)
- The status of authentication and validation as described above in this section
- CA-scrambling state and CCI value as defined in Section 10.1
- The option of the Card, for zero CCI-D content.

The Card SHALL CP-encrypt each MPEG program using a unique DESKey, when in Rescramble mode.

## 4.6 Card-Host Messages

Data channel and extended channel messages between the Card and Host do not require protection and are exchanged in-the-clear.

The Host SHALL deny any CP-related MMI dialog open request when in an off-state or any non-video-viewing state.

The Host SHALL grant any CP-related MMI dialog open request when in a video viewing state.

## 4.7 Debug Modes Prohibited

Cards containing a CHICA-designated production Card-DevCert SHALL NOT provide any operating modes, including test and debug modes, that bypass or compromise copy protection security.

Hosts containing a CHICA-designated production Host-DevCert SHALL NOT provide any operating modes, including test and debug modes, that bypass or compromise copy protection security.

## 5 CARD AND HOST MUTUAL AUTHENTICATION

The Card SHALL NOT CA-descramble any MPEG program until Mutual Authentication has successfully completed by confirming that the AuthKey requested from the Host is identical to its own.

### 5.1 Authentication Parameters

Three types of parameters are employed in the mutual authentication of Cards and Hosts.

#### 5.1.1 X.509 Certificates

The Card SHALL contain a CHICA-authorized X.509 version 3 Device Certificate [X.509] and matching private key used for creating digital signatures. This certificate includes a unique ID and the public key provided to other devices to validate the digital signatures.

The Host SHALL contain a CHICA-authorized X.509 version 3 Device Certificate [X.509] and matching private key.

The Card SHALL contain the Manufacturer Certificate that was used to sign its Device Certificate, and the CHICA Root Certificate that is used to sign all Manufacturer Certificates.

The Host SHALL contain the Manufacturer Certificate that was used to sign its Device Certificate, and the CHICA Root Certificate that is used to sign all Manufacturer Certificates.

#### 5.1.2 Copy Protection System Parameters

Table 5.1-1 defines system parameter length and source:

**Table 5.1-1 - System Parameters**

Parameter	Size (bits)	Source of Parameter
Card_ID (part of Device Certificate)	64 bits	CHICA and manufacturer
Host_ID (part of Device Certificate)	40 bits	CHICA and manufacturer
Diffie-Hellman prime (n)	1024 bits	CHICA
Diffie-Hellman base (g)	1024 bits	CHICA

The Card SHALL have a 64-bit Card\_ID value with bits numbered 63 through 40 (the 24 most significant bits) set to zero, bits 39 through 30 set to the Card's CHICA-assigned manufacturer number, and bits numbered 29 to 0 (the 30 least significant bits) set to a unit number in the range of zero to 999,999,999, inclusive, that is unique to the manufacturer number.

The Host SHALL have a 40-bit Host\_ID value with bits numbered 39 through 30 (the 10 most significant bits) set to their CHICA-assigned manufacturer number, and bits numbered 29 to 0 (the 30 least significant bits) set to a unit number in the range of zero to 999,999,999, inclusive, that is unique to the manufacturer number.

#### 5.1.3 Binding Specific Parameters

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

DH_pubKey <sub>C</sub>	The Card Diffie-Hellman public key. Derived and unique for each Card-Host binding.
DH_pubKey <sub>H</sub>	The Host Diffie-Hellman public key. Derived and unique for each Card-Host binding.
DHKey	The Diffie-Hellman shared secret key.
AuthKey <sub>C</sub>	The authentication key derived by the Card and used to verify AuthKey <sub>H</sub> request from the Host.
AuthKey <sub>H</sub>	The authentication key derived by the Host and sent to the Card for verification against AuthKey <sub>C</sub> .
AuthKey	The shared authentication key.
x, y	Diffie-Hellman private exponents for Card and Host respectively. Random integers generated uniquely for each binding.

**Table 5.1-2 - Length of Parameters in the Host Authentication**

Parameter	Size (bits)
Diffie-Hellman Private Exponents (x,y)	160
Diffie-Hellman public keys (DH_pubKey <sub>C</sub> DH_pubKey <sub>H</sub> )	1024
Diffie-Hellman shared secret key (DHKey)	1024
Authentication key (AuthKey)	160

## 5.2 Copy Protection Initialization

Card-Host CP Initialization SHALL occur on each power-up or Card hot-insertion as follows:

- The Host SHALL report the Copy Protection Resource to the Card during the profile inquiry process [CCIF].
- To initiate copy protection, the Card SHALL disable CA descrambling of all content and send an open\_session\_request() SPDU to a Copy Protection Resource reported by the Host and listed in Table 11.3-2.
- The Host SHALL reply with an open\_session\_response() SPDU and, if opened, a session number for all session APDUs.
- The Host SHALL open only Copy Protection resources listed in Table 11.3-2.
- The Card SHALL send the CP\_open\_req() APDU to evaluate the Host's support for the Card-CP system. See Section 11.3.1.
- In response to the CP\_open\_req() APDU, the Host SHALL reply with CP\_open\_cnf() indicating support for System 2 (CableCARD-CP System).
- If the CP\_open\_cnf() APDU received from the Host does not indicate support for System 2 (CableCARD-CP System), the Card SHALL treat the Host as if its Device Certificate is invalid.
- If the Card contains a stored, non-zero, AuthKey in its non-volatile memory, it SHALL request the Host's AuthKey.
- The Host SHALL respond to a Card's request with its AuthKey, if available, and otherwise, respond with a value of zero.
- The Card SHALL compare AuthKeys, and if the Host's AuthKey matches the Card's stored AuthKey, the authentication is complete, as restored from a previous binding.



- After completion of authentication, the Card SHALL proceed with ID reporting and Headend Validation. See Section 6.
- If the Host's AuthKey does not match the Card stored AuthKey, the Card 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.
- If the Card does not contain a stored non-zero AuthKey, it SHALL record the non-authenticated and non-validated states in non-volatile memory and proceed with Diffie-Hellman public key calculation.

### 5.2.1 Binding Reinitialization

The Card SHALL comply with a CA System request to initiate full copy protection reinitialization, as if the Card were inserted into its Host for the first time. Upon receipt of a headend reinitialization command, the Card SHALL disable all CA-descrambling, 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 may require the consumer to call the operator to report the Host and Card IDs for validation.

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

## 5.3 Diffie-Hellman Public Key Calculation

### 5.3.1 Diffie-Hellman Overview

The Diffie-Hellman Public Key Agreement procedure provides a method for Card 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 5.3-1 illustrates the two-step Diffie-Hellman operations conducted between the Card and Host.

### 5.3.2 Derivation of the Diffie-Hellman Public Keys

The Card SHALL derive a Diffie-Hellman (DH) public key and its signature as follows: generate a random private exponent,  $x$ , where  $1 \leq x \leq n-2$ ; compute its DH public key as  $DH\_pubKey_C = g^x \bmod n$ ; compute  $SIGN_C$  by signing  $DH\_pubKey_C$  with its X.509 private key.

The Host SHALL derive a Diffie-Hellman (DH) public key and its signature as follows: generate a random private exponent,  $x$ , where  $1 \leq x \leq n-2$ ; compute its DH public key as  $DH\_pubKey_H = g^x \bmod n$ ; compute  $SIGN_H$  by signing  $DH\_pubKey_H$  with its X.509 private key.

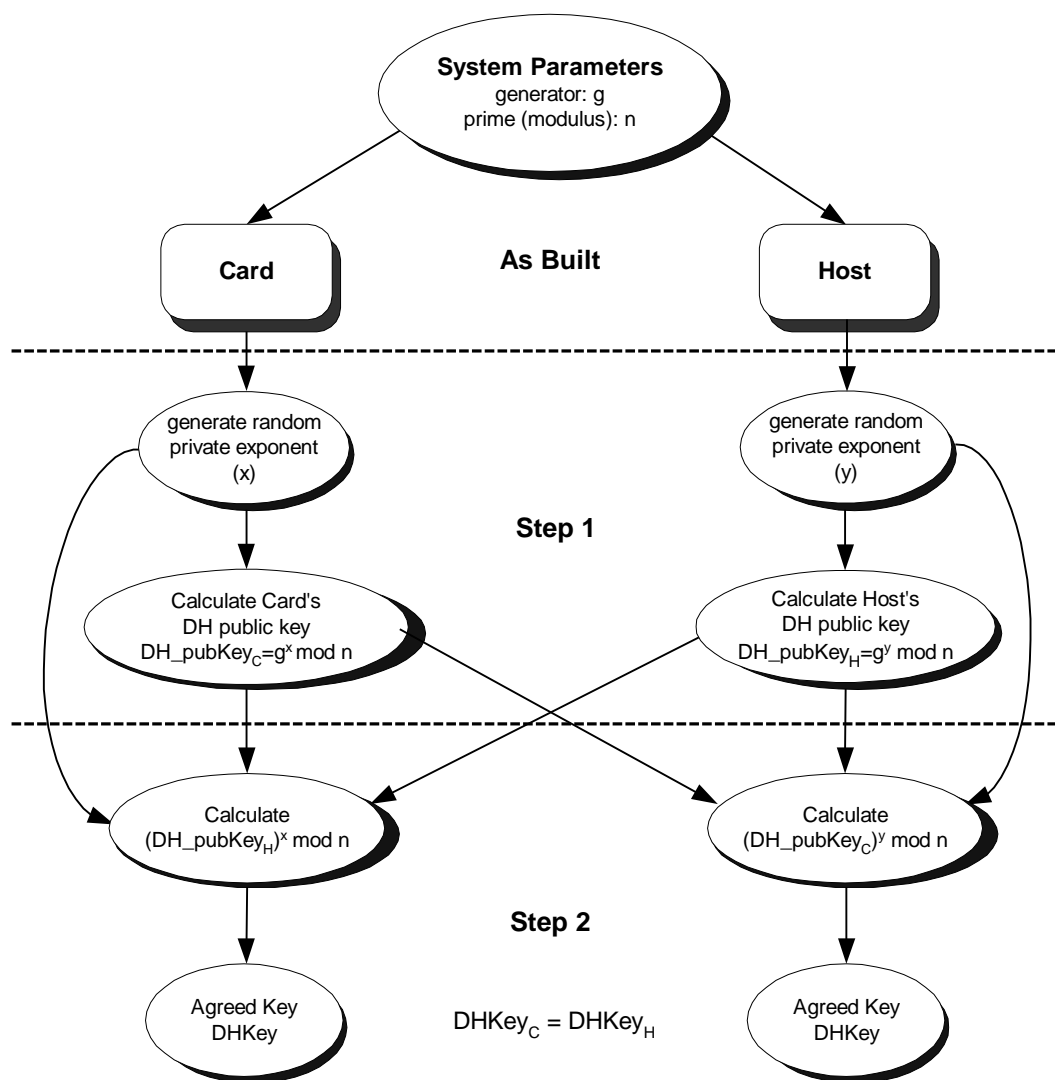


Figure 5.3-1 - Diffie-Hellman Key Agreement

### 5.3.3 Authentication Parameter Exchange and Verification

The Card and Host SHALL exchange and each verify the other's authentication data as follows:

1. The Card SHALL send Card\_DevCert, Card\_ManCert,  $DH\_pubKey_C$ , and  $SIGN_C$  to the Host with a request for the Host's authentication data using the CP\_data\_req() APDU.
2. In response to the CP\_data\_req() APDU requesting authentication data, the Host SHALL reply with the CP\_data\_cnf() APDU containing Host\_DevCert, Host\_ManCert,  $DH\_pubKey_H$  and  $SIGN_H$ .
3. After receiving authentication data from the Host, the Card SHALL verify Host\_DevCert, Host\_ManCert, and  $SIGN_H$  and extract the Host\_ID from the Host Device Certificate.
4. After receiving authentication data from the Card, the Host SHALL verify Card\_DevCert, Card\_ManCert, and  $SIGN_C$  and extract the Card\_ID from the Card Device Certificate.

### 5.3.4 Diffie-Hellman Shared Secret Key

After receiving the  $DH\_pubKey_H$  from the Host, the Card SHALL compute the 1024-bit shared secret key,  $DHKey = (DH\_pubKey_H)^x \bmod n$  and store it in non-volatile memory for calculation of AuthKey and CPKeys.

After receiving  $DH\_pubKey_C$  from the Card, the Host SHALL compute the 1024-bit shared secret key,  $DHKey = (DH\_pubKey_C)^y \bmod n$  and store it in non-volatile memory for calculation of AuthKey and CPKeys.

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

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

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

## 5.4 Shared Authentication Key

After the exchange of mutual authentication data, the Card SHALL compute its authentication key,  $AuthKey_C = SHA-1 [ DHKey | Host\_ID | Card\_ID ]$  and stored it in non-volatile memory.

After the exchange of mutual authentication data, the Host SHALL compute its authentication key,  $AuthKey_H = SHA-1 [ DHKey | Host\_ID | Card\_ID ]$  and store it in non-volatile memory.

## 5.5 Completion of Authentication

The Card and Host SHALL complete the authentication process as follows:

After calculation of  $AuthKey_C$ , the Card SHALL request  $AuthKey_H$  from the Host using the  $CP\_data\_req()$  APDU with  $Datatype\_id = 22$ .

In response to the  $CP\_data\_req()$  APDU with  $Datatype\_id = 22$ , the Host SHALL reply with the  $CP\_data\_cnf()$  APDU containing  $AuthKey_H$ .

After comparing  $AuthKey_H$  to  $AuthKey_C$ , if  $AuthKey_H = AuthKey_C$ , the Card SHALL store AuthKey and authentication status as complete in non-volatile memory, and proceed with CPKey generation.

After comparing  $AuthKey_H$  to  $AuthKey_C$ , if  $AuthKey_H \neq AuthKey_C$ , authentication has failed, and the Card SHALL follow the procedures in Section 5.6.3.

## 5.6 Authentication Failures

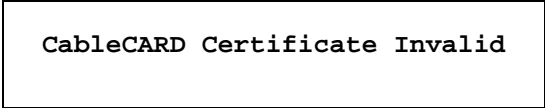
### 5.6.1 Host Response Time-out

When operating in S-Mode, the Card SHALL set the IIR flag if the Host fails to respond to any APDU within five seconds [CCIF].

When operating in M-Mode, the Card SHALL set the ER bit if the Host fails to respond to any APDU within five seconds [CCIF].

### 5.6.2 Invalid Certificate

The Host SHALL display a message informing the user when  $Card\_DevCert$  verification fails, for example:



**CableCARD Certificate Invalid**

**Figure 5.6-1 - Example Card Authentication Failure Notification Message**

The Card SHALL request the copy protection message screen and display a message informing the user when Host\_DevCert verification fails, for example:

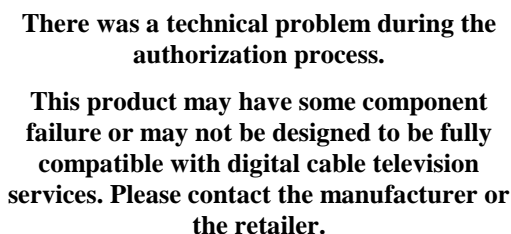


**Host Certificate Invalid**

**Figure 5.6-2 - Example Host Authentication Failure Notification Message**

### **5.6.3 Other Authentication Failures**

The Card SHALL disable all CA descrambling when any other part of the mutual authentication procedure fails, including signature or AuthKey verification, even if the subscriber would otherwise be authorized to receive CA protected services, and notify the headend by automated means (RDC, for example) if possible; open a session to the Host's MMI resource and MMI dialog (if not already open); display a message to the subscriber similar to that shown below in Figure 5.6-3.



**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 5.6-3 - Example CP System Failure Notification Message**

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

## **5.7 Card Operation with Multiple Hosts**

The Card SHALL bind to only one Host at a time and store only one set of Authentication Keys or other Host-specific information.

A Card 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 Card will initiate the binding procedure, including full mutual authentication. If this Card is later returned to the previous Host, it will again initiate the binding procedure, as it has authentication information only on the last Host to which it was bound.

## 6 ID REPORTING, VALIDATION, AND AUTHORIZATION

The Host\_ID and Card\_ID must be reported to the cable operator before the Card will provide non-zero CCI-D content to the Host.

The Card SHALL perform its CA and CP functions following each authentication procedure, if the ID reporting and validation status flag in non-volatile memory indicates that Card-Host binding is complete.

### 6.1 Card and Host Identity Reporting

In cable systems with RDC or other automated reporting functionality, the Host, cable plant, and headend all support compatible connections. This allows the Card to report the Host\_ID and Card\_ID to the headend in an authenticated CA System message.

For one-way cable systems, unidirectional Hosts, or any system without an automated reporting mechanism, the Card and Host IDs must be reported manually, e.g., by the subscriber or installer.

The Card SHALL report identification information, including Card\_ID and Host\_ID, to the headend by one of the automated or manual means specified in Section 6.1.1 or Section 6.1.2.

The Card SHALL store the reported Card\_ID and Host\_ID in non-volatile memory so they can be compared with the validated IDs received back from the headend.

#### 6.1.1 Automated ID Reporting

The Card MAY send the Card\_ID and Host\_ID to the cable headend as a private CA message via any available automated means.

#### 6.1.2 Manual ID Reporting

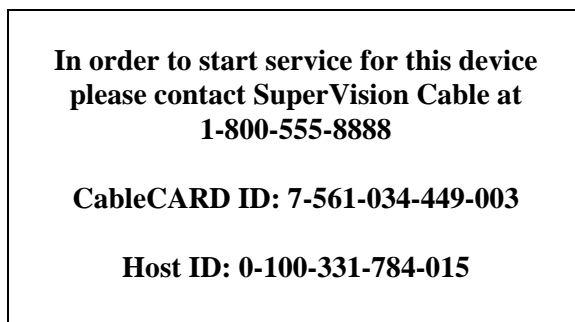
If the IDs are not reported by automated means, the Card SHALL cause the Host to display the “ID Reporting Screen” to the subscriber via the Host MMI resource. The ID Reporting Screen SHALL include the Card\_ID, Host\_ID, a reporting telephone number and any other information required to identify the Card and Host to the CA System.

The Host SHALL display the ID Reporting Screen upon request and confirm to the Card that the message has been displayed to the customer when it grants the MMI dialog open request.

When using the Host MMI resource for the ID Reporting Screen, the Card SHALL cause the Host to display the Host\_ID and 40-lsb's of the Card\_ID as 13-digit decimal numbers with display sequence M-MMU-UUU-UUU-UUL.

- M-MM is the decimal representation of the 10-bit CHICA assigned manufacturer number.
- U-UUU-UUU-UU is the decimal representation of the 30-bit manufacturer assigned unit number.
- L is a Luhn check digit calculated over the preceding 12 decimal digits, as defined in Annex A.
- The ID digit sequence is normative. The hyphens and other text in Figure 6.1-1 are optional.

The Card SHALL cause the Host to display the ID Reporting Screen only if one of these conditions applies: the message is selected through a user menu system; the user selects an MPEG program with CP active (non-zero CCI-D) before the binding is validated; the Card initiates the message display, e.g., at the request of the CA System.



**Figure 6.1-1 - Example ID Reporting Screen**

Additional Card data may be displayed, at the discretion of the Card and CA system vendor.

### **6.1.3 Host Request for ID Reporting Screen**

If the Host requests the ID Reporting Screen, the Card SHALL display the same information and format defined in Section 6.1.2 if authentication is complete, and "information not available" if authentication is not complete.

The Card SHALL support exactly one "ID Reporting application" with application\_type = 0x01, CableCARD Binding Information Application, defined in [CCIF], to support a Host request to display the ID Reporting Screen.

The Host SHALL NOT use the ID Reporting application for any reason except display of the ID Reporting Screen.

## **6.2 Headend Validation**

### **6.2.1 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 will examine all previously reported Host\_IDs, and, if there are any matches, it will notify the cable operator.

### **6.2.2 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 Card in a private authenticated CA System ID validation message.

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

The Card SHALL authenticate CA System ID validation messages and accept only authentic messages.

- If both the Validated\_Card\_ID and Validated\_Host\_ID values match its current stored Card\_ID and Host\_ID, the Card SHALL set validation status to complete and enable CA-descrambling of authorized non-zero CCI-D content.

- If one or both validated IDs do not match, the Card SHALL disable CA-descrambling of non-zero CCI-D content.

### **6.3 CA Authorization of Copy Protected Content**

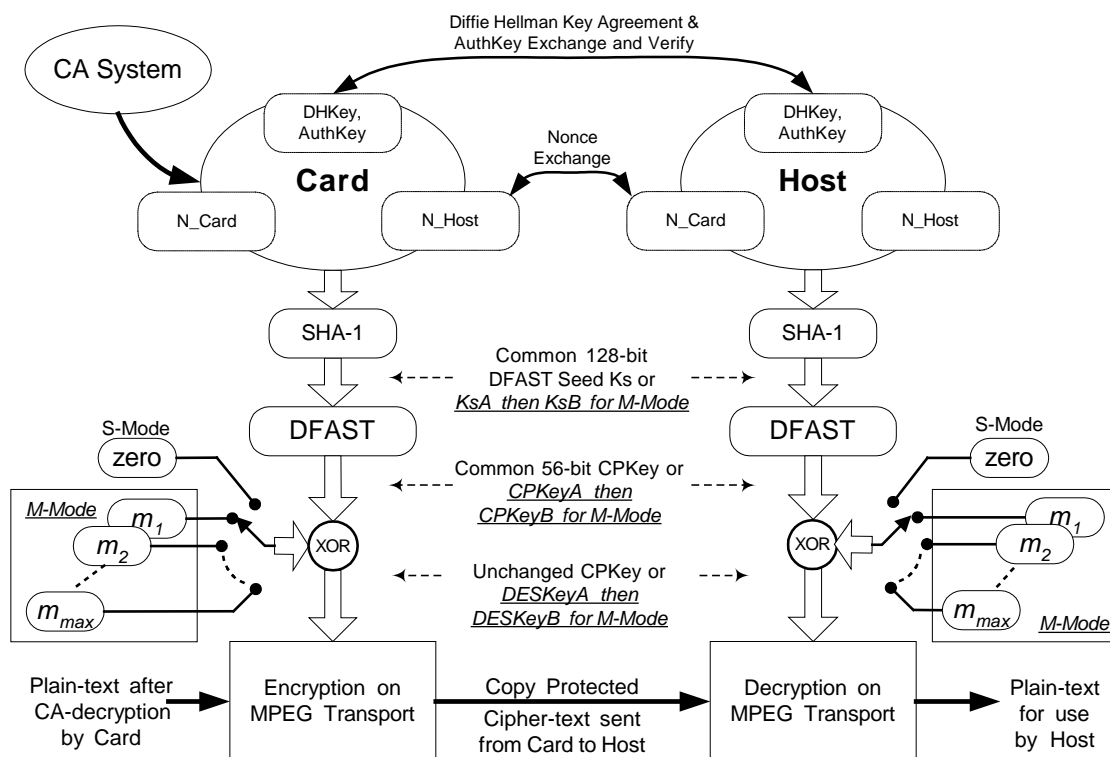
The cable operator authorizes the Card for services and content through its conditional access system. The CA System will deny authorization for specific protected content to any Card or Host whose CP security is considered insufficient for reception of that content.

The headend always has an opportunity to revoke content or services using CA System Entitlement Management Messages. 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.

## 7 CRYPTOGRAPHIC FUNCTIONS

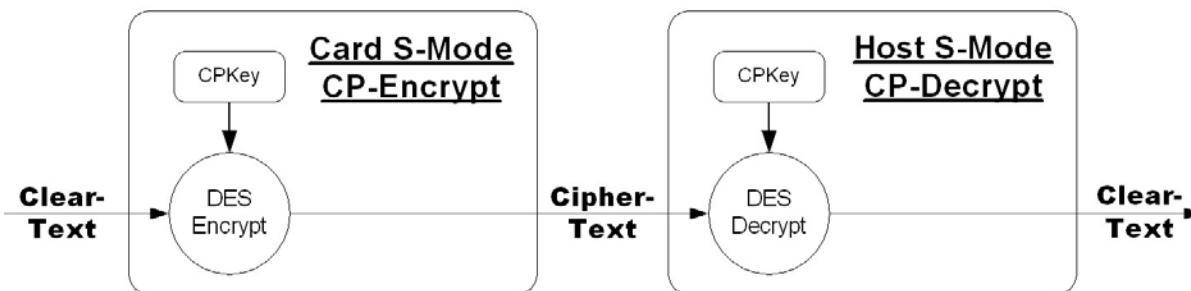
The basic copy protection key generation process is shown in Figure 7-1 below. The Card and Host generate a common DFAST seed, Ks, and root key, CPKey, from shared secrets and exchanged nonces.

S-Mode uses single DES, and Ks and CPKey are single values. M-Mode uses ABA triple DES, and Ks, CPKey, and DESKey are each a key pair, formed by running the same generation process twice using two different truncations of DHKey as described in Section 8.2. Unique DESKey-pairs are then generated for each MPEG-program by XOR'ing the CPKey-pair with the unique MPEG-program ID values 'm' as described in Section 10.4.2.



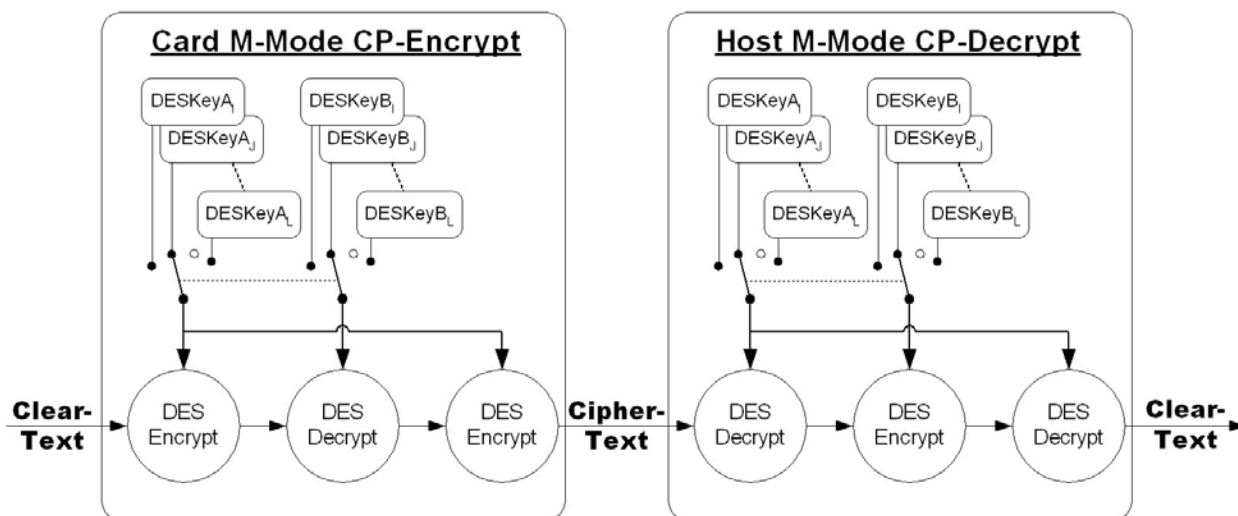
**Figure 7-1 - Encryption Key Generation**

The basic encryption-decryption methods are shown in Figure 7-2 and Figure 7-3 below.



**Figure 7-2 - S-Mode Encryption**





**Figure 7-3 - M-Mode Encryption**

## 7.1 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 7-1. Detailed information on DFAST design and implementation is obtained from the CHICA.

The DFAST function accepts a 128-bit input value ( $K_s$ ) and generates 56 bits of output (CPKey). For M-Mode, this output from the DFAST function is XOR'd with the value 'm'. The result of the XOR is  $\text{DESKeyA}_m$  and  $\text{DESKeyB}_m$  for the M-Mode DES-ECB encryption/decryption keys.

## 7.2 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 Card SHALL implement either a physical random integer generator compliant with [FIPS 140-2], section 4.7.1, or a pseudorandom (software) integer generator compliant with the SHA-1 based algorithm described in [FIPS 186-2], Appendix 3, Section 3.3, and including a unit-unique secret seed.

The Host SHALL implement either a physical random integer generator compliant with [FIPS 140-2], section 4.7.1, or a pseudorandom (software) integer generator compliant with the SHA-1 based algorithm described in [FIPS 186-2], Appendix 3, Section 3.3, and including a unit-unique secret seed.

## 7.3 SHA-1 Secure Hash Algorithm

The RSA signature algorithm is employed with SHA-1 for all X.509 digital certificates.

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

- **Host Device Certificate Signature Verification:** the signature algorithm is based on the RSA digital signature scheme defined in [RSA1], which uses the SHA-1 primitive.

- Card Device Certificate Signature Verification: the signature algorithm is based on the RSA digital signature scheme defined in [RSA1], which uses the SHA-1 primitive.
- Authentication Key generation as described in Section 5.4.
- Copy Protection Key generation, as described in Section 8.

## 7.4 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:

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

## 7.5 RSA Digital Signatures

The Card SHALL compute RSA digital signatures using RSASSA-PKCS1-v1\_5 as specified in [RSA1] with SHA-1 as the hash function.

The Host SHALL compute RSA digital signatures using RSASSA-PKCS1-v1\_5 as specified in [RSA1] with SHA-1 as the hash function.

The Card SHALL possess a Device Certificate and Manufacturer Certificate signed with RSA keys having a modulus length of 2048 bits.

The Host SHALL possess a Device Certificate and Manufacturer Certificate signed with RSA keys having a modulus length of 2048 bits.

Note: The RSA public key exponent “e” has value 65537 decimal (0x01 0001).

## 8 COPY PROTECTION KEY GENERATION

The Card initiates CPKey generation at the following times:

- After completion of the authentication 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 (PCMCIA) reset.

Channel change does not cause a key refresh to occur.

The Card and Host derive a common Copy Protection Key from random integers unique to each calculation: the shared Authentication Key (AuthKey) and the 1024-bit shared secret Diffie-Hellman key (DHKey). The derived common Copy Protection Key (CPKey) is then used to generate a unique DESKey for each MPEG program as described in Section 10.4. The DES-ECB processor uses these DESKeys to CP-encrypt or decrypt content in MPEG-TS packets sent from the Card to the Host.

### 8.1 Basic Key Generation Protocol

The Card uses the time period variable, T, to allow more time for CPKey calculation when operating in M-Mode.

The Card SHALL set the CPKey calculation time period variable, T=10 when operating in S-Mode.

The Card SHALL set the CPKey calculation time period variable, T=30 when operating in M-Mode.

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

- The Card SHALL check for a valid AuthKey and, if such an AuthKey is not present, then follow the whole authentication process as detailed in Section 5.2.
- The Card SHALL generate a new 64-bit random integer, N\_Card, per Section 7.2.
- The Card SHALL send N\_Card and Card\_ID in the clear to the Host using the protocol in Table 11.5-2 and immediately set a Key Refresh Timer to zero.
- The Host SHALL generate a new 64-bit random integer, N\_Host, per Section 7.2.
- The Host SHALL send N\_Host and Host\_ID in the clear to the Card using the protocol in Table 11.5-3 and compute CPKey as described in Section 8.2.
- The Card SHALL disable CA-descrambling of non-zero CCI-D content if the received Host\_ID is not equal to the previously stored Host\_ID.
- The Card SHALL compute CPKey, as described in the Section 8.2.
- When the Key Refresh Timer reaches T-1 seconds, the Card SHALL stop CP-encryption and issue CP\_sync\_req() using the protocol in Table 11.6-2.
- The Host SHALL reply with a CP\_data\_cnf() APDU within one second of receiving CP\_data\_req() and begin using with the newly-calculated CPKey for CP-encryption.
- If Key Refresh Timer reaches T seconds before receiving CP\_sync\_cnf(), the Card SHALL disable CA-descrambling until reception of CP\_sync\_cnf().

- Upon reception of CP\_syn\_cnf(), the Card SHALL restart CP-encryption using the newly calculated CPKey and set CP Session Timer to zero.

## 8.2 Copy Protection Key (CPKey) Calculation

The Card and Host compute copy protection keys in the same manner. First they calculate a DFAST seed value, Ks.

$$Ks = \text{SHA-1} [\text{AuthKey} \mid \text{DHKey} \mid N\_Host \mid N\_Card]_{\text{msb128}}$$

In M-Mode, Ks has two components, KsA and KsB (unique due to different DHKey truncation).

$$KsA = \text{SHA-1} [\text{AuthKey} \mid \text{DHKey}_{\text{lsb512}} \mid N\_Host \mid N\_Card]_{\text{msb128}}$$

$$KsB = \text{SHA-1} [\text{AuthKey} \mid \text{DHKey}_{\text{msb512}} \mid N\_Host \mid N\_Card]_{\text{msb128}}$$

The 160-bit SHA-1 output is truncated to its 128 msb, left-most bits, to generate a seed, Ks, with the proper length for input to the DFAST engine.

The Card and Host then apply the DFAST function to Ks to form CPKey, the copy protection key, which also has two components in M-Mode.

$$\text{In S-Mode: CPKey} = \text{DFAST} [Ks]$$

$$\text{In M-Mode: CPKeyA} = \text{DFAST} [KsA], \text{ and}$$

$$\text{CPKeyB} = \text{DFAST} [KsB]$$

DFAST details are specified in a separate document; contact the CHICA.

**Table 8.2-1 - Length of Keys and Parameters Used in the Key Generation**

Parameter	Size (bits)	Description
AuthKey	160 bits	Shared Authentication key, see Section 5.4.
DHKey	1024 bits	Shared Secret Diffie-Hellman key, see Section 5.3.4.
N_Host, N_Card (nonces)	64 bits each	Random integers unique to each calculation of CPKey.
Ks (DFAST seed)	S-Mode: 128 bits M-Mode: two 128-bit values	Intermediate key for input to DFAST engine. In M-Mode, the two components are denoted KsA and KsB.
CPKey (Copy Protection Key)	S-Mode: 56 bits M-Mode: two 56-bit values	Root for the CP encryption and decryption key. In M-Mode, the two components are denoted CPKeyA and CPKeyB.
DESKey	S-Mode: 56 bits M-Mode: two 56-bit values	The final CP encryption and decryption key. In M-Mode, the two components are denoted DESKeyA and DESKeyB.
max_key_session_period	16 bits	Sets the maximum time a CPKey may be used. In S-Mode, units are decaseconds (10s). In M-Mode, units are minutes (60s).

### 8.3 CP Session Timer

The Card periodically initiates refresh of the CPKey. The CA System will set the refresh period with a parameter, `max_key_session_period`, transmitted to the Card by the CA System with maximum security.

The key session period is the length of time the Card and Host use a given CPKey. The Card sets a Key Session Timer to zero each time a new CPKey is generated. This timer is not dependent on the MPEG program selected by the Host.

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`.

When operating in S-Mode, the Card SHALL initiate CPKey generation when CP Session Timer reaches 10 seconds times the value of `max_key_session_period`, unless `max_key_session_period` = 0.

When operating in M-Mode, the Card SHALL initiate CPKey generation when CP Session Timer reaches 60 seconds times the value of `max_key_session_period`, unless `max_key_session_period` = 0.

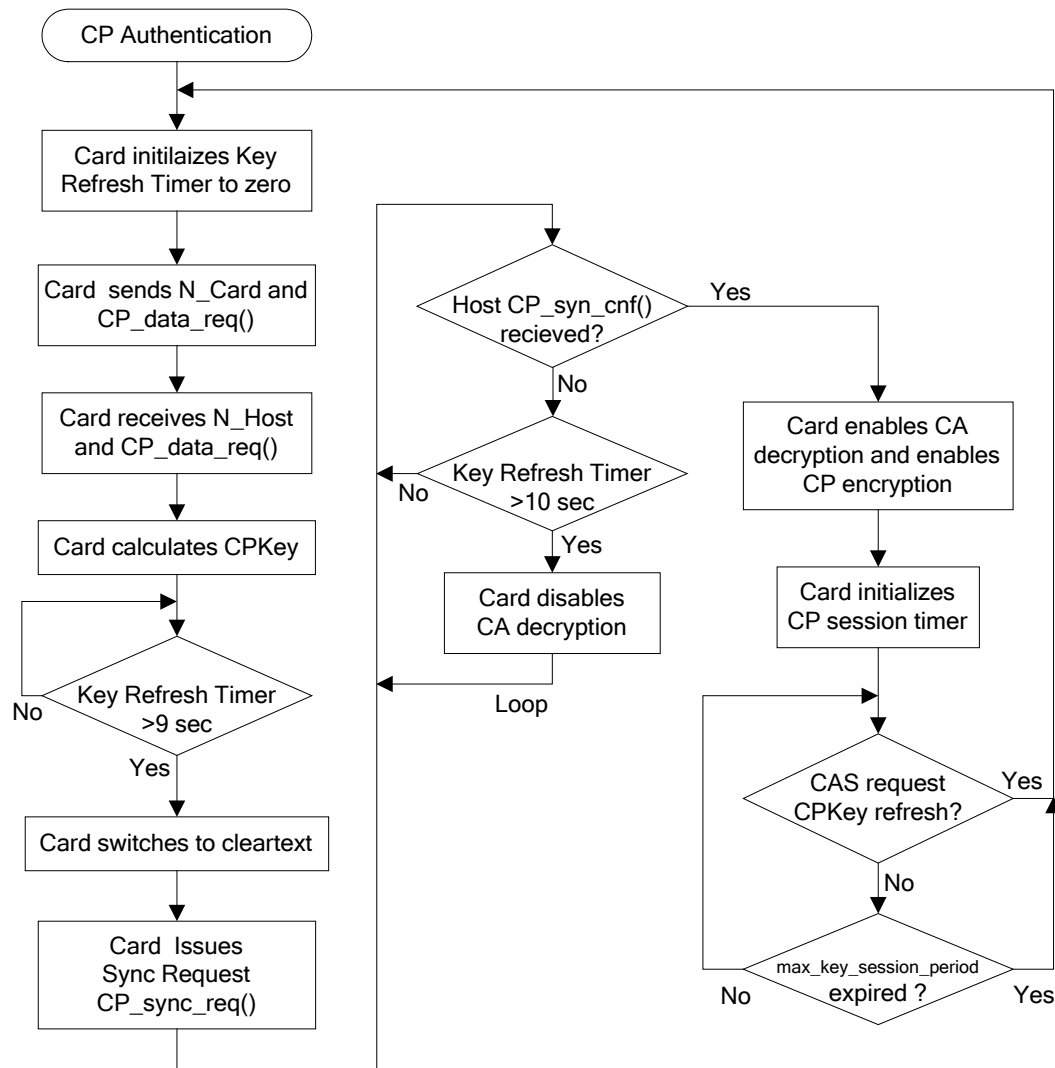
### 8.4 CPKey Generation Flow

The Card starts a Key Refresh Timer when it sends `N_Card` to the Host in the `CP_data_req()` APDU. When the Host receives the `CP_data_req()` APDU, the Host generates `N_Host` and sends it to the Card in the `CP_data_cnf()` APDU. The Host will reply with a `CP_data_cnf()` APDU within one second of receiving a `CP_data_req()` APDU.

When the Host issues the `CP_data_cnf()` APDU, the Card and Host start the calculation of the Copy Protection Key. The Card and Host each calculate CPKey within eight seconds when operating in S-Mode, and within 28 seconds when operating in M-Mode. When the Key Refresh Timer reaches nine seconds when operating in S-Mode, or 29 seconds when operating in M-Mode, the Card sends the `CP_sync_req()` APDU to the Host. This timing ensures that both the Card and Host have a minimum of eight seconds for S-Mode, and 28 seconds for M-Mode to complete key calculation. The Card `CP_sync_req()` APDU indicates that the Card has completed calculation of CPKey(s). The Host issues the `CP_sync_cnf()` APDU when it has received the `CP_sync_req()` APDU and has completed calculation of CPKey.

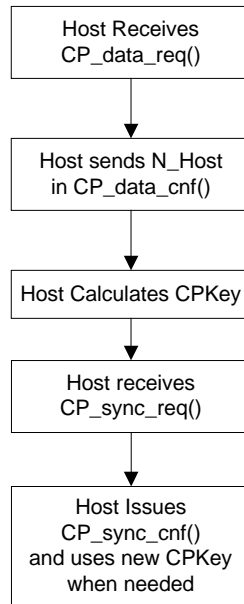
When the Card issues the `CP_sync_req()` APDU, the Card turns off all CP-encryption and sets the MPEG `transport_scrambling_control` field to 00. The Host receives in-the-clear data packets and will recognize these packets as unencrypted according to MPEG rules. If the key refresh has not completed when the Key Refresh Timer reaches ten seconds when operating in S-Mode, or 30 seconds when operating in M-Mode, the Card will disable CA-descrambling of all copy protected content (including other copy protected services on the interface) until the current or retry CPKey generation completes (the Host has responded with a `CP_sync_cnf()` APDU). The Card begins encrypting MPEG packets with the new CP Key at the completion of the CP\_Key refresh.

Figure 8.4-1 illustrates the Card actions during a CPKey generation.



**Figure 8.4-1 - Card CPKey Session Flow Chart for S-Mode**

Figure 8.4-2 illustrates Host actions during CPKey generation.



**Figure 8.4-2 - Host CPKey Session Flow Chart**

## 8.5 CA Initiated CPKey Refresh

The Card SHALL immediately initiate CPKey generation upon CA System command unless Mutual Authentication is incomplete or a CPKey refresh is currently in process.

## 9 COPY CONTROL INFORMATION (CCI)

The cable operator defines blocks of content with an assigned CCI value. The CA System will CA-scramble all content marked with a non-zero CCI value. Content marked with a zero CCI value (0x00) may be CA-scrambled or CA-unscrambled. The CA System will inform the Card of the CCI value of all CA-scrambled content.

No restrictions herein described apply to content delivered by the cable system in analog or CA-unscrambled form.

### 9.1 CCI Definition

CCI is an 8-bit field conveyed from Card to Host. Six of the eight bits are defined and the remaining two are reserved. The Card SHALL set reserved bits in the CCI byte to zero as shown in Table 9.1-1.

The Host SHALL use the reserved bit values in the CCI byte received from the Card for execution of the protocol described in Section 9.4.2 and ignore the reserved (rsvd) bit values thereafter.

**Table 9.1-1 - CCI Bit Assignments**

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

CCI-A is the subset of CCI consisting of the APS1, APS0, and CIT bits, which control protection on analog outputs. "Zero CCI-A" means all three bits have value zero. Non-Zero CCI-A means one or more bits are value one.

CCI-D is the subset of CCI consisting of the RCT, EMI1, and EMI0 bits, which control protection on digital outputs. "Zero CCI-D" means all three bits have value zero. Non-Zero CCI-D means one or more bits are value one.

#### 9.1.1 EMI - Digital Copy Control Bits

Bits 0 and 1 of the CCI byte are the EMI bits. The Host SHALL use the EMI bits contained in the CCI to control copy permissions for digital outputs as defined in Table 9.1-2 and supply the EMI bits to any Host digital output ports for control of copies made from those outputs. The EMI bits are defined in Table 9.1-2.

**Table 9.1-2 - EMI Values and Copy Permissions**

EMI Value	Digital Copy Permission
00b	Copying not restricted
01b	No further copying is permitted
10b	One generation copy is permitted
11b	Copying is prohibited

#### 9.1.2 APS - Analog Protection System

Bits 3 and 2 of CCI as shown in Table 9.1-1 are the APS bits 1 and 0 respectively. The Host SHALL use the APS bits contained in the CCI byte to control copy protection encoding of analog outputs as described in Table 9.1-3.



**Table 9.1-3 - APS Value Definitions**

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

### 9.1.3 CIT – Constrained Image Trigger

The Host SHALL control Image Constraint of high definition analog component outputs according to the CIT value in the CCI byte as shown in Table 9.1-4.

**Table 9.1-4 - CIT Value Definitions**

<b>CIT</b>	<b>Description</b>
0b	No Image Constraint asserted
1b	Image Constraint required

### 9.1.4 RCT – Redistribution Control Trigger

The Host SHALL activate redistribution control when RCT=1, as shown in Table 9.1-5.

The Host SHALL activate EPN Mode with EPN bit value of zero on DTCP outputs for content marked with EMI=00b when RCT=1.

Table 9.1-6 illustrates the dependence of RCT on EMI value for DTCP outputs.

**Table 9.1-5 - RCT Value Definitions**

<b>RCT</b>	<b>Description</b>
0b	No Redistribution Control asserted
1b	Redistribution Control required

**Table 9.1-6 - Host DTCP Outputs**

<b>Card CCI</b>		<b>DTCP Outputs</b>	
<b>EMI</b>	<b>RCT</b>	<b>Protection</b>	<b>Encryption</b>
00b	0	Not Required	Not Required
00b	1	EPN Mode	Yes
01b	0 or 1	Copy No More	Yes
10b	0 or 1	Copy Once	Yes
11b	0 or 1	Copy Never	Yes

## 9.2 Associating CCI with a Service

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

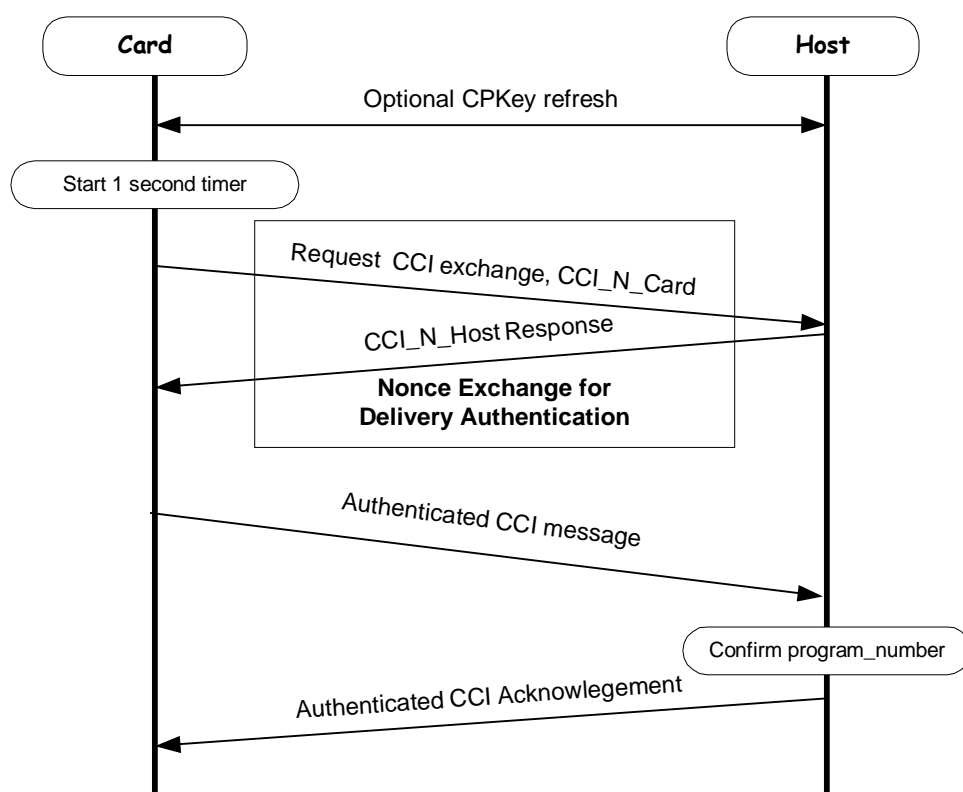
## 9.3 Conveying CCI from Headend to Card

The CA System will provide a private secure delivery methods (e.g., an eCM) to transfer CCI from the headend to the Card. This delivery method will preserve the association between CCI and the specific MPEG program and LTSID.

## 9.4 Conveying CCI from Card to Host

Delivery of CCI from Card to Host is authenticated via the exchange of messages as shown in Figure 9.4-1. The messages are based on a SHA-1 function performed on the CCI, CPKey, MPEG program number, and for M-Mode, also ECM-PID and LTSID.

The sequence is repeated for each MPEG program currently being descrambled by the Card. Each CCI exchange should be completed before the next one begins.



**Figure 9.4-1 - CCI Delivery Sequence**

### 9.4.1 CCI Delivery Instances

The Card SHALL NOT send CCI to the Host unless the Card and Host have successfully completed Authentication and ID Validation, and negotiated a shared CPKey.

The Card SHALL initiate CCI delivery for every MPEG program with non-zero CCI immediately after the Host sends a ca\_pmt() APDU with the ok\_descrambling command for that MPEG program unless the Card rejects the command with an error indication.

The Card SHALL initiate CCI delivery immediately after every change in any MPEG program that the Card is CA-descrambling or its associated CCI.

The Card SHALL NOT deliver CCI for MPEG programs unless it has received a ca\_pmt() APDU with the ok\_descrambling command.

The Card SHALL NOT CA-descramble any MPEG program until it has received a ca\_pmt() APDU with the ok\_descrambling command.

### 9.4.2 CCI Delivery Protocol

The Card SHALL disable CA-descrambling of the associated MPEG program following any error or failure of the CCI Delivery Protocol until the error is cleared by a successful completion of the CCI Delivery Protocol.

The Card SHALL delay initiating the CCI Delivery Protocol until CPKey refresh is complete when the Key Refresh Timer is greater than T-2 seconds. See informative Figure 9.4-2.

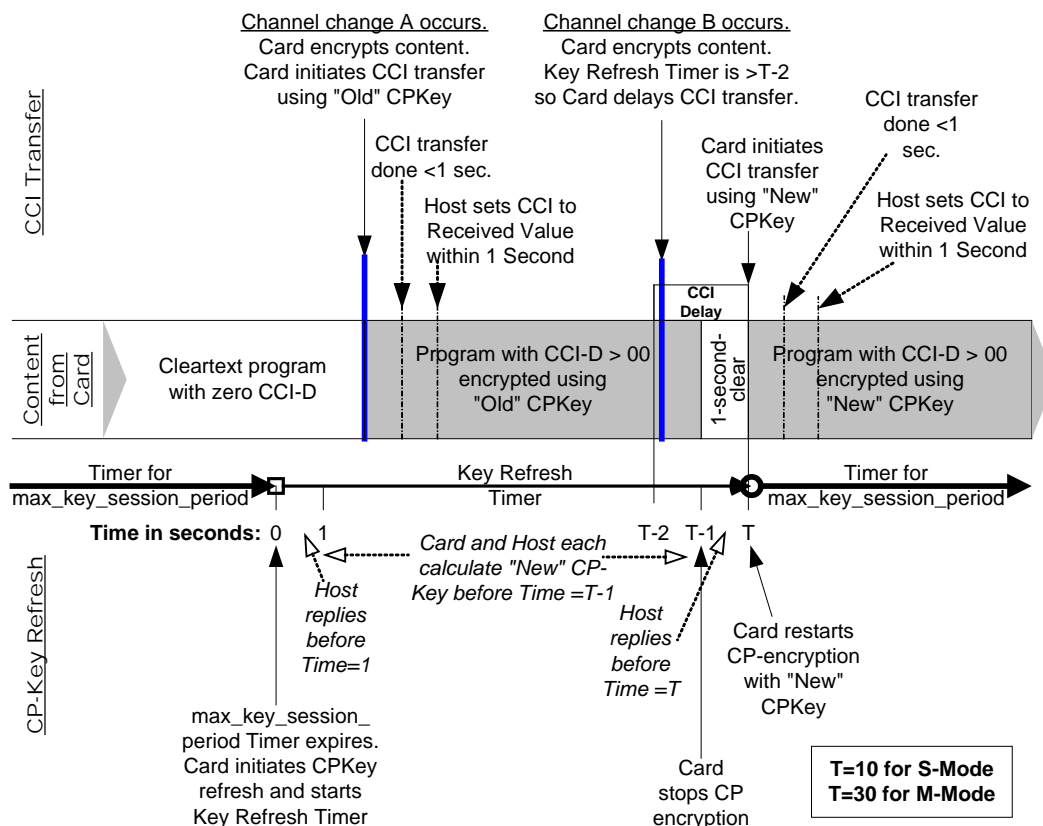
When operating in S-Mode, the Card and Host SHALL execute the CCI Delivery Protocol as follows:

- The Card SHALL generate a new random integer CCI\_N\_Card and start a 1-second time-out.
- The Card SHALL send CCI\_N\_Card, program\_number, and a request for CCI\_N\_Host.
- The Host SHALL generate a new random integer CCI\_N\_Host and reply with CCI\_N\_Host and program\_number (received in the step above).
- The Card SHALL calculate two values: CCI\_auth to authenticate CCI delivery, and CCI\_ack to authenticate Host acknowledgment of receipt, as: CCI\_auth = SHA-1(CCI | CPKey | CCI\_N\_Card | CCI\_N\_Host | program\_number ), and CCI\_ack = SHA-1(CCI | CPKey | CCI\_N\_Card | CCI\_N\_Host).
- The Card SHALL send CCI\_auth, CCI, and program\_number to the Host.
- The Host SHALL calculate CCI\_auth using the received CCI value.
- If the Host's calculated CCI-auth value does not match the CCI\_auth value received from the Card, it SHALL treat the content as if a CCI value = 0x03 was received and return a zero value CCI\_ack to the Card to indicate the failure.
- If the Host's calculated CCI-auth value matches the CCI\_auth value received from the Card, it SHALL control output of the associated MPEG program according to valid CCI within one second.
- The Host SHALL calculate CCI\_ack and send it to the Card.
- The Card SHALL compare the received CCI\_ack with its calculated value.
- If the CCI\_ack values match, the Card SHALL begin delivery of the associated MPEG program as indicated by the CCI. If the CCI\_ack values do not match, the Card SHALL disable CA-descrambling of the associated MPEG program until CCI delivery completes successfully with a CCI\_ack match.
- If the steps above are not completed before the one-second time-out expires, the Card SHALL disable CA-descrambling of copy protected content for the associated MPEG program until the CCI delivery protocol completes successfully.

When operating in M-mode, the Card and Host SHALL execute the CCI Delivery Protocol as follows:

- The Card SHALL generate a new random integer CCI\_N\_Card and start a 1-second time-out.

- The Card SHALL send CCI\_N\_Card, program\_number, LTSID, and a request for CCI\_N\_Host.
- The Host SHALL generate a new random integer CCI\_N\_Host and reply with CCI\_N\_Host, program\_number and LTSID (received in the step above).
- The Card SHALL calculate two values: CCI\_auth to authenticate CCI delivery, and CCI\_ack to authenticate Host acknowledgment of receipt, as:  $CCI\_auth = SHA-1( CCI \parallel CPKeyA \parallel CPKeyB \parallel CCI\_N\_Card \parallel CCI\_N\_Host \parallel program\_number \parallel LTSID \parallel 000b \parallel ECM-PID )$ , and  $CCI\_ack = SHA-1( CPKeyA \parallel CPKeyB \parallel CCI \parallel CCI\_N\_Card \parallel CCI\_N\_Host \parallel program\_number \parallel LTSID \parallel 000b \parallel ECM-PID )$ .
- The Card SHALL use the same ECM-PID value that was used to generate DESKey for the MPEG program, as defined in Section 10.4.2.
- If the Card chooses to send CCI for a program with no ECM-PID in the ca\_pmt() APDU, the Card SHALL use a default ECM-PID value of 0 0000 0000 000b in calculating CCI\_auth and CCI\_ack.
- The Card SHALL send CCI\_auth, CCI, program\_number, and LTSID to the Host.
- The Host SHALL calculate CCI\_auth using the received CCI value.
- If the Host's calculated CCI-auth value does not match the CCI\_auth value received from the Card, it SHALL treat the content as if a CCI value = 0x03 was received and return a zero value CCI\_ack to the Card to indicate the failure.
- If the Host's calculated CCI-auth value matches the CCI\_auth value received from the Card, the Host SHALL control output of the associated MPEG program according to valid CCI within one second.
- The Host SHALL calculate CCI\_ack and send it to the Card.
- The Card SHALL compare the received CCI\_ack with its calculated value.
- If the CCI\_ack values match, the Card SHALL begin delivery of the associated MPEG program as indicated by the CCI.
- If the CCI\_ack values do not match, the Card SHALL disable CA-descrambling of the associated MPEG program until CCI delivery completes successfully with a CCI\_ack match.
- If the steps above are not completed before the one-second time-out expires, the Card SHALL disable CA-descrambling of copy protected content for the associated MPEG program until the CCI Delivery Protocol completes successfully.
- If the Host supports M-Mode Class 176 Type 4 Version 2 and the Card opens M-Mode Class 176 Type 4 Version 2, the Host SHALL treat the Card's message digest (CCI\_auth) as not-match and respond appropriately for a message digest not-match.



**Figure 9.4-2 - Two Examples of CCI Transfer During CPKey Refresh**

### 9.4.3 Host Application of CCI

The Host SHALL set copy control parameters on its outputs according to the delivered CCI for all content CA-descrambled and CP-encrypted by the Card. The Host SHALL retain the temporal association of CCI with content to within two seconds.

The Host SHALL control output of content according to the originally associated CCI value, independent of any other action of the Host, including, but not limited to, recording and delayed playback or output of content, Host "power-off" while the Card remains powered, or Host tuning away to analog or clear digital channels and then back to content with non-zero CCI.

## 10 TRANSPORT ENCRYPTION FROM CARD TO HOST

The Card applies DES ECB based encryption to protect certain content as it passes from Card to Host. In S-Mode, CP-encryption is single DES. In M-Mode, CP-encryption is ABA Triple DES.

### 10.1 CP-Encryption as a Function of CA-Scrambling and CCI Value

When the Card is CA-authorized and fully bound to the Host, the Card will apply CP-encryption as shown in Table 10.1-1, where 'X' means "don't care", that is, CP-encryption is independent of the value.

**Table 10.1-1 - CP-Encryption Based on CA-Scrambling and CCI Value**

CA-Scrambling State	CCI-A Value	CCI-D Value	Card applies CP-Encryption	Comments
In-the-clear	Zero or none		No	
In-the-clear	Non-Zero		No	Undesired*
Scrambled	X	Zero	Permitted but Not Required	Card's Option
Scrambled	X	Non-zero	Yes	

\* Cable operators should CA-scramble all MPEG programs with non-zero CCI. Only CA-scrambled MPEG programs will be protected from unauthorized use.

### 10.2 CP-Encryption Rules

- a) When operating in S-Mode, the Card SHALL use single DES ECB for all CP-encryption and leave any residual blocks less than 64 bits in size in the clear.
- b) When operating in M-Mode, the Card SHALL use ABA Triple DES ECB for all CP-encryption and handle any residual blocks as defined in Annex B.
- c) The Card SHALL encrypt only the payload portion of MPEG transport stream packets.
- d) The Card SHALL NOT encrypt LTSID packet preheaders, MPEG packet headers, or MPEG adaptation headers.
- e) The Card SHALL set the transport\_scrambling\_control bits as described in Table 10.2-1.
- f) The Card SHALL apply Pass-through mode for CA-scrambled but unauthorized services and CA-scrambled and authorized but unselected services, rendering these services useless to the Host.
- g) The Card SHALL CP-encrypt every MPEG program with non-zero CCI-D that the Card CA-descrambles (Rescramble mode).
- h) The Card MAY, at its option, either CP-encrypt (Rescramble) or return in-the-clear (CA-Only) any MPEG programs with zero CCI-D that the Card has CA-descrambled.
- i) The Card SHALL immediately switch from Rescramble mode to Pass-through mode when the active MPEG program is de-authorized by the CA System.

- j) The Card SHALL NOT apply CP-encryption to data that remains CA-scrambled.
- k) The Card SHALL CP-encrypt only CA-authorized and selected MPEG programs.

### 10.2.1 Transport Scrambling Control Field

The transport\_scrambling\_control field of the MPEG transport packet provides control information for key changes.

**Table 10.2-1 - MPEG transport\_scrambling\_control Values**

Bit Values	CPKey Mode
00b	No scrambling of TS packet payload
01b	Reserved
10b	Reserved
11b	Transport packet payload scrambled

## 10.3 Timing of Encryption Mode Transitions

The Card SHALL accomplish CP-encryption mode changes in no more than 1.5 seconds after the event that causes the mode change.

The Card SHALL CP-encrypt all packet payloads of the relevant MPEG program within 1.5 seconds following a change from zero CCI-D to non-zero CCI-D. CA System events, such as encryption management or control messages or changes in the CA-scrambling mode, may affect CA-descrambling by the Card.

The Card SHALL continue to comply with all CP-encryption requirements while responding to any CA System messages or CA-scrambling mode changes.

## 10.4 DESKeys for Single and Multi-stream Modes

### 10.4.1 DESKey for Single Stream Mode

When operating in S-Mode, the Card SHALL encrypt the single selected MPEG program with  $\text{DESKey}_0 = \text{CPKey}$ .

### 10.4.2 DESKey for Multi-Stream Mode

When operating in M-Mode, the Card SHALL encrypt each copy protected MPEG program using an individual  $\text{DESKey}_m$ , consisting of two components,  $\text{DESKeyA}_m$  and  $\text{DESKeyB}_m$ , where:

- $\text{DESKeyA}_m = \text{CPKeyA} \oplus m$
- $\text{DESKeyB}_m = \text{CPKeyB} \oplus m$
- $m = \text{ECM-PID} \mid \text{LTSID}$ , padded with 35 left-most (MSB) zero bits to form a 56-bit value
- ECM-PID is the first ECM-PID value appearing in the ca\_pmt() APDU (see [CCIF]) at ES level for that MPEG program, or if no ECM-PID exists at ES level, then the ECM-PID for the program.

The Card SHALL disable CA-decryption for any programs with no ECM-PID at either the ES or program level.

The Host SHALL use an ECM-PID value of zero for any program with no ECM-PID value at ES or program level.

When operating in M-Mode, the Card SHALL use the computed DESKey<sub>m</sub> for CP-encryption immediately upon receipt of the CP\_sync\_cnf() APDU.

When operating in M-Mode, the Host SHALL use the computed DESKey<sub>m</sub> for CP-decryption immediately after transmitting the CP\_sync\_cnf() APDU.

When operating in M-Mode, the Card SHALL immediately apply any changes in ECM-PID or LTSID to the value DESKey<sub>m</sub> used to CP-encrypt the associated MPEG transport stream packets.

When operating in M-Mode, the Host SHALL immediately apply any changes in ECM-PID or LTSID to the value of DESKey<sub>m</sub> used to CP-decrypt the associated MPEG transport stream packets.



## 11 CARD-HOST MESSAGING PROTOCOLS

### 11.1 Message Protocol Overview

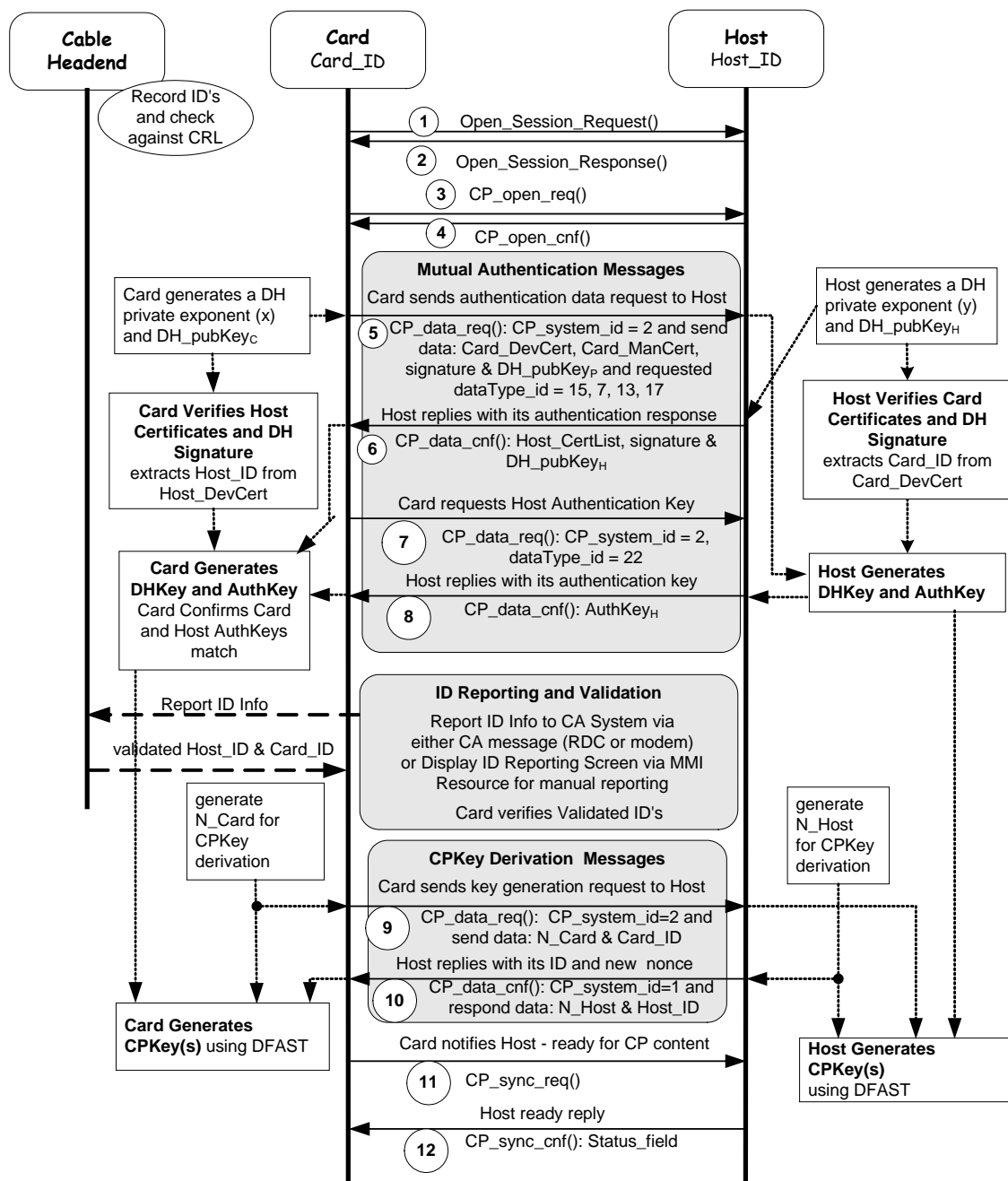


Figure 11.1-1 - Card-Host Message Protocol Flow

Table 11.1-1 gives an overview of the CP System message flow starting with initial Card-Host binding. After the Card and Host binding is authenticated or validated, only parts of the illustrated sequence will be repeated, as described in Sections 5 and 6, for example, following power-ups or for CPKey refresh.

**Table 11.1-1 - Message Reference Sections**

#	Message Name	Protocol Layer / Tag Value (hex)	Reference Section	Purpose
1	open_session_request	SPDU / 0x91	Section 11.3	Open CP session
2	open_session_response	SPDU / 0x92		
3	CP_open_req	APDU / 0x9F 9000	Section 11.3.1	Evaluate Host support for CP
4	CP_open_cnf	APDU / 0x9F 9001		
5	CP_data_req	APDU / 0x9F 9002	Section 11.4.1	Card & Host authentication data
6	CP_data_cnf	APDU / 0x9F 9003		
7	CP_data_req	APDU / 0x9F 9002	Section 11.4.2	Authentication Key verification
8	CP_data_cnf	APDU / 0x9F 9003		
9	CP_data_req	APDU / 0x9F 9002	Section 11.5	CPKey generation
10	CP_data_cnf	APDU / 0x9F 9003		
11	CP_sync_req	APDU / 0x9F 9004	Section 11.6	Card-Host CPKey Synchronization
12	CP_sync_cnf	APDU / 0x9F 9005		
13	CP_data_req	APDU / 0x9F 9002	Section 11.7	CCI Delivery Protocol
14	CP_data_cnf	APDU / 0x9F 9003		
15	CP_valid_req	APDU / 0x9F 9006	Section 11.8	Card Validation Status
16	CP_valid_cnf	APDU / 0x9F 9007		

## 11.2 Card-Host Message Parameters

**Table 11.2-1 - CP\_system\_id Values**

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

**Table 11.2-2 - CP System Message Parameters**

<b>Datatype_id</b>	<b>id value</b>	<b>Length (Bytes)</b>
Reserved	1	
Reserved	2	
Reserved	3	
Reserved	4	
Host_ID	5	5
Card_ID	6	8
Host_ManCert (Host's Manufacturer Certificate)	7	2048*
Card_ManCert (Card's Manufacturer Certificate)	8	2048*
Reserved	9	
Reserved	10	
N_Host (Host's random value for CPKey calculation)	11	8
N_Card (Card's random value for CPKey calculation)	12	8
DH_pubKey <sub>H</sub> (Host's DH public key)	13	128
DH_pubKey <sub>C</sub> (Card's DH public key)	14	128
Host_DevCert (Host's Device Certificate)	15	2048*
Card_DevCert (Card's Device Certificate)	16	2048*
SIGN <sub>H</sub> (the signature of Host's DH public key)	17	128
SIGN <sub>C</sub> (the signature of Card's DH public key)	18	128
CCI_N_Host (Host's random value for CCI exchange)	19	8
Reserved	20	
Reserved	21	
AuthKey <sub>H</sub> (Host Authentication Key before verification)	22	20
Reserved	23	
CCI_N_Card (Card's random value for CCI exchange)	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

\* The Host SHALL report 2048-byte values for Host\_DevCert and Host\_ManCert by adding NULL bytes (0x00) at the trailing end if the stored certificates are less than 2048-bytes.

\* The Card SHALL report 2048-byte values for Card\_DevCert and Card\_ManCert by adding NULL bytes (0x00) at the trailing end if the stored certificates are less than 2048-bytes.

For example, a 2000-byte certificate would be padded to 2048 bytes by adding 48 trailing NULL bytes.

### 11.3 Opening a CP Session

The Card SHALL request a session to be opened to the Copy Protection resource according to the protocol defined in [CCIF]. Since the Host provides the Copy Protection resource, it replies with a session number in its response, if the session is opened.

The two objects used here are defined at Session Protocol Data Unit (SPDU) layer. Detailed SPDU data structure and other SPDU objects are defined in [CCIF].

**Table 11.3-1 - Copy Protection Open Session Information**

SPDU	Tag Value	Action	Direction
open_session_request()	0x91	The Card requests a session of the Copy Protection resource to be opened.	Card → Host
open_session_response()	0x92	The Host responds with a session status. If opened, a session number is assigned. The session number is used for all subsequent exchanges of messages (APDUs) between Card and Host.	Card ← Host

The resource\_identifier requested by the Card in the *open\_session\_request()* SPDU for Copy Protection will match in class, type, and version of a resource that the Host has reported in its list of available resources. Copy protection resource coding is listed in Table 11.3-2.

**Table 11.3-2 - CableCARD Copy Protection Resource**

Resource	Mode	Class	Type	Version	Identifier (hex)
Copy Protection	S-Mode	176	3	1	0x00B000C1
Copy Protection	S-Mode	176	3	2	0x00B000C2
Copy Protection	M-Mode	176	4	3	0x00B00103
Copy Protection	M-Mode	176	4	4	0x00B00104

**Table 11.3-3 - CableCARD Copy Protection Resource - Host Optional**

Resource	Mode	Class	Type	Version	Identifier (hex)
Copy Protection	M-Mode	176	4	2	0x00B00102

The Host SHALL support all CP resource versions listed in Table 11.3-2 for each Mode that it supports.

The Host MAY support CP resource versions listed in Table 11.3-3.

The Host SHALL NOT report in its list of available resources any CP resource other than those listed in Table 11.3-2 or Table 11.3-3.

The Host SHALL refuse any request from the Card for a CP resource not listed on Table 11.3-2 or Table 11.3-3 with the appropriate return code as defined in [CCIF].

The Card SHALL support all CP resource versions listed in Table 11.3-2 for each Mode that it supports.

The Card SHALL request only the CP resources listed in Table 11.3-2 and no other versions.

The Card SHALL request only the highest value CP resource version that both Card and Host support.

The Card SHALL, when and only when using CP resource 0x00B000C1 or 0x00B00103, ignore any RCT value delivered by the CA System and set RCT to zero and only zero.

### 11.3.1 Host CP Support Capability Evaluation

The Card checks the Host's ability to support the CP System when the Card is powered-on and before starting the Key Exchange process.

**Table 11.3-4 - Host CP Support Capability Evaluation Messages**

APDU Tag / Object	Tag Value	Action	Direction
CP_open_req()	0x9F 9000	Card queries which copy protection system is supported by Host.	Card → Host
CP_open_cnf()	0x9F 9001	Host replies to Card	Card ← Host

#### 11.3.1.1 CP\_open\_req() Syntax

This CP\_open\_req() APDU is issued by the Card to query the Host's ability to support various copy protection systems.

**Table 11.3-5 - Card's CP Support Request Message Syntax**

Message Syntax	bits	bytes	Description
CP_open_req () { CP_open_req_tag length_field() }	24 8	3 1	0x9F 9000 0x00

#### 11.3.1.2 CP\_open\_cnf() Syntax

The CP\_open\_cnf() APDU is issued by the Host in response to the CP\_open\_req() APDU. If System 2 is not supported, the Card will treat the Host as if its Device Certificate was invalid.

**Table 11.3-6 - Host's CP Support Confirm Message Syntax**

Message Syntax	bits	bytes	Description
CP_open_cnf () { CP_open_cnf_tag length_field() CP_system_id_bitmask }	24 8 32	3 1 4	0x9F 9001 0x04 Values are listed in Table 11.3-7

**Table 11.3-7 - CP\_system\_id\_bitmask Values**

CP_system_id_bitmask	Bit Number	Description
System 1	0	reserved
System 2	1	CableCARD-CP System

CP_system_id_bitmask	Bit Number	Description
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.

## 11.4 Card-Host Mutual Authentication Message Protocol

### 11.4.1 Mutual Authentication Data Exchange Messages

The CP\_data\_req() APDU and the CP\_data\_cnf() APDU are used to exchange the authentication messages.

**Table 11.4-1 - Authentication Data Exchange Messages**

APDU Tag / Object	Tag Value	Action	Direction
CP_data_req()	0x9F 9002	Card sends its authentication data to Host	Card → Host
CP_data_cnf()	0x9F 9003	Host replies to Card with its authentication data	Card ← Host

#### 11.4.1.1 Card Authentication Data Message

The Card issues the CP\_data\_req() APDU to send Card\_DevCert, Card\_ManCert, DH\_pubKey<sub>C</sub> and SIGN<sub>C</sub> to the Host and request the Host's authentication data.

**Table 11.4-2 - Card Authentication Data Message Syntax**

Syntax	bits	bytes	Description
CP_data_req () {			
CP_data_req_tag	24	3	0x9F 9002
length_field()	24	3	0x82 1113
CP_system_id	8	1	0x02
Send_datatype_nbr	8	1	0x04
for (i=0; i<Send_datatype_nbr; i++) {	<b>(96)</b>	<b>(12)</b>	
Datatype_id	8	1	i = 0, Datatype_id = 16 (Card_DevCert)
	8	1	i = 1, Datatype_id = 8 (Card_ManCert)
	8	1	i = 2, Datatype_id = 14 (DH_pubKey <sub>C</sub> )
	8	1	i = 3, Datatype_id = 18 (SIGN <sub>C</sub> )
Datatype_length	16	2	i = 0, Datatype_length = 2048
	16	2	i = 1, Datatype_length = 2048
	16	2	i = 2, Datatype_length = 128
	16	2	i = 3, Datatype_length = 128
for (j=0; j<Datatype_length; j++) {	<b>34816</b>	<b>(4352)</b>	
Data_type	16384	2048	i = 0, Data_type = Card_DevCert
	16384	2048	i = 1, Data_type = Card_ManCert
	1024	128	i = 2, Data_type = DH_pubKey <sub>C</sub>
	1024	128	i = 3, Data_type = SIGN <sub>C</sub>
}			
}			
Request_datatype_nbr	8	1	0x04
for (i=0; i<Request_datatype_nbr; i++) {	<b>(32)</b>	<b>(4)</b>	
Datatype_id	8	1	i = 0, Datatype_id = 15 (Host_DevCert)
	8	1	i = 1, Datatype_id = 7 (Host_ManCert)
	8	1	i = 2, Datatype_id = 13 (DH_pubKey <sub>H</sub> )
	8	1	i = 3, Datatype_id = 17 (SIGN <sub>H</sub> )
}			
}			

**11.4.1.2 Host Authentication Data Message**

The Host issues the CP\_data\_cnf() APDU in response to the CP\_data\_req() APDU to send Host\_DevCert, Host\_ManCert, DH\_pubKey<sub>H</sub> and SIGN<sub>H</sub> to the Card.

**Table 11.4-3 - Host Authentication Data Message Syntax**

Syntax	bits	bytes	Description
CP_data_cnf () {			
CP_data_cnf_tag	24	3	0x9F 9003
length_field()	24	3	0x82 110E
CP_system_id	8	1	0x02
Send_datatype_nbr	8	1	0x04
for (i=0; i<Send_datatype_nbr; i++) {	(96)	(12)	
Datatype_id	8	1	i = 0, Datatype_id = 15 (Host_DevCert)
	8	1	i = 1, Datatype_id = 7 (Host_ManCert)
	8	1	i = 2, Datatype_id = 13 (DH_pubKey <sub>H</sub> )
	8	1	i = 3, Datatype_id = 17 (SIGN <sub>H</sub> )
Datatype_length	16	2	i = 0, Datatype_length = 2048
	16	2	i = 1, Datatype_length = 2048
	16	2	i = 2, Datatype_length = 128
	16	2	i = 3, Datatype_length = 128
for (j=0; j<Datatype_length; j++) {	(34816)	(4352)	
Data_type	16384	2048	i = 0, Data_type = Host_DevCert
	16384	2048	i = 1, Data_type = Host_ManCert
	1024	128	i = 2, Data_type = DH_pubKey <sub>H</sub>
	1024	128	i = 3, Data_type = SIGN <sub>H</sub>
}			
}			
}			

## 11.4.2 AuthKey Verification Messages

The CP\_data\_req() APDU and the CP\_data\_cnf() APDU are used to obtain the authentication key from the Host.

**Table 11.4-4 - AuthKey Verification Messages**

APDU Tag / Object	Tag Value	Action	Direction
CP_data_req()	0x9F 9002	Card requests Host authentication key	Card → Host
CP_data_cnf()	0x9F 9003	Host replies to Card with AuthKey <sub>H</sub>	Card ← Host

### 11.4.2.1 Card Request for Host AuthKey Message

The Card issues the CP\_data\_req() APDU with Datatype\_id = 22 to request the Host's authentication key (AuthKey<sub>H</sub>).



**Table 11.4-5 - Card Request for Host AuthKey Message Syntax**

Syntax	bits	bytes	Description
CP_data_req () {			
CP_data_req_tag	24	3	0x9F 9002
length_field()	8	1	0x04
CP_system_id	8	1	0x02
Send_datatype_nbr	8	1	0x00
Request_datatype_nbr	8	1	0x01
for (i=0; i<Request_datatype_nbr; i++) {			
Datatype_id	8	1	Datatype_id = 22 (AuthKey <sub>H</sub> )
}			
}			

**11.4.2.2 Reply Message with Host AuthKey**

The Host issues the CP\_data\_cnf() APDU with Datatype\_id = 22 to send its authentication key (AuthKey<sub>H</sub>) to the Card.

**Table 11.4-6 - Host Reply with AuthKey Message Syntax**

Syntax	bits	bytes	Description
CP_data_cnf () {			
CP_data_cnf_tag	24	3	0x9F 9003
length_field()	8	1	0x19
CP_system_id	8	1	0x02
Send_datatype_nbr	8	1	0x01
for (i=0; i<Send_datatype_nbr; i++) {	(184)	(23)	
Datatype_id	8	1	Datatype_id = 22 (AuthKey <sub>H</sub> )
Datatype_length	16	2	Datatype_length = 20
for (j=0; j<Datatype_length; j++) {			
Data_type	160	20	Data_type = AuthKey <sub>H</sub>
}			
}			
}			

**11.5 Copy Protection Key Generation Protocol****11.5.1 Copy Protection Key Data Exchange Messages**

The CP\_data\_req() APDU and the CP\_data\_cnf() APDU are used to exchange parameters used for CPKey generation.

**Table 11.5-1 - CPKey Generation Messages**

APDU Tag / Object	Tag Value	Action	Direction
CP_data_req()	0x9F 9002	Card requests the generation of a new transmission key. This message contains Card_ID and N_Card	Card → Host
CP_data_cnf()	0x9F 9003	Host replies to the Card with Host_ID and N_Host.	Card ← Host

### 11.5.2 Card CPKey Generation Message

The Card issues the CP\_data\_req() APDU to send Card\_ID and N\_Card and request Host\_ID and N\_Host.

**Table 11.5-2 - Card CPKey Generation Message Syntax**

Syntax	bits	bytes	Description
CP_data_req () {			
CP_data_req_tag	24	3	0x9F 9002
length_field()	8	1	0x1B
CP_system_id	8	1	0x02
Send_datatype_nbr	8	1	0x02
for(i=0; i<Send_datatype_nbr; i++) {	(48)	(6)	
Datatype_id	8	1	i = 0, Datatype_id = 6 (Card_ID)
	8	1	i = 1, Datatype_id = 12 (N_Card)
Datatype_length	16	2	i = 0, Datatype_length = 8
	16	2	i = 1, Datatype_length = 8
for (j=0; j<Datatype_length; j++) {	(128)	(16)	
Data_type	64	8	j = 0, Data_type = Card_ID
	64	8	j = 1, Data_type = N_Card;
}			
}			
Request_datatype_nbr	8	1	0x02
for (i=0;i<Request_datatype_nbr; i++) {	(16)	(2)	
Datatype_id	8	1	i = 0, Datatype_id = 5 (Host_ID)
	8	1	i = 1, Datatype_id = 11 (N_Host)
}			
}			

### 11.5.3 Host CPKey Generation Message

The Host issues the CP\_data\_cnf() APDU in response to the CP\_data\_req() APDU to send Host\_ID and N\_Host.

**Table 11.5-3 - Host's CPKey Generation Message Syntax**

Syntax	bits	bytes	Description
CP_data_cnf () {			
CP_data_cnf_tag	24	3	0x9F 9003
length_field()	8	1	0x15
CP_system_id	8	1	0x02
Send_datatype_nbr	8	1	0x02
for(i=0; I<Send_datatype_nbr; i++) {	(48)	(6)	
Datatype_id	8	1	i = 0, Datatype_id= 5 (Host_ID)
	8	1	i = 1, Datatype_id=11 (N_Host)
Datatype_length	16	2	i = 0, Datatype_length = 5
	16	2	i = 1, Datatype_length = 8
for(j=0; j<Datatype_length; j++) {	(104)	(13)	
Data_type	40	5	i = 0, Data_type = Host_ID
	64	8	i = 1, Data_type = N_Host
}			
}			
}			

## 11.6 Card-Host CPKey Synchronization Protocol

### 11.6.1 CPKey Sync Message

The CP\_sync\_req() APDU and the CP\_sync\_cnf() APDU are used to initiate transfer of CP-encrypted MPEG programs across the Card-Host interface.

**Table 11.6-1 - Card-Host CPKey Synchronization Messages**

APDU Tag / Object	Tag Value	Action	Direction
CP_sync_req()	0x9F 9004	The Card notifies the Host when it is ready to start to transmit the CP data.	Card → Host
CP_sync_cnf()	0x9F 9005	Host replies to confirm Host is ready.	Card ← Host

### 11.6.2 Card CPKey Ready Message

The Card issues the CP\_sync\_req() APDU to notify the Host of its intention to start CP-encryption of protected MPEG programs based on the derived CPKey.

**Table 11.6-2 - Card CPKey Ready Message Syntax**

Syntax	bits	bytes	Description
CP_sync_req () { CP_sync_req_tag length_field() }	24 8	3 1	0x9F 9004 0x00

### 11.6.3 Host CPKey Ready Message

The Host sends the CP\_sync\_cnf() APDU in response to the CP\_sync\_req() APDU.

**Table 11.6-3 - Host CPKey Ready Message Syntax**

Syntax	bits	bytes	Description
CP_sync_cnf () { CP_sync_cnf_tag length_field() Status_field }	24 8 8	3 1 1	0x9F 9005 0x01 Values are listed in Table 11.6-4

Status\_field reports the status of the CP\_sync\_req(). The Host will set status-field to 0x00 when it is ready to receive the incoming stream or otherwise as indicated in Table 11.6-4.

**Table 11.6-4 - Host Status\_field Value**

Status_field	Value
OK	0x00
Error – No CP support	0x01
Error – Host Busy	0x02
Reserved	0x03 to 0xFF

## 11.7 CCI Delivery Protocol

### 11.7.1 CCI Delivery Message

The CP\_data\_req() APDU and the CP\_data\_cnf() APDU are used to deliver the CCI byte to the Host in an authenticated manner. The authenticated protocol uses two pairs of request-confirm messages exchanged between the Card and Host.

**Table 11.7-1 - CCI Delivery Protocol Messages**

APDU Tag / Object	Tag Value	Action	Direction
CP_data_req	0x9F 9002	Card initiates CCI delivery protocol. The message contains the random value generated by the Card (CCI_N_Card) and the same program number found in the CA_pmt_req() message <i>and in M-Mode also the LTSID.</i>	Card → Host
CP_data_cnf	0x9F 9003	Host replies to Card with the random value generated by the Host (CCI_N_Host) and the program number, <i>and in M-Mode also the LTSID.</i>	Card ← Host
CP_data_req	0x9F 9002	Card sends the CCI (CCI_data), the program number and the calculated message digest (CCI_auth) <i>and in M-Mode also the LTSID.</i>	Card → Host
CP_data_cnf	0x9F 9003	Host replies to Card with CCI_ack and program number, <i>and in M-Mode also the LTSID.</i>	Card ← Host

### 11.7.2 Card CCI Challenge Message

The Card generates a nonce (CCI\_N\_Card) and sends the CP\_data\_req() APDU to the Host along with program\_number (and LTSID if operating in M-Mode).

**Table 11.7-2 - Card's CCI Challenge Message Syntax**

Syntax	bits	bytes	Description
CP_data_req(){			
CP_data_req_tag	24	3	0x9F 9002.
length_field()	8	1	For S-Mode value = 0x15 <b>For M-Mode value = 0x1A</b>
CP_system_id	8	1	0x02
Send_datatype_nbr	8	1	For S-Mode value = 2 <b>For M-Mode value = 3</b>
for (i=0; i<Send_datatype_nbr; i++) {			
Datatype_id	8	1	i = 0, Datatype_id = 24 (CCI_N_Card)
	8	1	i = 1, Datatype_id = 26 (program_number)
	<b>8</b>	<b>1</b>	<b>Only for M-Mode:</b> <b>i = 2, Datatype_id = 29 (LTSID)</b>
Datatype_length	16	2	i = 0, Datatype_length = 8
	16	2	i = 1, Datatype_length = 2
	<b>16</b>	<b>2</b>	<b>Only for M-Mode:</b> <b>i = 2, Datatype_length = 1</b>
for (j=0; j<Datatype_length; j++) {			
Data_type	64	8	i = 0, Data_type = CCI_N_Card
	16	2	i = 1, Data_type = program_number
	<b>8</b>	<b>1</b>	<b>Only for M-Mode:</b> <b>i = 2, Data_type = f LTSID</b>
}			
}			
Request_datatype_nbr	8	1	For S-Mode value = 2 <b>For M-Mode: value = 3</b>
for (i=0; i<Request_datatype_nbr; i++) {			
Datatype_id	8	1	When i=0, Datatype_id = 19 (CCI_N_Host)
	8	1	When i=1, Datatype_id = 26 (program_number)
	<b>8</b>	<b>1</b>	<b>Only for M-Mode:</b> <b>i=2, Datatype_id = 29 (LTSID)</b>
}			
}			

### 11.7.3 Host CCI Response Message

The Host generates a nonce (CCI\_N\_Host) and sends the CP\_data\_cnf() APDU to the Card along with program number (and LTSID if operating in M-Mode).

**Table 11.7-3 - Host's CCI Response Message Syntax**

Syntax	bits	bytes	Description
CP_data_cnf(){			
CP_data_cnf_tag	24	3	0x9F 9003
length_field()	8	1	For S-Mode value = 0x12 <b>For M-Mode value = 0x16</b>
CP_system_id	8	1	0x02.
Send_datatype_nbr	8	1	For S-Mode value = 2 <b>For M-Mode value = 3</b>
for (i=0; i<Send_datatype_nbr; i++) {			
Datatype_id	8	1	i = 0, Datatype_id = 19 (CCI_N_Host)
	8	1	i = 1, Datatype_id = 26 (program_number)
	<b>8</b>	<b>1</b>	<b>Only for M-Mode:</b> <b>i = 2, Datatype_id = 29 (LTSID)</b>
Datatype_length	16	2	i = 0, Datatype_length = 8
	16	2	i = 1, Datatype_length = 2
	<b>16</b>	<b>2</b>	<b>Only for M-Mode:</b> <b>i = 2, Datatype_length = 1</b>
for (j=0; j<Datatype_length; j++) {			
Data_type	64	8	i = 0, Data_type = CCI_N_Host
	16	2	i = 1, Data_type = program_number
	<b>8</b>	<b>1</b>	<b>Only for M-Mode:</b> <b>i = 2, Data_type = LTSID</b>
}			
}			
}			

#### 11.7.4 CCI Delivery Message

The Card calculates a message digest (CCI\_auth) using the CCI byte, CPKey(s), program number, CCI\_N\_Card, CCI\_N\_Host (and LTSID if operating in M-Mode) and sends it to the Host with the CCI byte using the CP\_data\_req() APDU.

**Table 11.7-4 - CCI Delivery Message Syntax**

Syntax	bits	bytes	Description
CP_data_req(){	24	3	0x9F 9002.
CP_data_req_tag	8	1	For S-Mode value = 0x25
length_field()			<b>For M-Mode value = 0x2A</b>
CP_system_id	8	1	0x02
Send_datatype_nbr	8	1	For S-Mode value = 3
			<b>For M-Mode value = 4</b>
for (i=0; i<Send_datatype_nbr; i++) {			
Datatype_id	8	1	i = 0, Datatype_id = 25 (CCI_data)
	8	1	i = 1, Datatype_id = 2 (program_number)
	8	1	i = 2, Datatype_id = 27 (CCI_auth)
	<b>8</b>	<b>1</b>	<b>Only for M-Mode:</b>
			<b>i = 3, Datatype_id = 29 (LTSID)</b>
Datatype_length	16	2	i = 0, Datatype_length = 1
	16	2	i = 1, Datatype_length = 2
	16	2	i = 2, Datatype_length = 20
	<b>16</b>	<b>2</b>	<b>Only for M-Mode:</b>
			<b>i = 3, Datatype_length = 1</b>
for (j=0; j<Datatype_length; j++) {			
Data_type	8	1	i = 0, Data_type = CCI_data
	16	2	i = 1, Data_type = program_number
	160	20	i = 2, Data_type = CCI_auth
	<b>8</b>	<b>1</b>	<b>Only for M-Mode:</b>
			<b>i = 3, Data_type = LTSID.</b>
}			
}			
Request_datatype_nbr	8	1	For S-Mode value = 2
			<b>For M-Mode value = 3</b>
for (i=0; i<Request_datatype_nbr; i++) {			
Datatype_id	8	1	i=0, Datatype_id = 28 (CCI_ack)
	8	1	i=1, Datatype_id = 26 (program_number)
	<b>8</b>	<b>1</b>	<b>Only for M-Mode:</b>
			<b>i=2, Datatype_id = 29 (LTSID)</b>
}			
}			

### 11.7.5 CCI Acknowledgement Message

The Host authenticates the Card's message digest (CCI\_auth), calculates a new message digest (CCI\_ack), and sends it to the Card using the CP\_data\_cnf() APDU.

**Table 11.7-5 - CCI Acknowledgement Message Syntax**

Syntax	bits	bytes	Description
CP_data_cnf() {			
CP_data_cnf_tag	24	3	0x9F 9003.
length_field()	8	1	For S-Mode value = 0x1E <b>For M-Mode value = 0x22</b>
CP_system_id	8	1	0x02
Send_datatype_nbr	8	1	For S-Mode value = 2 <b>For M-Mode value = 3</b>
for (i=0; i<Send_datatype_nbr; i++) {			
Datatype_id	8	1	i = 0, Datatype_id = 28 (CCI_ack)
	8	1	i = 1, Datatype_id = 26 (program_number)
	<b>8</b>	<b>1</b>	<b>For M-Mode only:</b> <b>i = 2, Datatype_id = 29 (LTSID)</b>
Datatype_length	16	2	i = 0, Datatype_length = 20
	16	2	i = 1, Datatype_length = 2
	<b>16</b>	<b>2</b>	<b>For M-Mode only:</b> <b>i = 2, Datatype_length = 1</b>
for (j=0; j<Datatype_length; j++) {			
Data_type	160	20	i = 0, Data_type = CCI_ack
	16	2	i = 1, Data_type = program_number
	<b>8</b>	<b>1</b>	<b>For M-Mode only:</b> <b>i = 2, Data_type = LTSID.</b>
}			
}			
}			

## 11.8 Card Validation Status

The Host requests the Card validation status by sending the CP\_valid\_req() APDU. The Card replies with the CP\_valid\_cnf() APDU containing the binding and authentication status of the Card.

After the first time the Host requests Card validation status, the Card SHALL send CP\_valid\_cnf() to the Host unsolicited, whenever the status value changes, with the current value as indicated in Table 11.8-4.

The Card SHALL NOT send CP\_valid\_cnf() unsolicited to the Host until the CP\_valid\_req() APDU has been received at least once.

**Table 11.8-1 - Card Validation Status Messages**

APDU Tag / Object	Tag Value	Action	Direction
CP_valid_req()	0x9F 9006	The Host requests from the Card the Card Validation Status	Card ← Host
CP_valid_cnf()	0x9F 9007	Card replies to Host with the Card Validation status.	Card → Host

**Table 11.8-2 - Host Validation Status Request Message Syntax (type 4 ver 3)**

Message Syntax	bits	bytes	Description
CP_valid_req() {			
CP_valid_req_tag	24	3	0x9F 9006
length_field()	8	1	0x00
}			



**Table 11.8-3 - Card Validation Status Reply Message Syntax (type 4 ver 3)**

Message Syntax	bits	bytes	Description
CP_valid_cnf ( ) { CP_valid_cnf_tag length_field() Status_field }	24 8 8	3 1 1	0x9F 9007 0x01 Values are listed in Table 11.8-4

**Table 11.8-4 - Card Validation Status\_field Value**

Status_field	Value
Card is busy with binding authentication process	0x00
Not bound for Card reasons	0x01
Not bound, Host Certificate Invalid	0x02
Not bound, failed to verify Host's SIGN <sub>H</sub>	0x03
Not bound, failed to match AuthKey from Host	0x04
Binding Failed, other reasons	0x05
Not Validated, Binding Authentication Complete, Validation message not received yet	0x07
Validated, validation message is received, authenticated, and the IDs match those in the current binding	0x06
Not Validated, validation revoked	0x08
Reserved	0x09 to 0xFF

## Annex A Luhn Check Digit (Normative)

The Luhn check digit is calculated over decimal values using the following algorithm.

1. Convert the value into the appropriate decimal format (see Section 3.2.1 of [SCTE 41]).
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 3B2C 021F (hexadecimal), made up from decimal manufacturer number 004 and Unit ID 992,739,871:

1. Concatenate to the 12-digit decimal value 004,992,739,871 per Section 3.2.1 of [SCTE 41].
2. Separate this decimal number into odd and even digits starting from the right (least significant digit):  
 digit #:  $^{12}_{11}^{10}987654321$   
 'odd' digits: 0, 9, 2, 3, 8, 1  
 'even' digits: 0, 4, 9, 7, 9, 7
3. Multiply each 'odd' digit by 2:  
 0, 9, 2, 3, 8, 1  $\rightarrow$  0, 18, 4, 6, 16, 2
4. Add the 'even' digits and each individual digit of the products above:  
 $[0 + 4 + 9 + 7 + 9 + 7] + [0 + 1 + 8 + 4 + 6 + 1 + 6 + 2] = 64$
5. Subtract this sum from 70 to form the check digit:  
 $70 - 64 = 6$

The Luhn check digit for this example is “6”.

## Annex 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 [FIPS 46-3] and [FIPS 81]. This specification defines generation of a 56-bit integer DESKey. The 64-bit key is generated from DESKey by adding a parity bit to each 7-bit block.

Starting with DESKey in a 56-bit format:

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

By adding parity bits after each 7 bits of DESKey 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 DESKey:

DESKey = 0x01 2345 6789 ABCD  
 = 0000 0001 0010 0011 0100 0101 0110 0111 1000 1001 1010 1011 1100 1101 b

Break it into eight 7-bit blocks:

CPKey= 00000000 1001000 1101000 1010110 0111100 0100110 1010111 1001101 b

Add a parity bit at the end of each 7-bit block, thus making it an octet, to get the 64-bit key:

$K_{64bit} = 00000001 10010001 11010000 10101101 01111001 01001100 10101110 10011011 b$   
 = 0x0191 D0AD 794C AE9B

## B.2 Examples of S-Mode CP Encryption of MPEG DATA in Transport Packets

This section shows examples of packets before and after DES encryption by the copy protection system for the Card operating in S-Mode. The encryption key used here is 0x**0123 4567 89AB CDEF** in 64-bit format (or 0x00 4513 3895 7377 in 56-bit format), which is shown in [FIPS 180-2] 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) as a sequence of hexadecimal digits.

### Example 1: A null packet (hex).

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

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

### Example 2: A packet without adaptation field that belongs to a copy protected MPEG program (hex).

```
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 **00b** to **11b** (4<sup>th</sup> byte: 0x1c to 0xdc). 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 MPEG program (hex).

```
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. The MPEG-TS **transport\_scrambling\_control** field is changed as described in Example 2 above.

## B.3 M-Mode Transport Packet Encryption with Triple DES

This section specifies the method of applying the triple DES cipher to scramble the payload of MPEG-2 transport packets for 2-key triple DES.

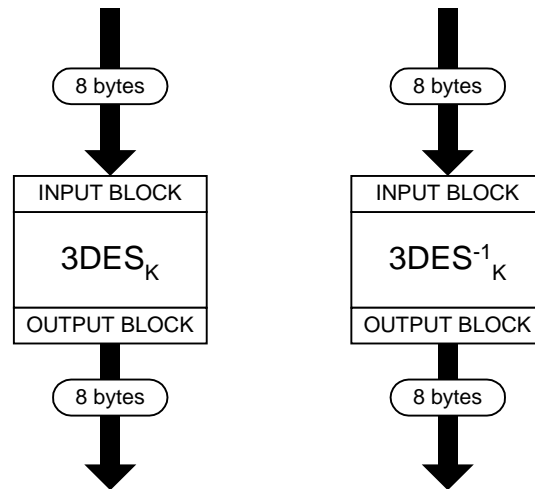
Under the MPEG standard, the scrambled payload length can be any value from 1 to 184 bytes, so special consideration is given to the case where this is not an integral multiple of the cipher block size.

The following variables are used in the specification of the copy protection encryption :

- N MPEG Transport Packet payload length, in bytes ( $0 \leq N \leq 184$ )
- b Cipher block size, in bytes.  $b = 8$  for the 3DES cipher.
- i Integer number of full cipher blocks in a Transport Packet payload
- p Number of residual Transport Packet payload bytes remaining after all full cipher blocks are formed.  
( $0 \leq p < b$ )

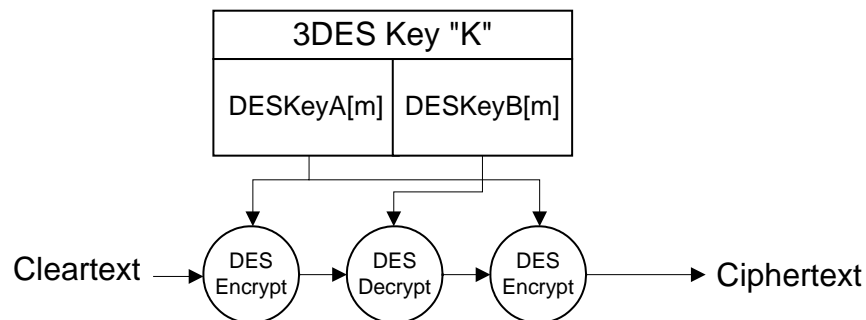
### B.3.1 3DES Cipher

The notation used for the Electronic Codebook (ECB) mode of 3DES, under key  $K$ , is shown in Figure B.3-1.  $3DES_K$  denotes the cipher operation (encryption), and  $3DES_K^{-1}$  denotes the inverse cipher operation (decryption). Heavy arrows indicate data flow, with the data size as indicated.

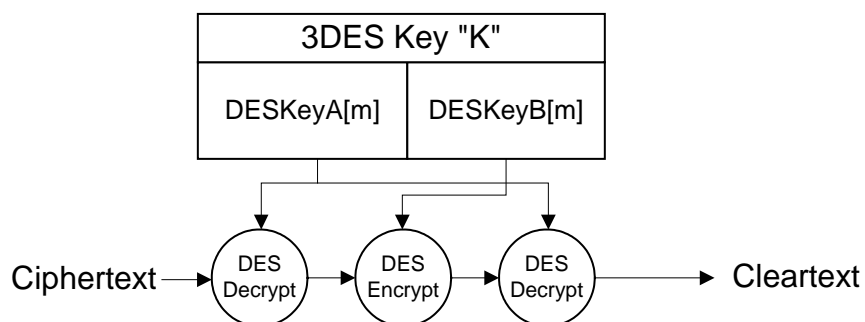


**Figure B.3-1 - 3DES Encryption and Decryption, ECB mode**

In M-Mode, the key  $K$  is a concatenation of CPKeyA and CPKeyB, and EDE-121 mode is used. (See Figure B.3-2 for the encrypt and Figure B.3-3 for the decrypt operations.)



**Figure B.3-2 - 2-key Triple DES, EDE-121 mode, Encryption**



**Figure B.3-3 - 2-key Triple DES, EDE-121 mode, Decryption**

### B.3.2 MPEG Transport Packet Scrambling and Descrambling

The MPEG standard specifies that scrambling be applied to Transport Packet payload only. The Transport Stream packet header, and adaptation field when present, shall not be scrambled. Because of the variable length adaptation field, the payload length (when payload exists) can be any value from 1 to 184 bytes, and is not constrained to be a multiple of a cipher block length. For the purpose of describing the scrambling mechanism, the table below defines three cases of the payload length:

**Table B.3-1 - Payload length cases**

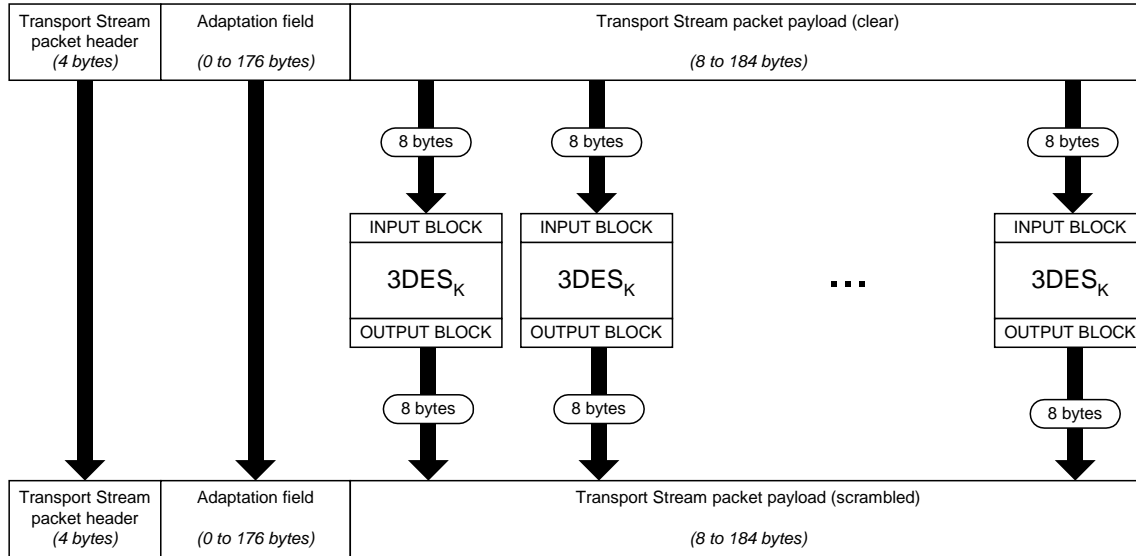
	Full cipher blocks (i)	Residual bytes (p)	Payload length
Case 1	$i > 0$	$p = 0$	Integral number of full cipher blocks
Case 2	$i > 0$	$1 \leq p \leq 7$	At least one full cipher block, plus some residual bytes
Case 3	$i = 0$	$1 \leq p \leq 7$	Less than one full cipher block

The three cases are addressed individually below.

#### B.3.2.1 Case 1: Integral number of full cipher blocks

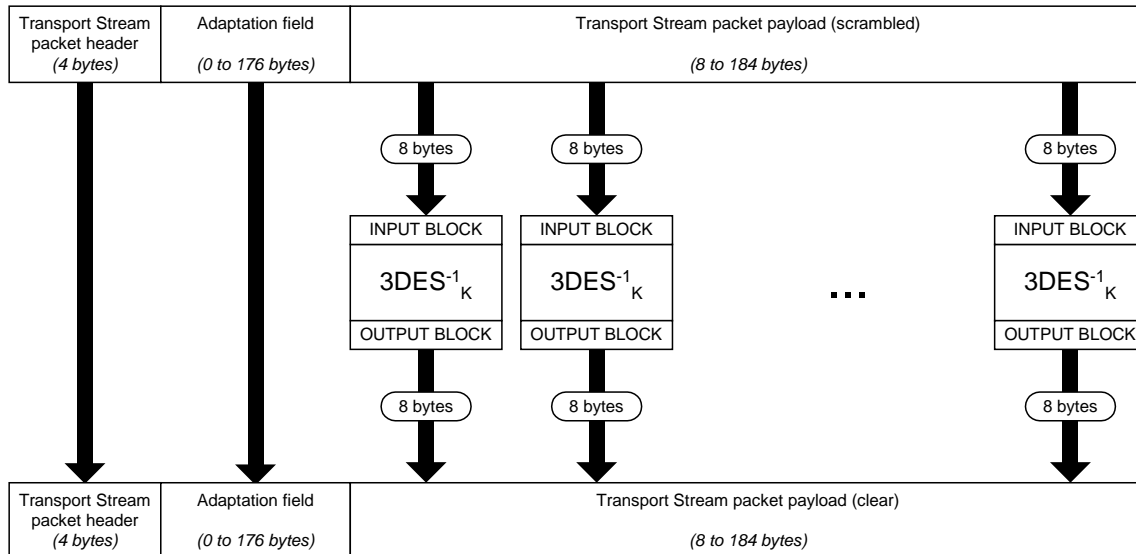
$$N = i * b \quad \text{for } i = 1, 2, \dots, 23$$

This is the simplest case. Each block of  $b$  bytes is encrypted using the Electronic Codebook mode, starting with the first payload byte. No chaining is used. Figure B.3-4 shows the scrambling operation.



**Figure B.3-4 - MPEG Transport Packet Scrambling, Case 1**

The descrambling process is similar, shown in Figure B.3-5:

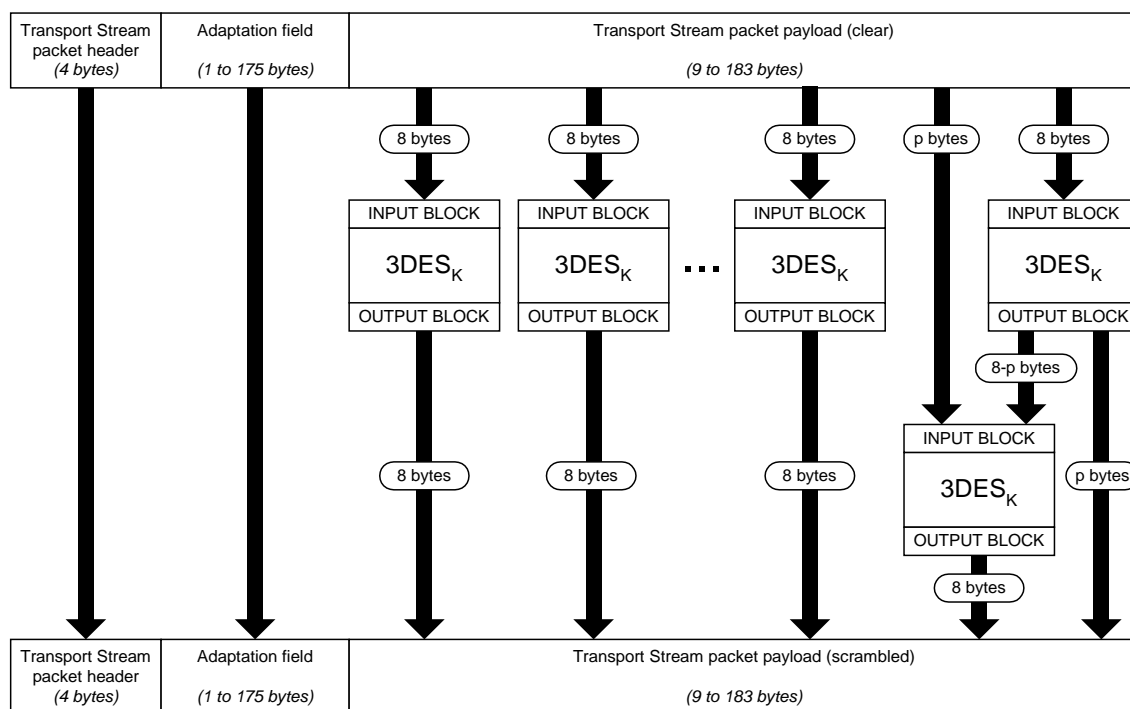


**Figure B.3-5 - MPEG Transport Packet Descrambling, Case 1**

### **B.3.2.2 Case 2: At least one full cipher block, plus residual bytes**

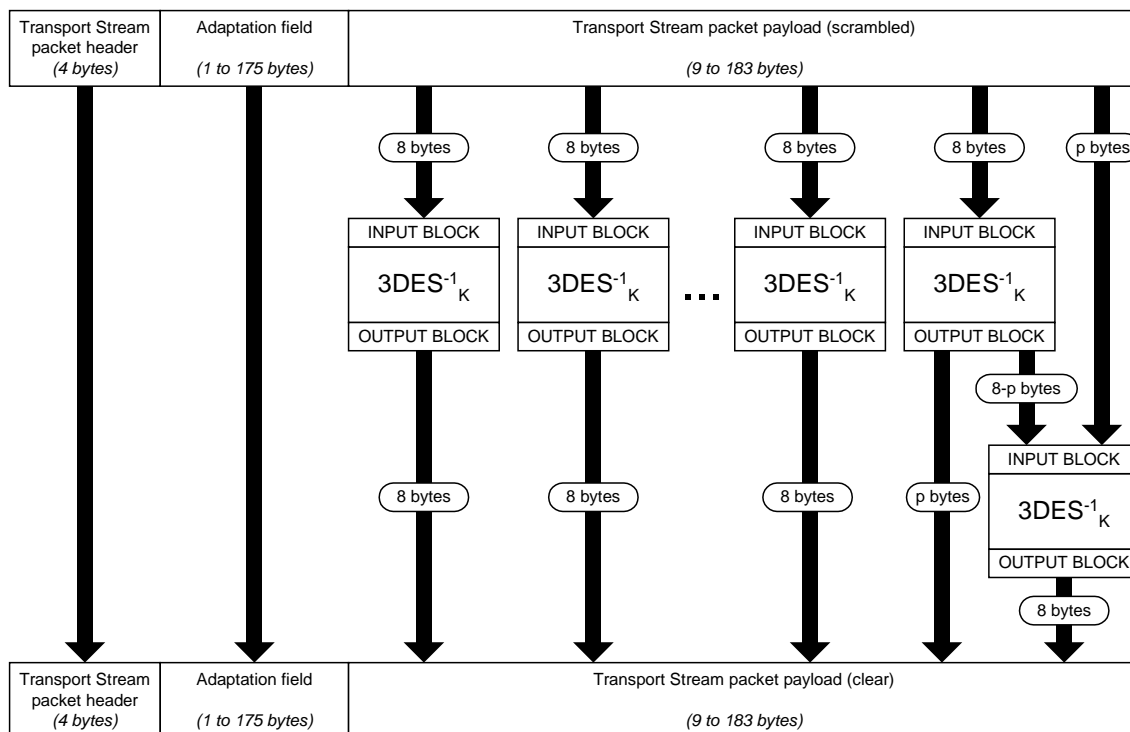
$$N = i * b + p \quad \text{for } 1 \leq i \leq 22, \text{ and } 1 \leq p \leq 7$$

The Transport Stream packet payload, starting from the first byte, is divided into (i-1) 8-byte blocks, one partial block p bytes in length, and one final 8-byte block. The (i-1) full blocks are encrypted using Electronic Codebook mode, similar to Case 1. The remaining partial block and last full block are combined, using the technique of Ciphertext Stealing [FIPS 46-3] and [FIPS 81] resulting in two cipher operations as shown in Figure B.3-6.



**Figure B.3-6 - MPEG Transport Packet Scrambling, Case 2**

The descrambling process is similar, but the Ciphertext Stealing is done in reverse order as shown in Figure B.3-7. The order has been optimized to avoid look-ahead operations in the descrambler.



**Figure B.3-7 - MPEG Transport Packet Descrambling, Case 2**



### B.3.2.3 Case 3: Less than one full cipher block

$$N = p \quad \text{for } 1 \leq p \leq 7$$

Since there is not enough data to fill even one cipher block, Ciphertext Stealing cannot be used. In this case, no encryption is applied to the payload. However, the transport\_scrambling\_control bits of the MPEG transport stream header are still marked as if the packet had been scrambled.

## B.4 Examples of CP Encryption of MPEG DATA in Transport Packets

This section shows examples of packets before and after 3-DES ABA encryption by the copy protection system.

### B.4.1 Case 1, an integral number of full cipher blocks, where N=184:

Key = {DESKeyA[m], DESKeyB[m]} = [0123456789abcdef, 0101232345456767] in 64-bit hex format

Example using 11 for transport scrambling control bits:

Clear Packet (hex):

```
47 00 20 10 01 02 03 04 05 06 07 08 09 0a 0b 0c 0d 0e 0f 10 11 12 13 14 15 16 17 18 19 1a 1b 1c
1d 1e 1f 20 21 22 23 24 25 26 27 28 29 2a 2b 2c 2d 2e 2f 30 31 32 33 34 35 36 37 38 39 3a 3b 3c
3d 3e 3f 40 41 42 43 44 45 46 47 48 49 4a 4b 4c 4d 4e 4f 50 51 52 53 54 55 56 57 58 59 5a 5b 5c
5d 5e 5f 60 61 62 63 64 65 66 67 68 69 6a 6b 6c 6d 6e 6f 70 71 72 73 74 75 76 77 78 79 7a 7b 7c
7d 7e 7f 80 81 82 83 84 85 86 87 88 89 8a 8b 8c 8d 8e 8f 90 91 92 93 94 95 96 97 98 99 9a 9b 9c
9d 9e 9f a0 a1 a2 a3 a4 a5 a6 a7 a8 a9 aa ab ac ad ae af b0 b1 b2 b3 b4 b5 b6 b7 b8
```

Encrypted Packet (hex):

```
47 00 20 90 79 42 c9 50 3a 3b 1d 97 51 4a 8c 16 d3 ad 04 8e 5b 00 8e 15 50 2c e8 44 37 61 c4 ad
c3 37 fa b9 af a2 c8 47 fb 28 ef 8c 7b 86 cd 0a dc c5 79 4d 35 31 55 f8 8c b6 b7 2b ed 96 e0 a2
83 39 4b 11 91 d9 77 39 a5 e6 c9 f9 94 d2 66 52 5c be df 8b d3 5a 36 11 e5 f2 dd d9 c1 f1 1c c7
06 27 86 4d 6c ea a0 fe 65 17 45 8a 40 83 d9 4e cc 82 03 27 0e 4d c9 05 69 b1 72 91 2c 60 c5 f5
6e b0 20 c4 23 09 69 2b 63 f8 f1 94 b7 96 c4 c5 78 5f 77 00 54 73 6b 08 98 db a0 fd 17 07 1d 1f
40 95 31 a7 20 cc f8 41 e3 58 d6 a3 48 91 6d 23 ed a4 8f 5e 3a 96 3b df c5 18 71 87
```

### B.4.2 Case 2: At Least one full Cipher Block, plus residual bytes

N= 173

Key = {DESKeyA[m], DESKeyB[m]} = [8989ababdcdefef, fedcba9876543210] in 64-bit hex format

Example using 11 for transport scrambling control bits:

Clear:

```
47 00 30 30 0a 00 00 00 00 00 00 00 00 00 01 02 03 04 05 06 07 08 09 0a 0b 0c 0d 0e 0f 10 11
12 13 14 15 16 17 18 19 1a 1b 1c 1d 1e 1f 20 21 22 23 24 25 26 27 28 29 2a 2b 2c 2d 2e 2f 30 31
32 33 34 35 36 37 38 39 3a 3b 3c 3d 3e 3f 40 41 42 43 44 45 46 47 48 49 4a 4b 4c 4d 4e 4f 50 51
52 53 54 55 56 57 58 59 5a 5b 5c 5d 5e 5f 60 61 62 63 64 65 66 67 68 69 6a 6b 6c 6d 6e 6f 70 71
72 73 74 75 76 77 78 79 7a 7b 7c 7d 7e 7f 80 81 82 83 84 85 86 87 88 89 8a 8b 8c 8d 8e 8f 90 91
92 93 94 95 96 97 98 99 9a 9b 9c 9d 9e 9f a0 a1 a2 a3 a4 a5 a6 a7 a8 a9 aa ab ac ad
```

Encrypted:

```
47 00 30 f0 0a 00 00 00 00 00 00 00 00 00 00 7e 06 9d c3 ac b7 39 8b 3d c6 88 36 cb 9e 70 24 81
00 6d 06 1c e6 70 97 d0 46 b4 bb de 41 df 87 e5 8b 7e dd 1c f6 91 a5 b2 a6 9c ae 86 86 8d a2 a5
```

62 c0 af c7 db d9 34 e0 ba 1b 90 d9 b1 21 e0 f5 28 a4 80 48 cd dd 39 c8 ff 61 22 44 89 73 1f 8c  
c7 98 fa e1 4e eb b3 92 1a 6b 0e 20 8c f6 8f 4f 33 4f b9 d0 d2 dd fb 0e 7b 31 1e cf 62 ba c8 c7  
94 fc 3d 80 64 a3 94 a8 00 c4 1c 6a b0 6b c0 a8 98 79 7d ce 71 06 bf 79 93 e7 34 be 33 bd 5f e4  
8b 98 e5 11 22 18 04 cd b3 50 3d 58 9a d1 11 55 fe 45 0d b4 bd 00 f1 93 14 40 5a 4b

### B.4.3 Case 3: Less than one Full Cipher Block

$$N=5$$

Key = {DESKeyA[m], DESKeyB[m]} = [0101010123232323, 4545454567676767] in 64-bit hex format

Example using 11 for transport scrambling control bits:

Clear:

[illegible]

Encrypted:

[illegible]

## Appendix I Revision History

The following ECNs were incorporated into OC-SP-CCCP2.0-I02-050708:

Number	Description	Date
CCCP2.0-N-05.0784-3	Remove Host Default CCI	6/24/2005

The following ECNs were incorporated into OC-SP-CCCP2.0-I03-060622:

Number	Description	Date
CCCP2.0-N-06.0893-2	APDU for returning Card Validation status	6/9/06

The following ECNs were incorporated into OC-SP-CCCP2.0-I04-060803:

Number	Description	Date
CCCP2.0-N-06.0902-1	Clarification of DESKey Generation	6/16/06

The following ECNs were incorporated into OC-SP-CCCP2.0-I05-070105:

Number	Description	Date
CCCP2.0-N-06.0939-3	Choose one ECM-PID for DESKey and CCI Protocol	11/10/06
CCCP2.0-N-06.0953-1	CCI Delivery	12/22/06
CCCP2.0-N-06.0969-1	Padding ECM-PID Clarification	1/2/07

The following ECNs were incorporated into OC-SP-CCCP2.0-I06-070323:

Number	Description	Date
CCCP2.0-N-07.0981-1	Clarify choice of ECM-PID	2/23/07
CCCP2.0-N-07.0985-2	ECM-PID correction	3/9/07

The following ECNs were incorporated into OC-SP-CCCP2.0-I07-070615:

Number	Description	Date
CCCP2.0-N-07.1019-1	Delete CCI_auth, CCI_ackcalculation for Type 4 Ver 2	3/20/07
CCCP2.0-N-07.1041-1	Copy Protection resource version usage	6/1/07

The following ECN was incorporated into OC-SP-CCCP2.0-I08-071109:

Number	Description	Date
CCCP2.0-N-07.1109-2	Omnibus to enable linking spec to PICS via ReqPro	10/23/07

The following ECN was incorporated into OC-SP-CCCP2.0-I09-090508:

Number	Description	Date
CCCP2.0-N-08.1232-7	Add RCT bit to CCI	4/17/09

The following ECNs were incorporated into OC-SP-CCCP2.0-I10-090904:

Number	Description	Date
CCCP2.0-N-08.1268-4	DESKey calculation for Multi-Stream Mode without ECM_PID	6/2/09
CCCP2.0-N-09.1394-2	No CCI exchange required if CCI is zero or there is no ECM PID in ca_pmt()	8/7/09

The following ECN was incorporated into OC-SP-CCCP2.0-I11-110512:

Number	Description	Date
CCCP2.0-N-11.1655-2	CCCP: Reference edits for OpenCable bundle inclusions	5/9/11

The following ECN was incorporated into OC-SP-CCCP2.0-I12-050531:

Number	Description	Date
CCCP2.0-N-11.1704-2	Partial Copy Protection M-Mode Version 2 compatibility to allow cards to upgrade firmware	2/24/12

---