OpenCable™ Specifications

CableCARD™ Copy Protection 2.0 Specification

OC-SP-CCCP2.0-I05-070105

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.

Document Status Sheet

Document Control Number:	OC-SP-CCCP2.0-I05-070105			
Document Title:	CableCARD™ Copy Protection 2.0 Specification			
Revision History:	I01—March 31, 2005 I02—July 8, 2005 I03—June 22, 2006 I04—August 3, 2006 I05—January 5, 2007			
Date:	January 5, 2007			
Responsible Editor:	Jeff Hamilton			
Status:	~~Work in Progress~~	~~Draft~~	Issued	~~Closed~~
Distribution Restrictions:	~~author only~~	~~CL/Member~~	~~CL/ Member/ Vendor~~	Public

Key to Document Status Codes:

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

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

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

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

TRADE MARKS:

Contents

List of Figures

List of Tables

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 rescramble 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 2.0].

This document defines the characteristics and normative specifications for the system that prevents unauthorized copying of high value content as it crosses 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 [EIA 679], [SCTE 28], and the S-Mode specified in [CCIF 2.0].

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 2.0].

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 Card and Host devices

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 Host devices 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 2.0].

1.4 Historical Perspective

This specification has its origins in [EIA 679], 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.

[CCIF 2.0] OC-SP-CCIF2.0-I09-070105, OpenCable CableCARD Interface 2.0 Specification, January 5, 2007, Cable Television Laboratories, Inc.

[EIA 679] EIA 679-B, Part B: National Renewable Security Standard, March 2000.

[X.509] ITU-T Recommendation X.509: Information Technology – Open Systems Interconnection – The Directory: Public-key and Attribute Certificate Frameworks, March 2000.

[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.

[RSA1] PKCS #1 v2.1: RSA Cryptography Standard, June 14, 2002.

[MPEG] ISO/IEC 13818-1 (2000): Information Technology - Generic coding of moving pictures and associated audio information: Systems.

[SCTE 41] ANSI/SCTE 41 2004, POD Copy Protection System.

[SCTE 28] ANSI/SCTE 28 2004, HOST-POD Interface Standard.

2.2 Informative References

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

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

[HOST2.0] OC-SP-HOST2.0-CFR-I12-070105, OpenCable Host Device 2.0 Core Functional Requirements, January 5, 2007, Cable Television Laboratories, Inc.

[SEC] OC-SP-SEC-I07-061031, OpenCable System Security Specification, October 31, 2006, Cable Television Laboratories, Inc.

2.3 Reference Acquisition

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

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

EIA Standards: Electronic Industries Association

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

IEEE Standards: Institute of Electrical and Electronic Engineers

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

ISO: International Standards Organization

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

ITU-T: International Telecommunications Union – Telecom Standardization

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

FIPS Publications: Federal Information Processing Standards Publications

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

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 703-605-6000; FAX: 703-321-8547; e-mail orders: orders@ntis.fedworld.gov, URL: <http://www.itl.nist.gov/fipspubs/>

RSA Security

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

SCTE Standards: Society of Cable Telecommunications Engineers

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

3 ACRONYMS, ABBREVIATIONS, DEFINED TERMS AND SYMBOLS

APDU Application Protocol Data Unit: a command, query, and reply message exchange protocol between Card and Host.

APS Analog Protection System for copy control of analog output video.

AuthKey Authentication Key, calculated by both the Card and Host as part of the Host authentication process.

Authenticated Employing or resulting from the use of Authentication: trusted.

Authentication 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.

Binding 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.

Bound 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 IDs by the CA System. The Card-Host pair is fully bound when both steps have completed successfully.

CA, CA System Conditional Access, Conditional Access System – secures delivery of cable services to the Card.

CA-only The Card mode of CA-descrambling zero EMI content and returning it to the Host in-the-clear.

Cable The Cable Television industry, services, systems, or equipment

CableCARD-CP CableCARD copy protection, as specified in this document

Card A PCMCIA card distributed by cable providers and inserted into a Host device to enable reception of premium services without a separate cable receiver, also called CableCARD Device and "Point of Deployment" (POD) module.

Card_ID The Card's unique identification number

CCI Copy Control Information

CHICA CableCARD-Host Interface Certificate Authority, root X.509 certificate administrator for X.509 certificates on the Card-Host interface identified under CHILA; formerly PHICA.

CHILA CableCARD-Host Interface Licensing Agreement, formerly identified as PHILA, covers the DFAST technology and specifies the Certificate Authority –CHICA.

CIT Constrained Image Trigger. Controls use of Image Constraint on high definition analog outputs.

Constrained Image 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.

CP Copy Protection

CPKey The Copy Protection Key derived between the Card and Host, and used by the Card to CP-scramble protected content sent to the Host.

CP System The copy protection system described in this specification

CRL Certificate Revocation List: the means of reporting bad Card and Host IDs to cable headends

DES Data Encryption Standard

DESKey$_m$ The final key entered into the DES-ECB processor to encrypt or decrypt MPEG program 'm'

Device Certificate The XCA certificate unique to each Card or Host device, Card_DevCert and Host_DevCert.

DFAST Dynamic Feedback Arrangement Scrambling Technique, a component of the encryption algorithm

DH Diffie-Hellman, a public key agreement protocol based on the intractability of taking discrete logarithms over the integer field.

DSG DOCSIS Set-top Gateway, a method of using DOCSIS protocols to report IDs to the headend.

ECB Electronic Code Book

ECM-PID Entitlement Control Message – Packet Identifier. This is a 13-bit field indicating the PID of the Transport Stream packets that shall contain the ECM information for the CA_system_ID reported by the Card (ECM_stream) and with a specific MPEG program requested by the Host. In this document, each copy-protected MPEG program will use one and only one ECM-PID value; see Section 4.3.

EMI "Encryption Mode Indicator" defines the copy protection mode for digital outputs.

Encrypted Data modified to prevent unauthorized access (compare with "scrambled").

Headend The cable operator's facility, which acts as the source of cable signals, services, and conditional access control.

High Value Content Content marked with non-zero EMI and/ or CIT= 1.

Host The consumer device used to access and navigate cable content. Typically a digital TV or set-top DTV receiver.

Host_ID The Host device's unique identification number found in the Host Device Certificate.

IIF Initialize Interface Request.

Image Constraint See Constrained Image.

In-the-clear without encryption or scrambling by either the CA or CP system.

lsb Least Significant Bit, of a specified binary value.

LTSID Local Transport Stream ID, assigned by the Host in M-Mode.

Manufacturer Certificate 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.

MMI Man Machine Interface

MPEG The ISO/IEC 13818 specifications and ISO/IEC 13818-1 in particular

MPEG program 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.

msb Most Significant Bit, of a specified binary value.

M-Mode Multi-Stream Mode as defined in [CCIF 2.0].

Nonce A short-term use random value generated fresh for each Card-Host protocol sequence to improve security by making each instance unique.

Non-zero EMI EMI values other than zero, such as 01, 10 or 11, indicating copy protection is asserted on digital interfaces.

Pass-through The Card processing mode of passing a CA-scrambled content back to the Host unchanged, leaving it unusable by the Host.

POD module 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.

RDC Return Data Channel: a communication channel on the coaxial cable that delivers the main cable service but running "upstream" from home to the headend.

Rescramble The Card processing mode of CA-descrambling and CP-scrambling selected content.

Root Certificate The root XCA certificate used by CHICA to issue Manufacturer Certificates and by the Card and Host to validate each other's Manufacturer Certificate.

RSA algorithm An RSA Security defined commercial public key cryptographic algorithm.

Scrambled Content modified to prevent unauthorized access (compare with "encrypted").

SHA-1 Secure Hash Algorithm, a cryptographic compression function, see [FIPS 180-2].

SPDU Session Protocol Data Unit (SPDU)

S-Mode Single Stream Mode as defined in [CCIF 2.0]

Validation 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 IDs back to the Card, and the Card confirming the message was sent by the CA-system and that the IDs match the values the Card reported.

X.509 ITU-T Recommendation X.509

XCA X.509 certificate authority

Zero EMI EMI value of zero (0), not greater than zero, indicating copy protection is not asserted on digital interfaces.

| 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

C^D Exponents are displayed in superscript. C^D means raise the first value, C, to the exponential power of the second, superscripted, value D. For C = 2 and D = 3, $C^D = 2^3 = 8$.

$\oplus$ 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.

0x 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-encrypted MPEG programs, passed to the Card for CA-decryption, 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-decrypts and will apply CP-encryption to programs marked with non-zero EMI, 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 SHALL NOT CA-descramble any content until it has successfully completed authentication of the Host.

The Card and Host each:

a) Calculate a Diffie-Hellman public key from stored secrets and a generated random integer.

b) Sign their DH public key with their X.509 private key (that matches the public key embedded in their X.509 Device Certificate).

c) Send their Device Certificate, Manufacturer Certificate, DH public key, and signature of that key to the other device.

d) Verify the signatures of the other device's signature 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) Confirm that the received AuthKey matches the internally calculated AuthKey.

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

If authorized by the CA System the Card SHALL enable CA-descrambling of content marked with zero EMI.

4.2 ID Reporting and Headend Validation

The Card SHALL NOT CA-descramble any MPEG program with non-zero EMI until 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 and Host_ID from the authenticated Device Certificate and sends them to the headend. Two means are employed to report these IDs. In order of preference:

Automated: If the Card has an active means to report IDs to the headend by electronic means, such as an active cable plant RDC and a Host RDC transmitter, or via a DOCSIS cable modem, the Card MAY use them 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. After receiving and authenticating this message, the Card checks that the received IDs are the same as its current binding, and if so, enables CA decryption of content marked as High Value Content to the bound Host.

4.3 Calculation of Copy Protection Keys

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

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

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

In Single Stream Mode (S-Mode) the shared CPKey is used directly as $DESKey_0$ to initialize the DES processor in the Card and Host. In Multi-Stream Mode (M-Mode) the Card and Host each generate a unique $DESKey_m$ for each current MPEG program by performing an XOR operation on the CPKey and the ECM PID | LTSID for the program to be encrypted or decrypted and the copy protection scrambling algorithm used SHALL be triple-DES (ABA) in ECB mode.

The Card and Host SHALL generate and apply a single DESKey for each MPEG program, using the ECM-PID of the first elementary stream listed in the ca_pmt() message (see [CCIF 2.0]) for that MPEG program, or if none exists at elementary stream level, then the ECM-PID for the program.

4.4 Copy Control Information

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

The Card SHALL deliver a CCI value to the Host for every MPEG program with non-zero CCI immediately after the Host's ok_descrambling command, unless the Card rejects the command with an error indication. The Card SHALL initiate CCI delivery immediately after a change in CCI value for content it is CA-descrambling. The Card SHALL NOT deliver CCI for MPEG programs unless it has received the ok_descrambling command. The Card SHALL NOT CA-decrypt any MPEG program until commanded by the Host. The Card will deliver CCI after every change in the selected MPEG program or its associated CCI.

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

4.5 Scrambling of Copy Protected Content

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

After the Host is authenticated, the Card may CA-descramble content marked with zero EMI and return it to the Host without CP-scrambling.

When fully bound to an authenticated and validated Host, the Card may also CA-descramble High Value Content and SHALL CP-scramble it to protect the content as it returns to the Host across the CableCARD interface.

In Single stream mode (S-Mode), the CP-scrambling uses 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 scrambling 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:

- Clear: no change of an in-the-clear, zero EMI, MPEG program, which returns to the Host "in-the-clear"
- Pass-through: no change of a CA-scrambled MPEG program, returning it to the Host unrecognizable
- CA-only: CA-descrambles an MPEG program with zero EMI, which returns to the Host "in-the-clear"
- Rescramble: CA-descrambles and CP-scrambles an MPEG program with non-zero EMI

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 in this Section 4.5 above
- CA-encryption state and EMI value as defined in Section 10.2.

In Rescramble mode the Card SHALL CP-scramble using that MPEG program's unique DESKey as described in Section 10.4.

4.6 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. When this occurs, 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.

4.7 Card-Host Messages

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

4.8 Debug Modes Prohibited

Production Cards and Hosts 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-decrypt any MPEG program until it has successfully completed authentication of the Host by confirming a shared AuthKey.

5.1 Authentication Parameters

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

5.1.1 X.509 Certificates

The Card and Host each contain an X.509 version 3 Device Certificate [X.509] and a matching private key used for creating digital signatures. The certificate includes a unique ID and the public key provided to other devices to validate the digital signatures. Each device also has the Manufacturer 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

Card and Host manufacturers SHALL assign the Card_ID and Host_ID as follows:

Set the 24 most significant bits of the 64-bit Card_ID, bits numbered 63 through 40, to zero.

Set bits 39 through 30 to their CHICA assigned manufacturer number.

Set the 30 least significant bits, bits numbered 29 to 0, 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_C	The Card Diffie-Hellman public key. Derived and unique for each Card-Host binding.
DH_pubKeyH	The Host Diffie-Hellman public key. Derived and unique for each Card-Host binding.
DHKey	The Diffie-Hellman shared secret key.
AuthKeyC	The authentication key derived by the Card, before verifying that it equals AuthKeyH.
AuthKeyH	The authentication key derived by the Host, before verifying that it equals AuthKeyC.

AuthKey The shared public 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 Device Parameters in the Host Authentication

Parameter	Size (bits)
Diffie-Hellman Private Exponents (x,y)	160
Diffie-Hellman public keys (DH_pubKey$_C$ DH_pubKey$_H$)	1024
Diffie-Hellman shared secret key (DHKey)	1024
Authentication key (AuthKey)	160

5.2 Initialization

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

1. The Host SHALL report copy protection as a resource during the profile inquiry process. Upon such report, the Card SHALL disable CA decryption of all content.
2. The Card SHALL open a session to the copy protection resource. See Section 11.3.
3. The Card SHALL evaluate the Host's support for the Card-CP system. See Section 11.3.1.
4. If the Card contains a stored, non-zero, AuthKey in its non-volatile memory, it SHALL request the Host's AuthKey. If not, the Card SHALL record the non-authenticated and non-validated states in non-volatile memory and proceed with Diffie-Hellman public key calculation.
5. The Host SHALL respond with its AuthKey, if available. If it is not available, then it SHALL respond with a value of zero.
6. The Card SHALL compare the Host's AuthKey with its stored AuthKey.
7. If they are identical, authentication is complete, as restored from a previous binding. The Card SHALL store AuthKey in non-volatile memory and proceed with ID reporting. See Section 6.
8. 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.

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

1. The Card SHALL generate a random private exponent, x, where $1 \le x \le n - 2$.

2. The Host SHALL generate a random private exponent, y, where $1 \le y \le n - 2$.
3. The Card SHALL compute its DH public key as: $DH\_pubKey_C = g^x \bmod n$.
4. The Host SHALL compute its DH public key as: $DH\_pubKey_H = g^y \bmod n$.
5. The Card SHALL derive $SIGN_C$ by signing DH_pubKey_C with its X.509 private key.
6. The Host SHALL derive $SIGN_H$ by signing DH_pubKey_H with its X.509 private key.

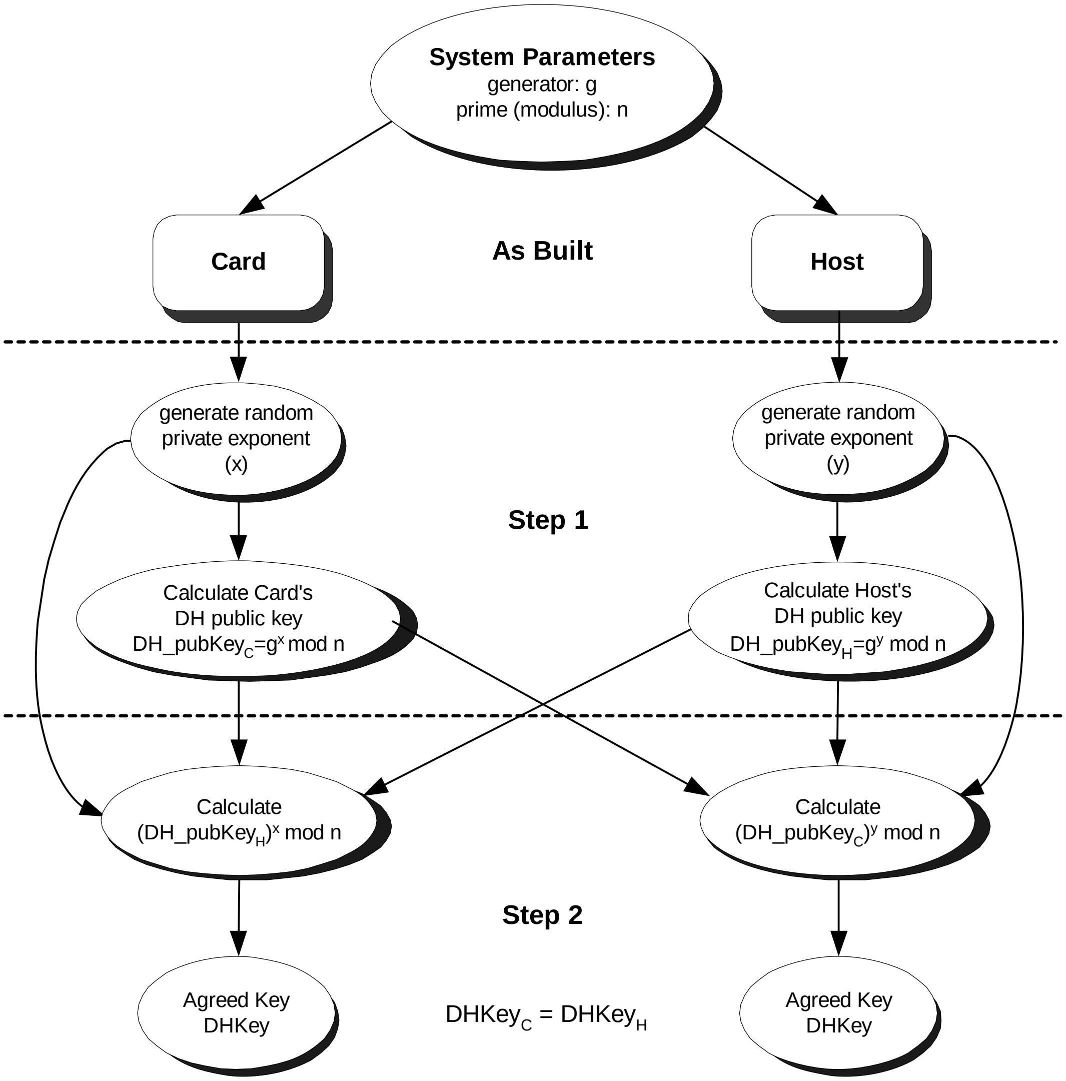

System Parameters
generator: g
prime (modulus): n
Card
As Built
Host
generate random private exponent (x)
generate random private exponent (y)
Step 1
Calculate Card's DH public key DH_pubKey$_C$=g^x mod n
Calculate Host's DH public key DH_pubKey$_H$=g^y mod n
Calculate (DH_pubKey$_H$)^x mod n
Calculate (DH_pubKey$_C$)^y mod n
Step 2
Agreed Key DHKey
DHKey$_C$ = DHKey$_H$
Agreed Key DHKey

Figure 5.3-1 - Diffie-Hellman Key Agreement

5.4 Authentication Parameter Exchange and Verification

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

1. The Card SHALL send its certificate data (Card_DevCert and Card_ManCert), DH_pubKey_C, and $SIGN_C$ to the Host with a request for the Host's authentication parameters.
2. The Host SHALL reply with its certificate data (Host_DevCert and Host_ManCert), DH_pubKey_H and $SIGN_H$.

3. The Card SHALL verify Host_DevCert, Host_ManCert, and SIGN$_H$ and extract the Host_ID from the Host Device Certificate.
4. The Host SHALL verify Card_DevCert, Card_ManCert, and SIGN$_C$ and extract the Card_ID from the Card Device Certificate.

5.5 Shared Binding Key Calculation

5.5.1 Diffie-Hellman Shared Secret Key

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

1. The Card derives the 1024 bit shared key DHKey = (DH_pubKeyH)x mod n; and
2. The Host derives the 1024 bit shared key DHKey = $(\text{DH\_pubKey}_C)^y$ mod n;

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

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

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

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

5.5.2 Shared Public Authentication Key

The Card and Host each compute the shared public key, AuthKey, from the shared secret DHKey, Card_ID, and Host_ID, as follows:

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

   $$\text{AuthKey}_C = \text{SHA-1 [DHKey | Host\_ID | Card\_ID]}$$

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

   $$\text{AuthKey}_H = \text{SHA-1 [DHKey | Host\_ID | Card\_ID]}$$

The Host SHALL store AuthKey$_H$ in non-volatile memory for CPKey generation and refresh.

5.5.3 Completion of Authentication

The Card SHALL request and the Host SHALL provide AuthKey$_H$. The Card SHALL compare AuthKey$_H$ to AuthKey$_C$. If they are identical, authentication is complete; the Card SHALL store AuthKey in non-volatile memory, and proceed with CPKey generation. Non-identical AuthKeys constitute a failure; see Section 5.6.3.

5.6 Authentication Failures

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

5.6.1 Host Response Time-out

If the Host fails to respond to any APDU within 5 seconds, the Card SHALL set the IIR flag. See [CCIF 2.0].

5.6.2 Invalid Certificate

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

```
CableCARD Certificate Invalid
```

Figure 5.6-1 - Example Card Authentication Failure Notification Message

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

```
Host Certificate Invalid
```

Figure 5.6-2 - Example Host Authentication Failure Notification Message

5.6.3 Other Authentication Failures

If any other part of the mutual authentication procedure described above fails, including signature or AuthKey verification, the Card SHALL disable all CA decryption even if the subscriber would otherwise be authorized to receive CA protected services. The Card SHALL then take the following steps:

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

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

5.7 Card Operation with Multiple Hosts

Each Card SHALL bind to exactly one Host at a time. No Card SHALL store two or more sets of Authentication Keys or other Host-specific information. A given 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 SHALL initiate the binding procedure, including full mutual authentication. If this Card is later returned to the previous Host, it SHALL again initiate the binding procedure, as it has authentication information only on the last Host to which it was bound.

6 ID REPORTING, VALIDATION, AND AUTHORIZATION

The Host_ID and Card_ID must be reported to the cable operator before the Card will provide High Value Content to the Host.

Following each authentication procedure, the Card SHALL check the reporting and validation status in non-volatile memory. If validation is complete, then Card-Host binding is complete and the Card SHALL perform its CA and CP functions.

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 following automated or manual means. The Card SHALL store Card_ID and Host_ID in non-volatile memory so they can be compared with the validated IDs received back from the headend.

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 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.

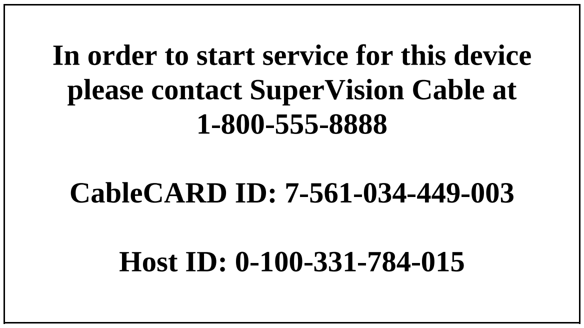
If the Host is in an off state or any non-video viewing state, it SHALL deny the MMI dialog open request. When the Host is in a video viewing state, it SHALL grant the open MMI dialog request. The Host SHALL display the message and confirm to the Card that the message has been displayed to the customer.

The Card SHALL 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, where:

- 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, see Annex A
- The ID digit sequence is normative. The hyphens and other text in Figure 6.1-1 are optional.

The ID Reporting Screen SHALL be displayed only if:

- The message is selected through a user menu system
- The user selects an MPEG program with CP active (non-zero EMI) before the binding is validated
- The Card initiates the message display, e.g., at the request of the CA System



In order to start service for this device please contact SuperVision Cable at 1-800-555-8888

CableCARD ID: 7-561-034-449-003

Host ID: 0-100-331-784-015

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 and authentication is complete, the Card SHALL display the same information and format defined above for Manual ID Reporting. If authentication is not complete, the Card SHALL display "Information not available."

In order to support a Host request to display the ID Reporting Screen, the Card SHALL support exactly one "ID Reporting application" with application_type = 0x01, CableCARD Binding Information Application, defined in [CCIF 2.0]. The ID Reporting application SHALL be defined strictly as display of the ID Reporting information screen as defined in this section above.

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 SHALL examine all previously reported Host_IDs and if there are any matches, it SHALL 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 the ID validation message and confirm the validated ID values match its current stored Card_ID and Host_ID. If they match, validation is complete: the Card SHALL enable CA decryption of authorized High Value Content. If they do not match, the Card SHALL continue to limit its CA decryption to content with EMI value 00.

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. In S-Mode a common CPKey is generated by the Card and Host from shared secrets and exchanged nonces.

In M-Mode the Card and Host each generate a common key pair, CPKeyA and CPKeyB, 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 10.5.2.

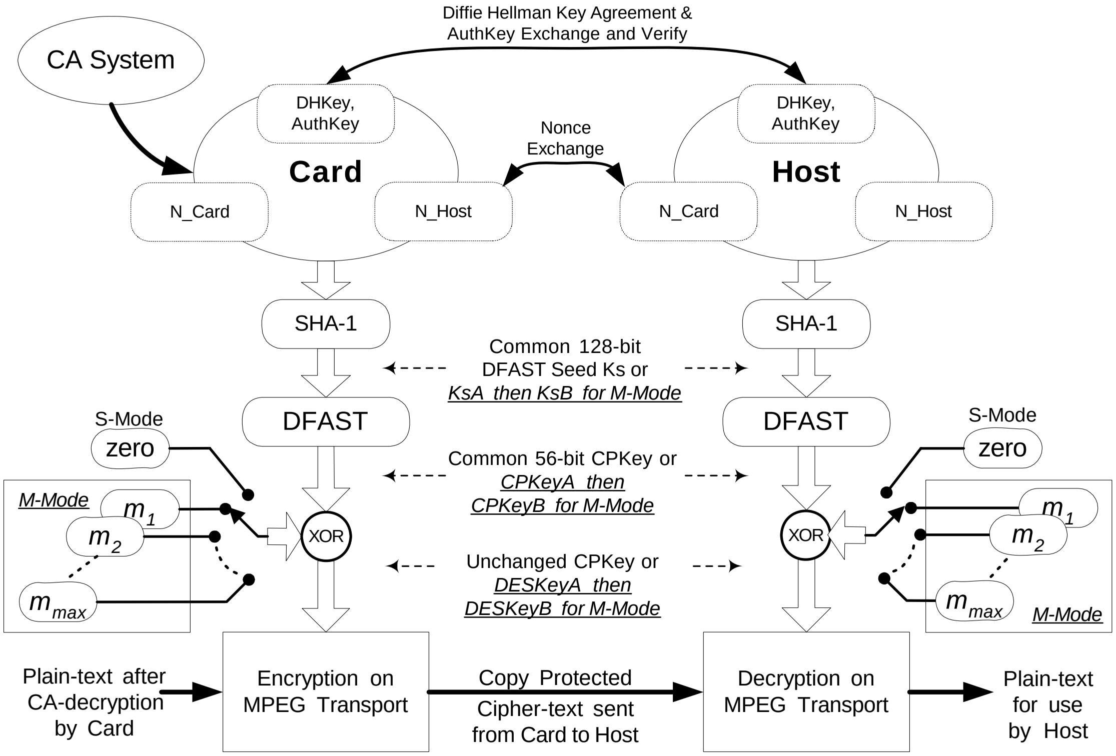


Figure 7–1 - Encryption Key Generation

The basic encryption-decryption methods are shown in Figure 7–2 and Figure 7–3 below. In S-Mode the single-DES ECB function uses the single CPKey for encryption and decryption. In M-Mode the triple-DES ECB function uses discrete DESKey-pairs for each MPEG-program for copy protection encryption and decryption.

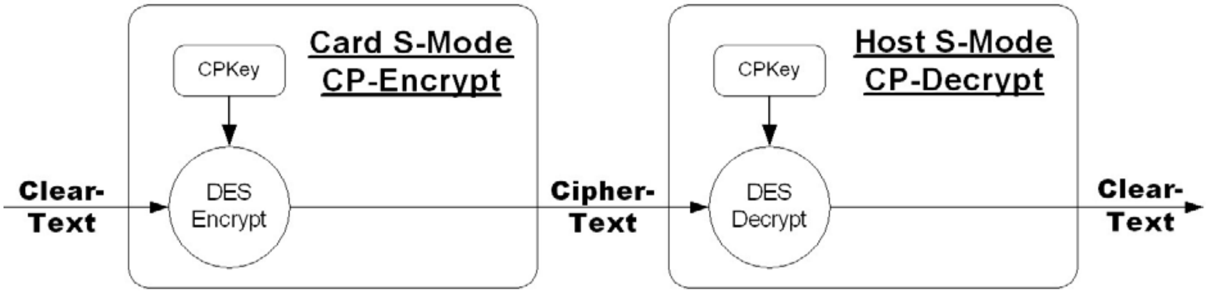


Figure 7–2 - S-Mode Encryption

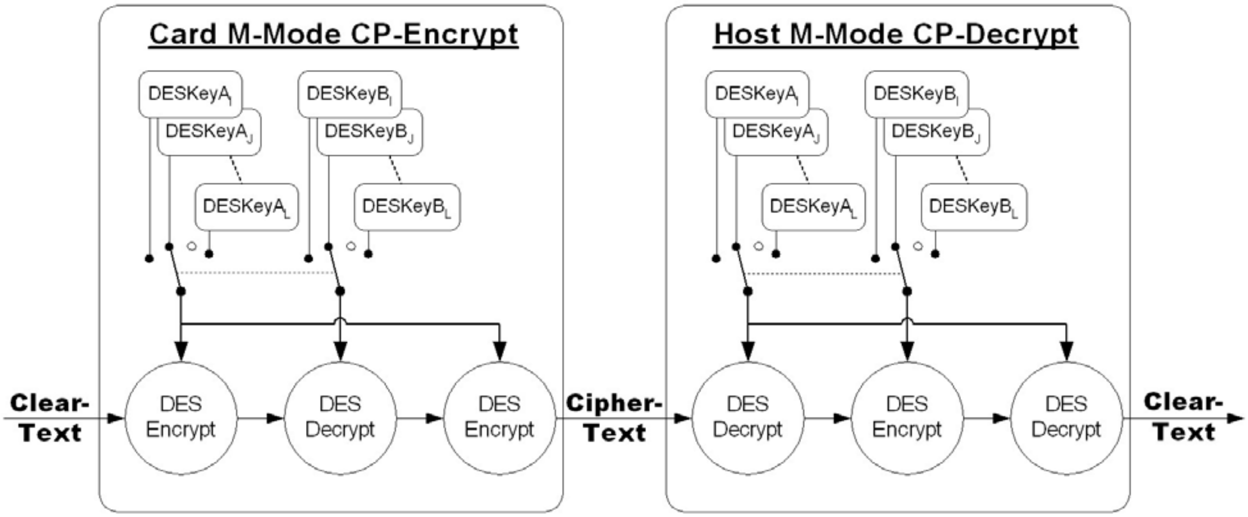


Figure 7–3 - M-Mode Encryption

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 (*Ks*) 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 $\text{DESKeyA}_{\text{m}}$ and $\text{DESKeyB}_{\text{m}}$ 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 and Host SHALL each implement a physical or a pseudorandom (software) integer generator. Pseudorandom generators SHALL be compliant with the SHA-1 based algorithm described in [FIPS 186-2], Appendix 3, Section 3.3, and include a unit-unique secret seed. Physical random integer generators SHALL comply with [FIPS 140-2], Section 4.7.1 test for randomness.

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 5.5.2.
- 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*8 bits should be converted into an octet block PV of length k, where PV_1, ..., 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

RSA digital signatures SHALL be computed using RSASSA-PKCS1-v1_5 as specified in [RSA1] with SHA-1 as the hash function.

The RSA public key exponent "e" has value 65537 decimal (0x01 0001).

RSA keys used to sign the Device Certificates and Manufacturer Certificates SHALL have modulus length of 2048 bits.

8 COPY PROTECTION KEY GENERATION

The Card SHALL generate and refresh the CPKey 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 reset.

Channel change SHALL NOT cause a key refresh to occur.

Highly randomized variables are used as new random integers ("nonces"). These nonces along with IDs are exchanged between the Card and Host interface. In S-Mode a common Copy Protection Key between the Card and Host is derived from these newly exchanged random integers, the Authentication Key (AuthKey$_c$ or AuthKey$_H$) and the 1024 bit shared secret Diffie-Hellman key (DHKey). The derived common Copy Protection Key (CPKey) is then used to generate DESKey to the DES-ECB processor which encrypts/decrypts content in MPEG-TS packets sent from the Card to the Host. In M-Mode, two base Copy Protection keys are derived (CPKeyA and CPKeyB). The Card and Host will derive unique DES keys, DES_KeyA[m] and DES_KeyB[m], for each copy protected program, where 'm' is defined in Section 10.5.2.

8.1 Basic Key Generation Protocol

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

1. The Card checks for a valid AuthKey. If such an AuthKey is not present, then follow the whole authentication process as detailed in Section 5.2.
2. The Card generates its 64 bit random integer (N_Card).
3. The Card sends this N_Card and Card_ID in the clear to the Host.
4. The Host generates its 64 bits random integer (N_Host).
5. The Host sends N_Host and its Host_ID in the clear to the Card.
6. The Card SHALL check that the received Host_ID is equal to the previously stored Host_ID. REQ976.7 If they are the same, Card SHALL proceed with the key generation process; otherwise, the Card SHALL CA decrypt only services with zero EMI.
7. The Card computes a CPKey for S-Mode, and CPKeyA and CPKeyB for M-Mode, based on long-term keys and newly exchanged random integer using the SHA-1 hash function and the DFAST algorithm, as described in the following section.
8. The Host computes a CPKey for S-Mode, and CPKeyA and CPKeyB for M-Mode, also based on long-term keys and newly exchanged random integer using the SHA-1 hash function and the DFAST algorithm, as described in the following section.

8.2 Copy Protection Key Calculation

The Card operating in S-Mode and Host SHALL each calculate the DFAST seed value, Ks, as:

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

The Card operating in M-Mode and Host SHALL each calculate the DFAST seed value, Ks, as:

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

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

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

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

For M-Mode, the Card and the Host SHALL apply the DFAST function to Ks to produce two Copy Protection Keys CPKeyA and CPKeyB:

$$\text{CPKeyA} = \text{DFAST} \, [\ \text{KsA}]$$

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

DFAST details are specified in a separate document; contact the CHICA. Table 8.2-1 defines the size of keys, as well as the parameters used to derive them.

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

Parameter	Size (bits)	Description
AuthKey	160 bits	Shared Public Authentication key, see Section 5.5.2
DHKey	1024 bits	Shared Secret Diffie-Hellman key, see Section 5.5.1
Nonces (N_Host, N_Card)	64 bits each	Random integers unique to each calculation of CPKey.
DFAST seed (Ks) or (KsA and KsB)	128 bits	Intermediate key for input to DFAST engine. (Ks) for S-Mode and (KsA and KsB) for M-Mode.
Copy Protection Key (CPKey) or (CPKeyA and CPKeyB)	56 bits	Final copy protection encryption and decryption key (CPKey) for S-Mode, and (CPKeyA and CPKeyB) for M-Mode.

8.3 CPKey Refresh

The Card SHALL periodically initiate refresh of the CPKey for S-Mode, and CPKeyA and CPKeyB for M-Mode. 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. In M-Mode, when a CPKey refresh occurs, both CPKeyA and CPKeyB SHALL be refreshed.

The Card SHALL initiate each CPKey refresh cycle and start a key refresh timer. The Card SHALL stop CP-scrambling during the synchronization of keys. It SHALL start to CP-scramble again on the earlier of 1) successful completion of the authenticated CPKey refresh cycle, or 2) transmitting unencrypted data for one second.

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

8.3.1 CPKey Session Timer

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

If this timer reaches the value of the max_key_session_period, the Card SHALL initiate a CPKey refresh. For S-Mode, the max_key_session_period is a 16-bit value with a resolution of 10 seconds (one decasecond). If the value of max_key_session_period is zero, the maximum CPKey session period is unlimited. For M-Mode, the max_key_session_period is a 16-bit value with a resolution of 60 seconds. If the value of max_key_session_period is zero, the maximum CPKey session period is unlimited. The Host is not aware of max_key_session_period.

The same key session period SHALL apply to the CPKey(s) on the interface, as it is a global parameter for the CPKey(s).

8.3.2 CPKey Refresh Timer

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

When the Host issues the CP_data_cnf(), the Card and Host SHALL start the calculation of the Copy Protection Key. The Card and Host SHALL each calculate CPKey within eight seconds when operating in S-Mode, and CPKeyA and CPKeyB 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 SHALL send the CP_sync_req() 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() message indicates that the Card has completed calculation of CPKey(s). The Host SHALL issue the CP_sync_cnf() message when it has received the CP_sync_req() message and has completed calculation of the Host CPKey when operating in S-Mode, and CPKeyA and CPKeyB and the resulting DES_keys when operating in M-Mode.

When the Card issues the CP_sync_req() message, the Card SHALL turn off all CP-scrambling and set 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. When the Key Refresh Timer reaches ten seconds when operating in S-Mode, and 30 seconds when operation in M-Mode, the Card SHALL immediately restore CP-scrambling with the new CP key. 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 SHALL disable CA decryption of all copy protected content (including other copy protected services on the interface) until the current or retry CP_Key refresh is completed (the Host has responded with a CP_sync_cnf() message). The Card SHALL commence scrambling of the MPEG packets with the new CP Key at the completion of the CP_Key refresh.

For S-Mode, Figure 8.3-1 illustrates the Card flow during a Key Refresh cycle. Figure 8.3-2 illustrates the Host flow during a Key Refresh cycle.

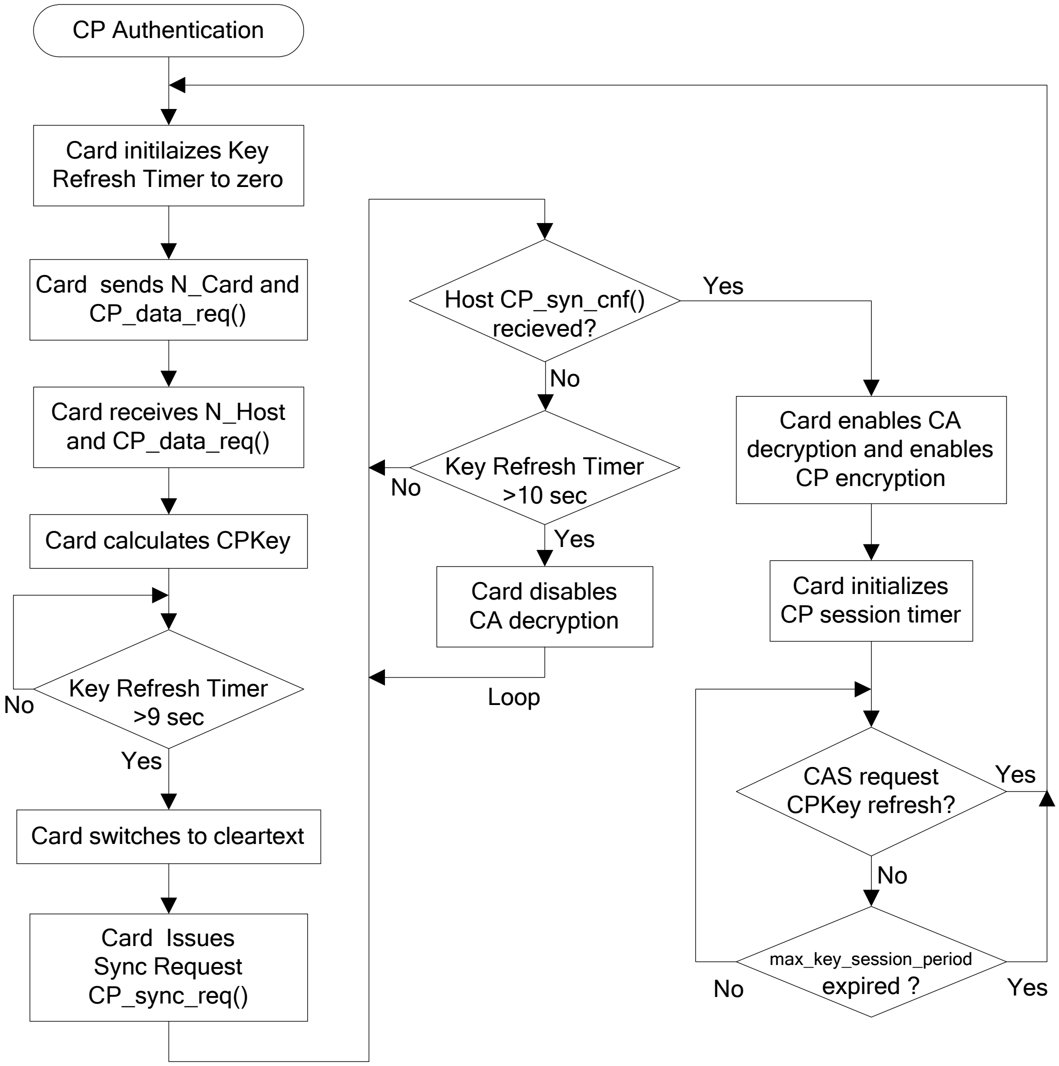

CP Authentication
Card initilaizes Key Refresh Timer to zero
Card sends N_Card and CP_data_req()
Card receives N_Host and CP_data_req()
Card calculates CPKey
Key Refresh Timer >9 sec
No
Yes
Card switches to cleartext
Card Issues Sync Request CP_sync_req()
Host CP_syn_cnf() recieved?
Yes
No
Key Refresh Timer >10 sec
No
Yes
Card disables CA decryption
Loop
Card enables CA decryption and enables CP encryption
Card initializes CP session timer
CAS request CPKey refresh?
Yes
No
max_key_session_period expired ?
No
Yes

Figure 8.3-1 - Card CPKey Refresh Session Flow Chart for S-Mode

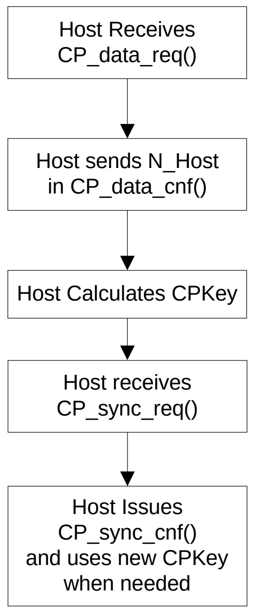


Figure 8.3-2 - Host CPKey Refresh Flow Chart for S-Mode

For M-Mode, Figure 8.3-3 illustrates the Card flow during a Key Refresh cycle. Figure 8.3-4 illustrates the Host flow during a Key Refresh cycle.

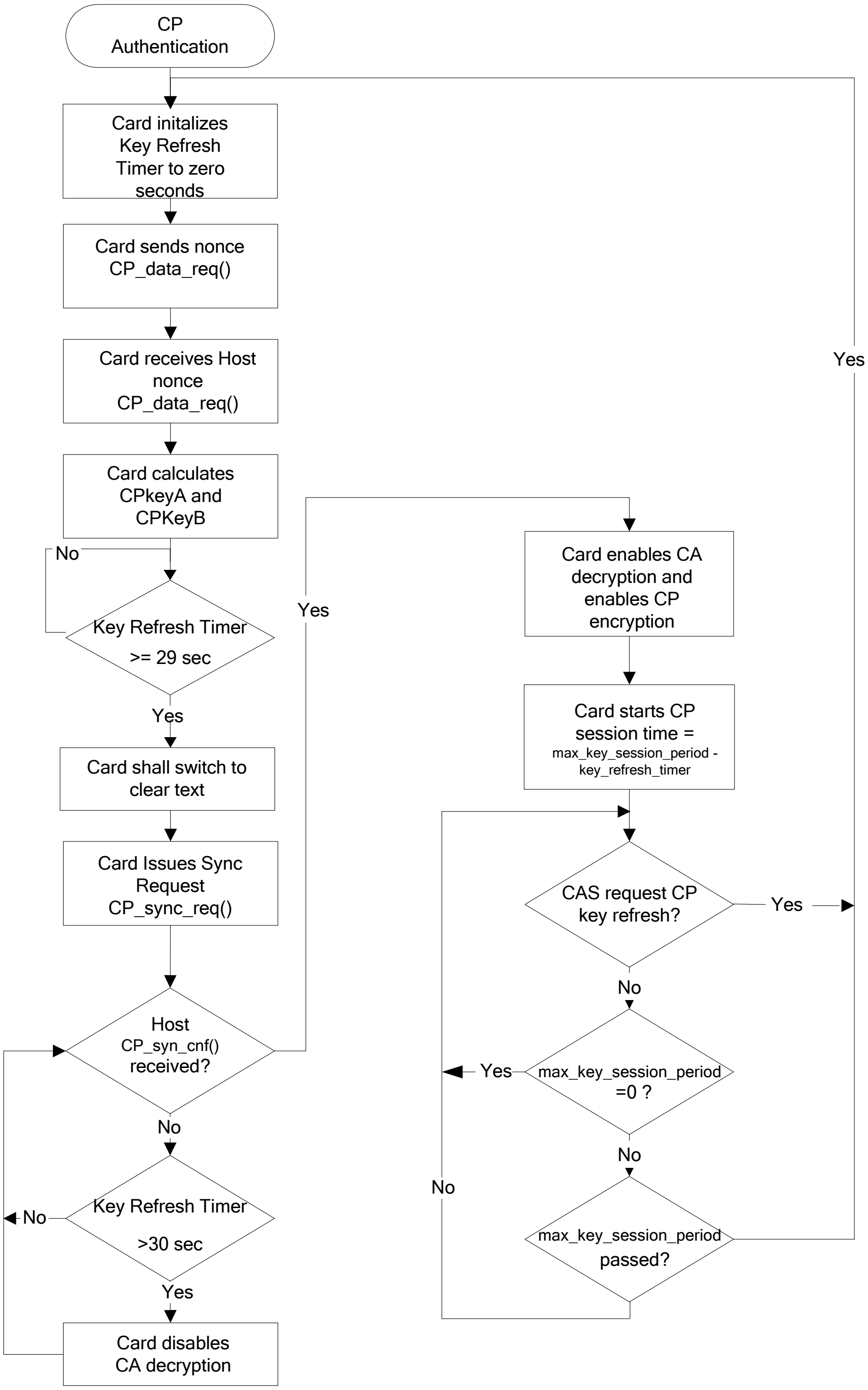

CP Authentication
Card initalizes Key Refresh Timer to zero seconds
Card sends nonce CP_data_req()
Card receives Host nonce CP_data_req()
Card calculates CPkeyA and CPKeyB
No
Key Refresh Timer >= 29 sec
Yes
Card shall switch to clear text
Card Issues Sync Request CP_sync_req()
Host CP_syn_cnf() received?
Yes
No
Key Refresh Timer >30 sec
No
Yes
Card disables CA decryption
Card enables CA decryption and enables CP encryption
Card starts CP session time = max_key_session_period - key_refresh_timer
CAS request CP key refresh?
Yes
Yes
No
max_key_session_period =0 ?
Yes
No
No
max_key_session_period passed?

Figure 8.3-3 - CPKey Refresh Session Flow Chart for M-Mode

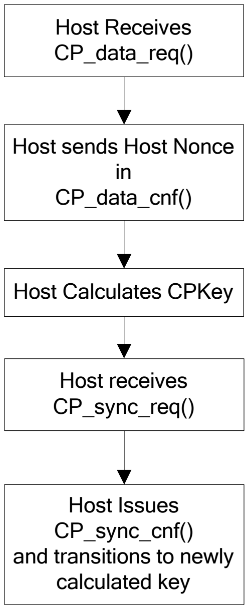


Figure 8.3-4 - Host CPKey Refresh Flow Chart for M-Mode

8.3.3 CA Initiated CPKey Refresh

The Card SHALL be capable of initiating a key refresh at the command of the CA System. The Card SHALL immediately initiate CPKey refresh 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 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 a single byte, 8-bit, field conveyed from Card to Host. Five of the eight bits are defined. The remaining three are reserved. The reserved bits SHALL be set to zero by the Card as shown in Table 9.1-1. The Host SHALL use the reserved bit values received from the Card only for execution of the protocol described below. The Host SHALL ignore the reserved 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	0	CIT	APS1	APS0	EMI1	EMI0
Host interprets as	rsvd	rsvd	rsvd	CIT	APS1	APS0	EMI1	EMI0

9.1.1 EMI - Digital Copy Control Bits

The two lsb of the CCI byte are the EMI bits. They SHALL control copy permissions for digital copies. The EMI bits SHALL be supplied to any Host digital output ports for control of copies made from those outputs. The EMI bits are defined in Table 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 to control copy protection encoding of analog outputs as described in Table 9.1-3 and detailed in [EIA 679].

Table 9.1-3 - APS Value Definitions

APS	Description
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 value of CIT as shown in Table 9.1-4.

Table 9.1-4 - CIT Value Definitions

CIT	Description
0b	No Image Constraint asserted
1b	Image Constraint required

9.2 Associating CCI with a Service

The CA System SHALL 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 means (e.g., an ECM) to transfer CCI from the headend to the Card. This delivery means SHALL 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, LTSID also.

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

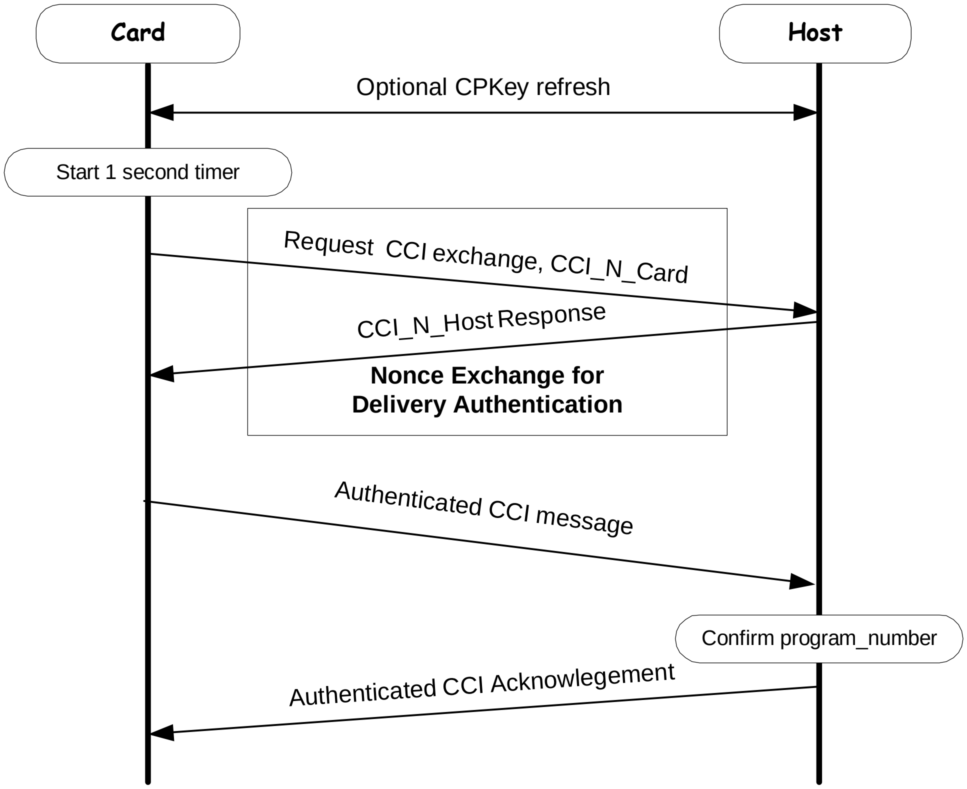


Figure 9.4-1 - CCI Delivery Sequence

9.4.1 CCI Delivery Instances

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

- The Host requests descrambling of an MPEG program, or
- The CCI bits change for any MPEG program that the Card is CA-descrambling.

9.4.2 CCI Delivery Protocol for Single and Multi-Stream Modes

Following any error or failure of the steps described below: A) the Card SHALL disable CA-descrambling of the associated MPEG program, and B) the Host shall set the Host Error CCI value for that MPEG program. The error is cleared by a successful completion of the CCI delivery protocol.

The Card shall check the status of CP Key refresh before initiating this CCI delivery protocol. When the Card is operating in S-Mode and the CPKey Refresh Timer is between 8 and 10 seconds the Card shall delay CCI transfer until CPKey refresh is complete. See informative Figure 9.4-2. When the Card is operating in M-Mode and the CPKey Refresh Timer is between 28 and 30 seconds the Card SHALL delay CCI transfer until the refresh for both CPKeyA and CPKeyB is complete. See informative Figure 9.4-3.

For each requested MPEG program:

Step 1. The Card SHALL generate a new random integer CCI_N_Card and starts a 1-second time-out.

Step 2. The Card SHALL send:

For S-Mode: CCI_N_Card, program_number, and a request for CCI_N_Host

For M-Mode: CCI_N_Card, program_number, LTSID, and a request for CCI_N_Host

Step 3. The Host SHALL generate a new random integer CCI_N_Host.

Step 4. The Host SHALL reply with:

For S-Mode: CCI_N_Host and program_number (received in step 2 above)

For M-Mode: CCI_N_Host, program_number and LTSID (received in step 2 above)

Step 5. The Card SHALL calculate two values: CCI_auth to authenticate CCI delivery, and CCI_ack to authenticate Host acknowledgment of receipt, as:

For S-Mode: CCI_auth = SHA-1(CCI | CPKey | CCI_N_Card | CCI_N_Host | program_number)

CCI_ack = SHA-1(CCI | CPKey | CCI_N_Card | CCI_N_Host)

For M-Mode (Type 4 Version 2): CCI_auth = SHA-1(CCI | CPKeyA| CPKeyB | CCI_N_Card | CCI_N_Host | program_number | LTSID | ECM-PID)

CCI_ack = SHA-1(CPKeyA| CPKeyB | CCI | CCI_N_Card | CCI_N_Host | program_number | LTSID | ECM-PID)

For M-Mode (Type 4 Version 3): CCI_auth = SHA-1(CCI | CPKeyA| CPKeyB | CCI_N_Card | CCI_N_Host | program_number | LTSID | ECM-PID')

CCI_ack = SHA-1(CPKeyA| CPKeyB | CCI | CCI_N_Card | CCI_N_Host | program_number | LTSID | ECM-PID')

The ECM-PID' is equal to the ECM-PID padded with 3 zeros (000) to the left, where ECM-PID is the same value used to generate DESKey for the MPEG program, see Section 4.3.

Step 6. The Card SHALL send CCI_auth, CCI, and program_number to the Host.

For S-Mode: CCI_auth, CCI, and program_number to the Host

For M-Mode: CCI_auth, CCI, program_number and LTSID to the Host

Step 7. The Host SHALL calculate CCI_auth using the received CCI value and compare it with the CCI_auth value received from the Card. If they do not match the Host SHALL treat the content as if a CCI value of 0x03 was received and return a zero value CCI_ack to the Card to indicate the failure.

Step 8. The Host SHALL control output of the associated MPEG program according to valid CCI within one second.

Step 9. The Host calculates CCI_ack and sends it to the Card.

Step 10. The Card compares the received CCI_ack with the value calculated in Step 5 above. If they match, the Card SHALL begin delivery of the associated MPEG program as indicated by the CCI. If they 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.

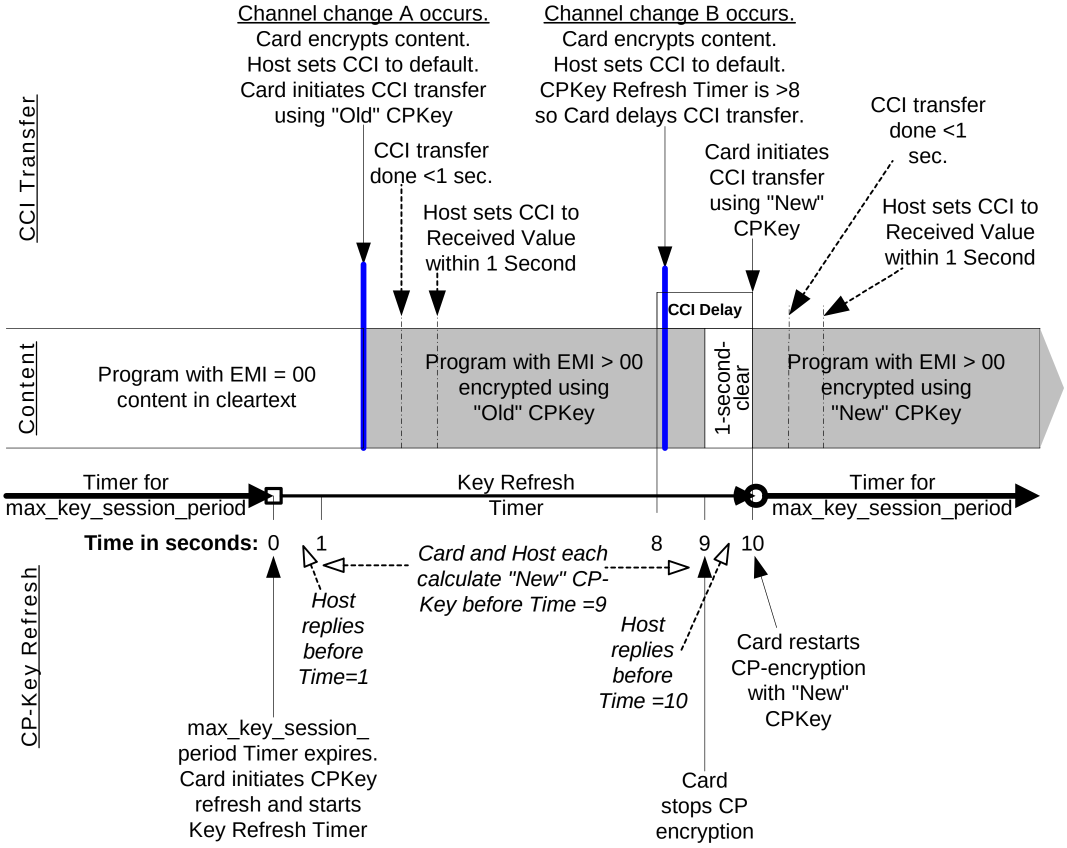


Figure 9.4-2 - Two Examples of CCI Transfer During CPKey Refresh in S-Mode

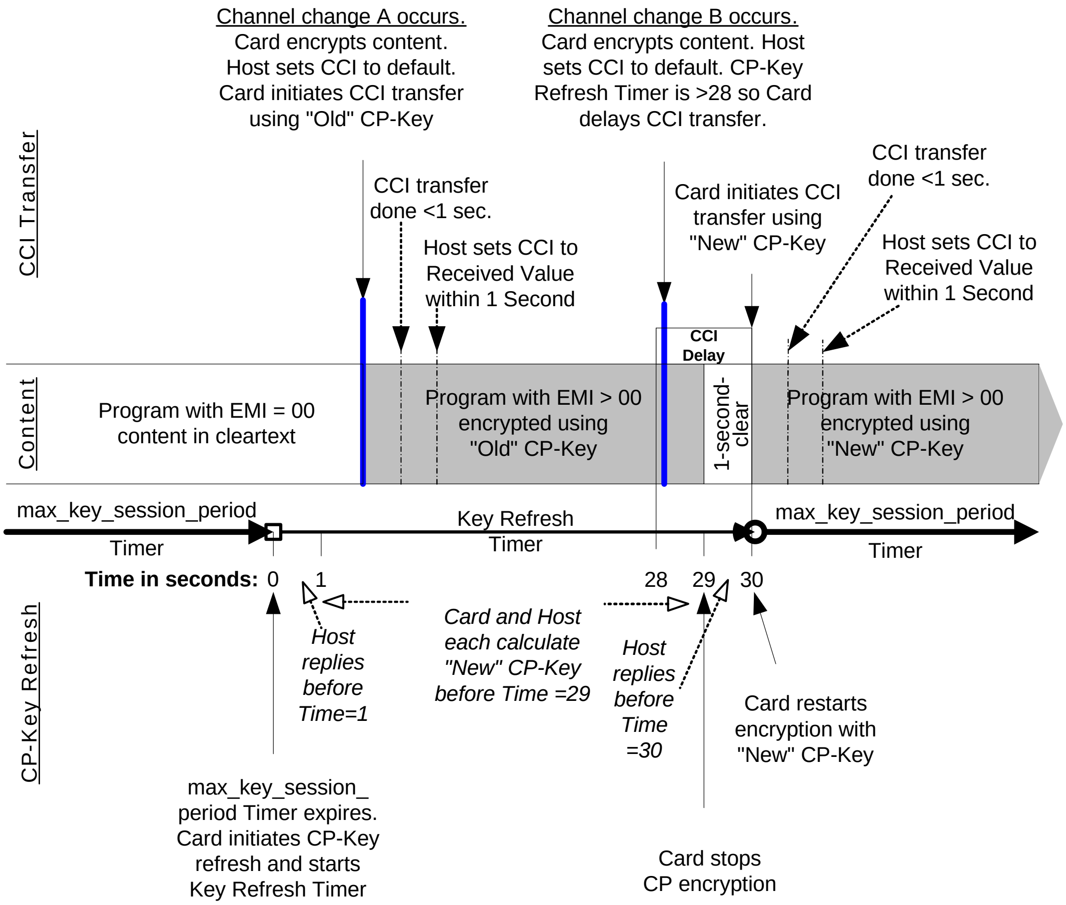


Figure 9.4-3 - Two Examples of CCI Transfer During CPKey Refresh in M-Mode (Informative)

9.4.3 Host Application of CCI

The Host SHALL apply content protection on analog and digital sources to set copy control parameters on Host outputs of content CA-descrambled 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 operating in S-Mode applies DES based encryption to protect High Value Content as it passes from Card to Host. The Card Operating in M-Mode applies ABA Triple DES ECB to protect non-zero EMI content as it passes from the Card to the Host.

10.1 Card Processing Modes

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

- Clear: no change of an in-the-clear, zero EMI, MPEG program, which returns to the Host "in-the-clear"
- Pass-through: no change of a CA-scrambled MPEG program, returning it to the Host unrecognizable
- CA-only: CA-descrambles an MPEG program with zero EMI, which returns to the Host "in-the-clear"
- Rescramble: CA-descrambles and CP-scrambles an MPEG program with non-zero EMI

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

When the Card is CA-authorized and fully bound to the Host, the Card SHALL apply copy protection scrambling to selected MPEG programs as shown in Table 10.2-1.

Table 10.2-1 - CP-Encryption Based on CA-Encryption and EMI Value

CA-Scrambling State	EMI Value	Card applies CP-Scrambling	Comments
In-the-clear	00b	No	
In-the-clear	01b, 10b, or 11b	No	Undesired*
Scrambled	00b	No	
Scrambled	01b, 10b, or 11b	Yes	

* Cable operators SHOULD CA-scramble all MPEG High Value Content programs. Only CA-Scrambled MPEG programs will be protected from unauthorized digital copying.

10.3 Scrambling Rules

- For S-Mode, DES-ECB SHALL be used to scramble all MPEG programs with non-zero EMI in the Card and to descramble them in the Host. Any residual blocks less than 64 bits in size SHALL be left in the clear.
- For M-Mode, ABA Triple DES ECB SHALL be used to scramble copy protected MPEG programs in the Card and to descramble them in the Host. Any residual blocks shall be handled as defined in Annex B.
- The Card SHALL encrypt only the payload portion of MPEG transport stream packets. LTSID packet preheaders, MPEG packet headers, and MPEG adaptation headers SHALL NOT be encrypted.
- The MPEG transport_scrambling_control bits output from the Card SHALL be set as described in Table 10.3-1.

- CA-scrambled but unauthorized services and CA-scrambled and authorized but unselected services SHALL pass through Card unaltered, remaining CA-scrambled and therefore useless to the Host.
- CP-scrambling SHALL only be applied to selected MPEG programs with non-zero EMI.
- The Card SHALL CP-scramble only authorized and selected MPEG programs. The Card SHALL immediately switch from rescramble mode to pass-through mode when the active MPEG program is deauthorized by the CA System.
- No data SHALL be double scrambled with both CA- and CP-scrambling.

The Card SHALL return MPEG programs with zero EMI to the Host in-the-clear. Such content may or may not be CA-scrambled during delivery from the headend to the Card.

10.3.1 Transport Scrambling Control Field

The transport_scrambling_control field of the MPEG transport packet provides control information for key changes.

Table 10.3-1 - 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

10.4 Timing of Scrambling Mode Transitions

The Card SHALL accomplish CP-scrambling mode changes quickly and in no case 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 as soon as possible and within 1.5 seconds following a change from zero EMI to non-zero EMI. CA System events, such as encryption management or control messages or changes in the CA-encryption mode, may affect CA-decryption by the Card. The Card SHALL continue to comply with all CP-scrambling requirements while responding to any such messages or mode changes.

10.5 DESKeys for Single and Multi-stream Modes

10.5.1 DESKey for Single Stream Mode

For S-Mode the Card SHALL encrypt the single selected MPEG program with DESKey_0 = CPKey.

10.5.2 DESKey for Multi-Stream Mode

For M-Mode the Card SHALL encrypt each copy protected MPEG program m using an individual DESKey_m where:

$m = \text{ECM-PID} \mid \text{LTSID}$

$\text{DESKeyA}_m = \text{CPKeyA} \oplus m$

$\text{DESKeyB}_m = \text{CPKeyB} \oplus m$

Where m has been padded with 35 left-most zero bits to match the length of CPKey.

Each instance of the DESKey_m SHALL be immediately available for use by the DES-ECB cryptographic elements in the Card and Host upon transmission of the CP_sync_cnf() APDU that follows a CPKey generation or refresh. After receipt of a new ca_pmt() APDU from the Host, the Card SHALL immediately apply the new values of ECM-PID or LTSID to the calculation of the DESKey_m used to encrypt the associated MPEG transport stream packets.

The Card SHALL immediately apply any changes in ECM-PID for the program or LTSID in the value of DESKey_m used to encrypt the affected MPEG transport stream packets.

11 CARD-HOST MESSAGING PROTOCOLS

11.1 Message Protocol Overview

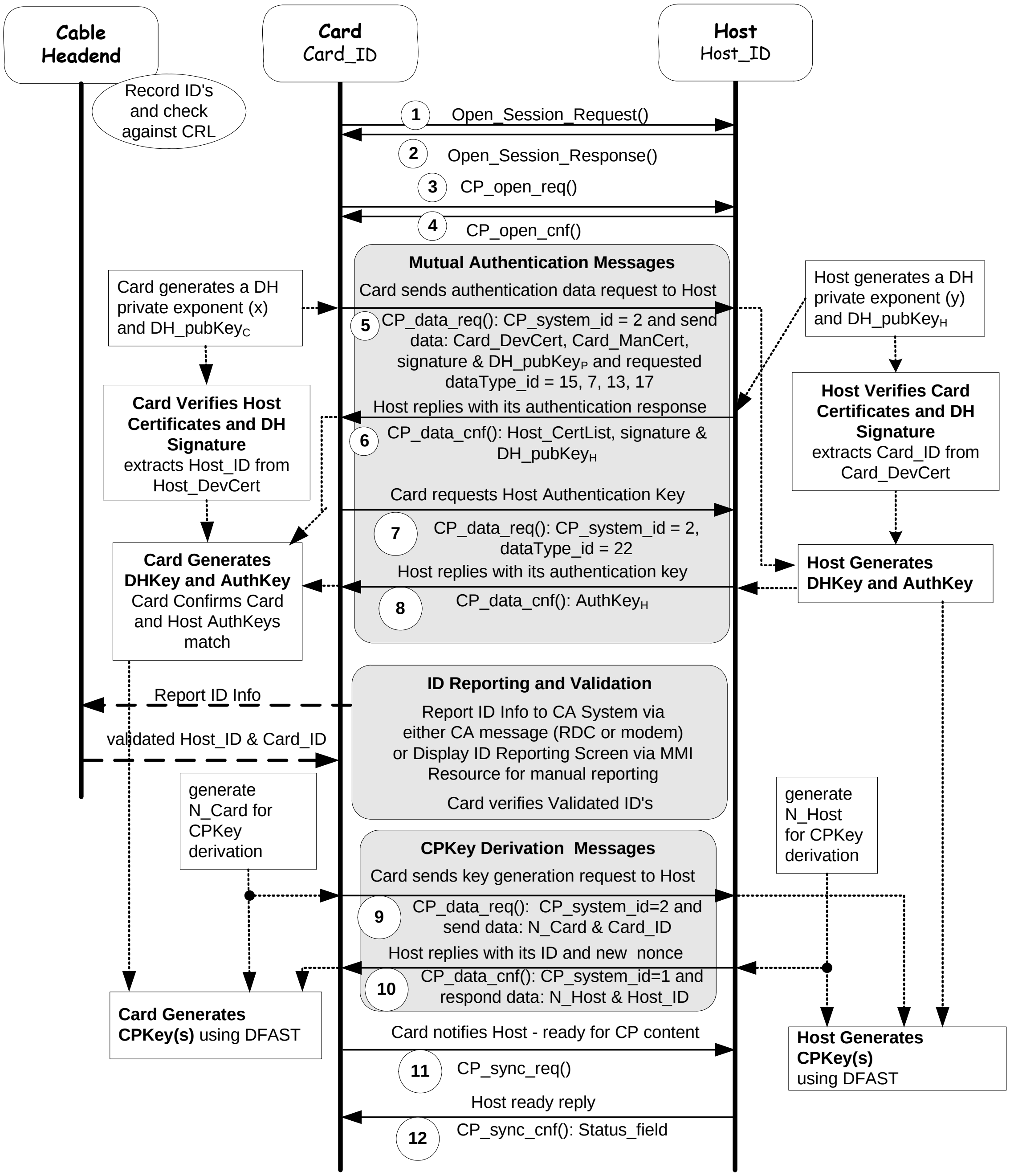

Cable Headend
Card
Card_ID
Host
Host_ID
Record ID's and check against CRL
1 Open_Session_Request()
2 Open_Session_Response()
3 CP_open_req()
4 CP_open_cnf()
Mutual Authentication Messages
Card sends authentication data request to Host
Card generates a DH private exponent (x) and DH_pubKey$_C$
Host generates a DH private exponent (y) and DH_pubKey$_H$
5 CP_data_req(): CP_system_id = 2 and send data: Card_DevCert, Card_ManCert, signature & DH_pubKey$_P$ and requested dataType_id = 15, 7, 13, 17
Host replies with its authentication response
Card Verifies Host Certificates and DH Signature extracts Host_ID from Host_DevCert
Host Verifies Card Certificates and DH Signature extracts Card_ID from Card_DevCert
6 CP_data_cnf(): Host_CertList, signature & DH_pubKey$_H$
Card requests Host Authentication Key
7 CP_data_req(): CP_system_id = 2, dataType_id = 22
Card Generates DHKey and AuthKey Card Confirms Card and Host AuthKeys match
Host Generates DHKey and AuthKey
Host replies with its authentication key
8 CP_data_cnf(): AuthKey$_H$
ID Reporting and Validation
Report ID Info
Report ID Info to CA System via either CA message (RDC or modem) or Display ID Reporting Screen via MMI Resource for manual reporting
validated Host_ID & Card_ID
Card verifies Validated ID's
generate N_Card for CPKey derivation
generate N_Host for CPKey derivation
CPKey Derivation Messages
Card sends key generation request to Host
9 CP_data_req(): CP_system_id=2 and send data: N_Card & Card_ID
Host replies with its ID and new nonce
10 CP_data_cnf(): CP_system_id=1 and respond data: N_Host & Host_ID
Card Generates CPKey(s) using DFAST
Host Generates CPKey(s) using DFAST
Card notifies Host - ready for CP content
11 CP_sync_req()
Host ready reply
12 CP_sync_cnf(): Status_field

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	Section 11.3	
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 (The CableCARD 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

Datatype_id	id value	Length (Bytes)
Reserved	1	
Reserved	2	

Datatype_id	id value	Length (Bytes)
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$_H$ (Host's DH public key)	13	128
DH_pubKey$_C$ (Card's DH public key)	14	128
Host_DevCert (Host's Device Certificate)	15	2048*
Card_DevCert (Card's Device Certificate)	16	2048*
SIGN$_H$ (the signature of Host's DH public key)	17	128
SIGN$_C$ (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$_H$ (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

* Certificates shorter than 2048 shall be padded to 2048 bytes by adding NULL bytes (0x00) at the trailing end. 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 a resource on its transport connection. Since the Host provides resources, it replies directly with a session number in its open session response.

Two objects defined at Session Protocol Data Unit (SPDU) layer, open_session_request() and open_session_response() are used here. Detailed SPDU data structure and other SPDU objects are defined in [CCIF 2.0].

Table 11.3-1 - Copy Protection Open Session Information

SPDU Tag / Object	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 SHALL then be used for all subsequent exchanges of messages (APDUs) between Card and Host.	Card ← Host

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

The resource_identifier SHALL match in both class and type of resource that the Host has in its list of available resources. Copy protection resource coding is listed in Table 11.3-2 below.

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	M-Mode	176	4	3	0x00B00103

If the version field of the supplied resource identifier is zero, then the Host SHALL use the current version in its list. If the version number in the request is less than or equal to the current version number in the Host's list, then the current version is used. If the requested version number is higher than the version in the Host's list, the Host SHALL refuse the request with the appropriate return code as defined in [CCIF 2.0].

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

11.3.1 Host Capability Evaluation

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

The Card SHALL use CP_open_req() and the Host SHALL use CP_open_cnf().

Table 11.3-3 - 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 APDU object is issued by the Card to query the Host's ability to support various copy protection systems.

Table 11.3-4 - Card's CP Support Request Message Syntax

Message Syntax	bits	bytes	Description
`CP_open_req () {`			
`CP_open_req_tag`	24	3	`Has the value: 0x9F 9000`

`Length_field()`	8	1	`Has the value: 0x00`
			`(length_field is defined in [CCIF 2.0], section 7. Because the message includes 0 bytes following this value the length is less than 127 bytes and size_indicator (the first bit of length_field) is 0. The length_value (the remaining 7-bits of length_field) is sufficient to indicate the length of 0, therefore length_field has the value: 0x00.)`

`}`			

11.3.1.2 CP_open_cnf( ) Syntax

This object is issued by the Host to the Card. If System 2 is not supported, the Card shall treat the Host as if its Device Certificate was invalid.

Table 11.3-5 - Host's CP Support Confirm Message Syntax

Message Syntax	bits	bytes	Description
`CP_open_cnf () {`			
`CP_open_cnf_tag`	24	3	`Has the value: 0x9F 9001`
`Length_field()`	8	1	`Has the value: 0x04`
`CP_system_id_bitmask`	32	4	`Values are listed in Table 11.3-6`
`}`			

Table 11.3-6 - CP_system_id_bitmask Values

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

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

11.4 Card-Host Mutual Authentication Message Protocol

11.4.1 Mutual Authentication Data Exchange Messages

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

Table 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's Authentication Data Message

The Card issues this APDU to send its authentication data to the Host. Card_DevCert, Card_ManCert, DH_pubKey$_C$ and SIGN$_C$ are included in this message.

Table 11.4-2 - Card's Authentication Data Message Syntax

Message Syntax	bits	bytes	Description
CP_data_req () {			
CP_data_req_tag	24	3	Has the value: 0x9F 9002
Length_field()	24	3	Has the value: 0x82 1113 (length of 4371 bytes)
			As defined in [EIA 679], section 7:
			size_indicator = 1 (1 bit, bslbf)
			length_field_size = 2 (7 bits, uimsbf)
			length_value_byte[0] = 17 (8 bits, bslbf)
			(most significant byte)
			length_value_byte[1] = 19 (8 bits, bslbf)
			(least significant byte)
CP_system_id	8	1	Has the value: 2 (CableCARD-CP System)
Send_datatype_nbr	8	1	Has the value: 4
For(i=0; i<Send_datatype_nbr; i++) {	(96)	(12)	
Datatype_ID	8	1	When i = 0, Datatype_ID value = 16 (Card_DevCert)
	8	1	When i = 1, Datatype_ID value = 8 (Card_ManCert)
	8	1	When i = 2, Datatype_ID value = 14 (DH_pubKey$_C$)
	8	1	When i = 3, Datatype_ID value = 18 (SIGN$_C$)
Datatype_length	16	2	When i = 0, Datatype_length value = 2048
	16	2	When i = 1, Datatype_length value = 2048
	16	2	When i = 2, Datatype_length value = 128
	16	2	When i = 3, Datatype_length value = 128
For (j=0; j<Datatype_length; j++) {		(4352)	
Data_type	16384	2048	When i = 0, Data_type = Card_DevCert
	16384	2048	When i = 1, Data_type = Card _ManCert
	1024	128	When i = 2, Data_type = DH_pubKey$_C$
	1024	128	When i = 3, Data_type = SIGN$_C$
}			
}			
Request_datatype_nbr	8	1	Has the value: 4
For(i=0; i<Request_datatype_nbr; i++) {	(32)	(4)	
Datatype_ID	8	1	When i = 0, Datatype_ID value = 15 (Host_DevCert)
	8	1	When i = 1, Datatype_ID value = 7 (Host_ManCert)
	8	1	When i = 2, Datatype_ID value = 13 (DH_pubKey$_H$)
	8	1	When i = 3, Datatype_ID value = 17 (SIGN$_H$)
}			
}			

11.4.1.2 Host's Authentication Data Message

The Host issues this APDU response to the Card with Host_DevCert, Host_ManCert, DH_pubKey$_H$ and SIGN$_H$.

Table 11.4-3 - Host's Authentication Data Message Syntax

Message Syntax	bits	bytes	Description
`CP_data_cnf () {`			
`CP_data_cnf_tag`	24	3	Has the value: 0x9F 9003
`Length_field()`	24	3	Has the value: 0x82 110E (length of 4366 bytes)
`CP_system_id`	8	1	Has the value: 2 CPS)
`Send_datatype_nbr`	8	1	Has the value: 4
`For(i=0; i<Send_datatype_nbr; i++) {`	(96)	(12)	
`Datatype_ID`	8 8 8 8	1 1 1 1	When i = 0, Datatype_ID = 15 (Host_DevCert) When i = 1, Datatype_ID = 7 (Host_ManCert) When i = 2, Datatype_ID = 13 (DH_pubKey$_H$) When i = 3, Datatype_ID = 17 (SIGN$_H$)
`Datatype_length`	16 16 16 16	2 2 2 2	When i = 0, Datatype_length = 2048 When i = 1, Datatype_length = 2048 When i = 2, Datatype_length = 128 When i = 3, Datatype_length = 128
`For (j=0; j<Datatype_length; j++) {`	34816	(4352)	
`Data_type`	16384 16384 1024 1024	2048 2048 128 128	When i = 0, Data_type = Host_DevCert When i = 1, Data_type = Host_ManCert When i = 2, Data_type = DH_pubKey$_H$ When i = 3, Data_type = SIGN$_H$
`}`			
`}`			
`}`			

11.4.2 AuthKey Verification Messages

Two objects, CP_data_req() and CP_data_cnf(), as defined at Application Protocol Data Unit (APDU) layer, are used for the Card 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$_H$	Card ← Host

11.4.2.1 Card's Request for Host AuthKey Message

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

Table 11.4-5 - Card's Request for Host AuthKey Message Syntax

Message Syntax	bits	bytes	Description
`CP_data_req () {`			
`CP_data_req_tag`	24	3	Has the value: 0x9F 9002
`length_field()`	8	1	Has the value: 0x04
`CP_system_id`	8	1	CP_system_id = 2
`Send_datatype_nbr`	8	1	Has the value: 0.
`Request_datatype_nbr`	8	1	Has the value: 1.
`For(i=0; i<Request_datatype_nbr; i++) {`	(8)	(1)	
`Datatype_ID`	8	1	Has the value: 22 (AuthKey$_H$).
`}`			
`}`			

11.4.2.2 Reply Message with Host's AuthKey

The Host issues this APDU to send its authentication key (AuthKey$_H$) to the Card.

Table 11.4-6 - Host's Reply with AuthKey Message Syntax

Message Syntax	bits	bytes	Description
`CP_data_cnf () {`			
`CP_data_cnf_tag`	24	3	Has the value: 0x9F 9003
`length_field()`	8	1	Has the value: 0x19
`CP_system_id`	8	1	Has the value: 2
`Send_datatype_nbr`	8	1	Has the value: 1
`For(i=0; i<Send_datatype_nbr; i++) {`	(16)	(2)	
`Datatype_ID`	8	1	Has the value: 22 (AuthKey$_{H,}$)
`Datatype_length`	16	2	Has the value: 20
`For (j=0; j<Datatype_length; j++)`			
`{ Data_type`	160	20	Data_type = AuthKey$_H$
`}`			
`}`			
`}`			

11.5 Copy Protection Key Generation

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

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

Table 11.5-2 - Card's CPKey Generation Message Syntax

Message Syntax	bits	bytes	Description
`CP_data_req () {`			
`CP_data_req_tag`	24	3	Has the value: 0x9F 9002
`Length_field()`	8	1	Has the value: 0x1B
`CP_system_id`	8	1	Has the value: 2
`Send_datatype_nbr`	8	1	Has the value: 2
`For(i=0; i<Send_datatype_nbr; i++) {`	(48)	(2*3)	
`Datatype_ID`	8	1	When i = 0, Datatype_id = 6 (Card_ID)
	8	1	When i = 1, Datatype_id = 12 (N_Card)
`Datatype_length`	16	2	When i = 0, Datatype_length = 8
	16	2	When i = 1, Datatype_length = 8
`For (j=0; j<Datatype_length; j++)`	(128)	(16)	
`{ Data_type`	64	8	When j = 0, Data_type = Card_ID
	64	8	When j = 1, Data_type = N_Card;
`}`			
`}`			
`Request_datatype_nbr`	8	1	Has the value: 2
`For(i=0;i<Request_datatype_nbr; i++)`	(16)	(2*1)	
`{ Datatype_id`	8	1	When i = 0, Datatype_id = 5 (Host_ID)
	8	1	When i = 1, Datatype_id = 11 (N_Host)
`}`			
`}`			

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

Message Syntax	bits	bytes	Description
`CP_data_cnf () {`			
`CP_data_cnf_tag`	24	3	Has the value: 0x9F 9003
`Length_field()`	8	1	Has the value: 21
`CP_system_id`	8	1	Has the value: 2 (see `Table 11.2-1`)
`Send_datatype_nbr`	8	1	Has the value: 2
`For(i=0; I<Send_datatype_nbr; i++)`	(48)	(2*3)	
`{ Datatype_id`	8	1	When i = 0, Datatype_id= 5 (Host_ID)
	8	1	When i = 1, Datatype_id=11 (N_Host)
`Datatype_length`	16	2	When i = 0, Datatype_length = 5
	16	2	When i = 1, Datatype_length = 8
`(j=0; j<Datatype_length; j++)`	(104)	(13)	
`{Data_type`	40	5	When i = 0, Data_type = Host_ID
	64	8	When i = 1, Data_type = N_Host
`}`			
`}`			
`}`			

11.6 Card-Host CPKey Synchronization

The Card notifies the Host of its intention to start CP encryption of protected MPEG programs based on the new CPKey by sending CP_sync_req(). The Host replies when it is ready with CP_sync_cnf().

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

Table 11.6-2 - Card's CPKey Ready Message Syntax

Message Syntax	bits	bytes	Description
CP_sync_ req () {			
CP_sync_req_tag	24	3	Has the value: 0x9F 9004
Length_field()	8	1	Has the value: 0x00
}			

Table 11.6-3 - Host's CPKey Ready Message Syntax

Message Syntax	bits	bytes	Description
CP_sync_cnf () {			
CP_sync_req_tag	24	3	Has the value: 0x9F 9005
Length_field()	8	1	Has the value: 0x01
Status_field	8	1	Values are listed in Table 11.6-4
}			

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

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

Table 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 ***and in M-Mode also the LTSID***.	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, ***and in M-Mode also the LTSID***.	Card ← Host
CP_data_req	0x9F 9002	Card sends the CCI (CCI_data), the program number and the calculated message digest (CCI_auth) ***and in M-Mode also the LTSID.***	Card → Host
CP_data_cnf	0x9F 9003	Host replies to Card with CCI_ack and program number, ***and in M-Mode also the LTSID***.	Card ← Host

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

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

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

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

Table 11.7-4 - CCI Delivery Message Syntax

Message Syntax	bits	bytes	Description
CP_data_req(){			
CP_data_req_tag	24	3	Has the value: 0x9F 9002.
length_field()	8	1	For S-Mode: value = 37, ***For M-Mode value = 42***
CP_system_id	8	1	Has the value: 2
Send_datatype_nbr	8	1	For S-Mode value =3. ***For M-Mode value = 4.***
for(i=0; i<Send_datatype_nbr; i++)			
{ Datatype_id	8	1	For i = 0, Datatype_id = 25 (CCI_data)
	8	1	For i = 1, Datatype_id = 26 (program_number)
	8	1	For i = 2, Datatype_id = 27 (CCI_auth)
	8	***1***	***For M-Mode only: i = 3, Datatype_id = 29 (LTSID)***
Datatype_length	16	2	For i = 0, Datatype_length = 1
	16	2	For i = 1, Datatype_length = 2
	16	2	For i = 2, Datatype_length = 20
	16	***2***	***For M-Mode only: i = 3, Datatype_length = 1***
for (j=0; j<Datatype_length; j++)			
{ Data_type	8	1	For i = 0, Data_type = CCI_data.
	16	2	For i = 1, Data_type = program_number.
	160	20	For i = 2, Data_type = CCI_auth.
	8	***1***	***For M-Mode only: i = 3, Data_type = LTSID.***
}			
}			
Request_datatype_nbr	8	1	For S-Mode value = 2. ***For M-Mode value = 3.***
for(i=0; i<Request_datatype_nbr; i++)			
{ Datatype_id	8	1	For i=0, Datatype_id = 28 (CCI_ack)
	8	1	For i=1, Datatype_id = 26 (program_number)
	8	***1***	***For M-Mode only: i=2, Datatype_id = 29(LTSID)***
}			
}			

Table 11.7-5 - CCI Acknowledgement Message Syntax

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

11.8 Card Validation Status

The Host requests the Card validation status by sending CP_valid_req(). The Card replies when it is ready with CP_valid_cnf(). After the first time the Host requests Card validation status, the Card SHALL send CP_valid_cnf() to the Host unsolicited, whenever the CP_status value changes. CP_valid_cnf() SHALL NOT be sent unsolicited to the Host until after CP_valid_req() has been received one or more times.

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 2)

Message Syntax	bits	bytes	Description
CP_valid req () {			
CP_valid_req_tag	24	3	Has the value: 0x9F 9006
Length_field()	8	1	Has the value: 0x00
}			

Table 11.8-3 - Card Validation Status Reply Message Syntax (type 4 ver 2)

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

Status_field SHALL return the status of the CP_valid_req() as indicated 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_H	0x03
Not bound, failed to match AuthKey from Host Device	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 → 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 = $K_1\ K_2\ K_3 \ldots K_{56}$ 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} = $K_1\ K_2 \ldots K_7\ \mathbf{P_1}\ \ K_8 \ldots K_{14}\ \mathbf{P_2}\ \ \ldots\ \ldots\ \ K_{50} \ldots K_{56}\ \mathbf{P_8}$

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= 0000000 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} = 0000000**1** 1001000**1** 1101000**0** 1010110**1** 0111100**1** 0100110**0** 1010111**0** 1001101**1** $b_{)}$
= 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 ff ff ...
E: 47 1f ff 10 ff ff ff ff ff ff ff ff ff ff ff ff ff ...
```

CP encryption leaves the packets that don't belong to a copy protected 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** (4th byte: 0x**1**c to 0x**d**c). 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 ($0 \le N \le 184$)
b	Cipher block size, in bytes. b = 8 for the 3DES cipher.
n	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. ($0 \le p < b$)

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 $3DES^{-1}_K$ denotes the inverse cipher operation (decryption). Heavy arrows indicate data flow, with the data size as indicated.

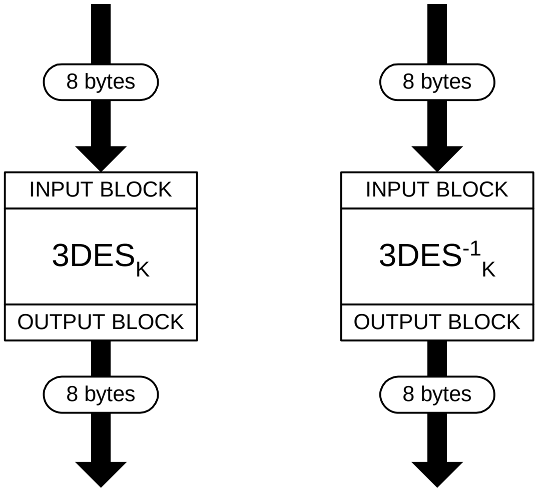


Figure B.3–1 - 3DES Encryption and Decryption, ECB mode

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.)

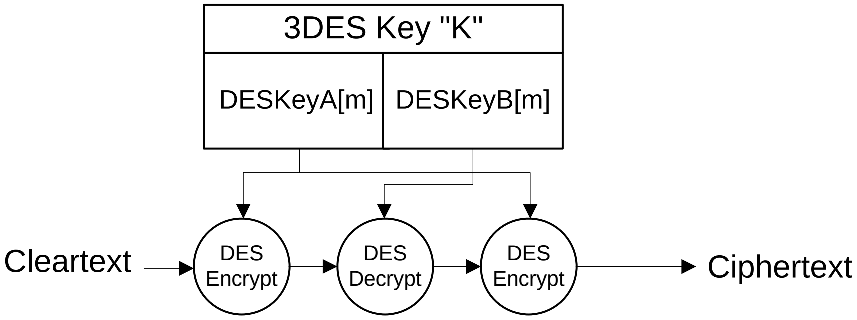


Figure B.3–2 - 2-key Triple DES, EDE-121 mode, Encryption

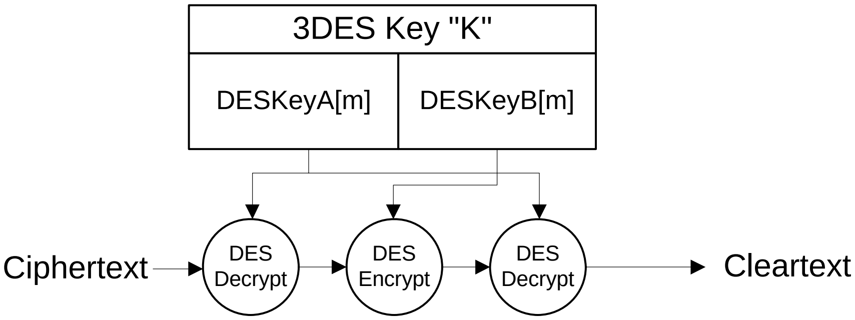


Figure B.3–3 - 2-key Triple DES, EDE-121 mode, Decryption

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 (n)	Residual bytes (p)	Payload length
Case 1	n > 0	p = 0	Integral number of full cipher blocks
Case 2	n > 0	1 ≤ p ≤ 7	At least one full cipher block, plus some residual bytes
Case 3	n = 0	1 ≤ p ≤ 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 = n * b for n = 1, 2, …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.

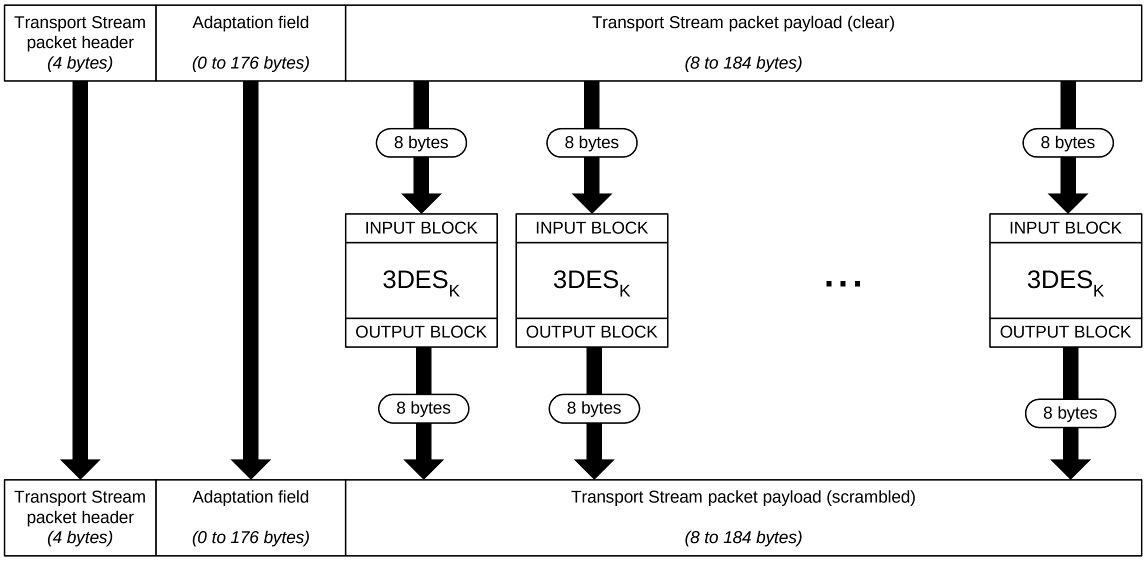

Transport Stream packet header (4 bytes)
Adaptation field (0 to 176 bytes)
Transport Stream packet payload (clear) (8 to 184 bytes)
8 bytes
8 bytes
8 bytes
INPUT BLOCK
3DES_K
OUTPUT BLOCK
INPUT BLOCK
3DES_K
OUTPUT BLOCK
...
INPUT BLOCK
3DES_K
OUTPUT BLOCK
8 bytes
8 bytes
8 bytes
Transport Stream packet header (4 bytes)
Adaptation field (0 to 176 bytes)
Transport Stream packet payload (scrambled) (8 to 184 bytes)

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

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

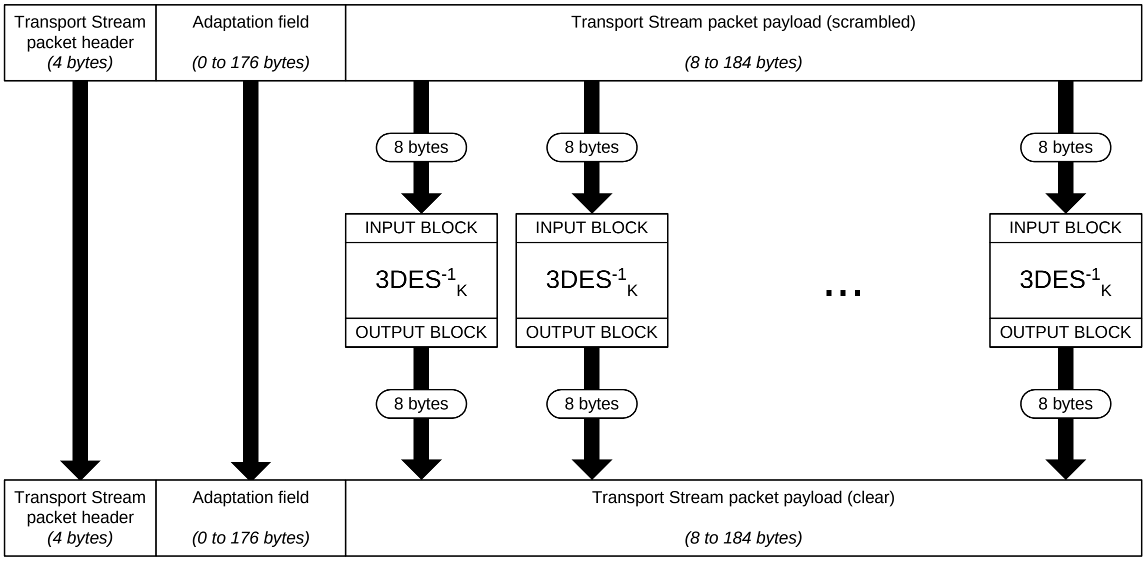

Transport Stream packet header (4 bytes)
Adaptation field (0 to 176 bytes)
Transport Stream packet payload (scrambled) (8 to 184 bytes)
8 bytes
INPUT BLOCK
3DES-1K
OUTPUT BLOCK
8 bytes
...
Transport Stream packet header (4 bytes)
Adaptation field (0 to 176 bytes)
Transport Stream packet payload (clear) (8 to 184 bytes)

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

$N = n * b + p$ for $1 \leq n \leq 22$, and $1 \leq p \leq 7$

The Transport Stream packet payload, starting from the first byte, is divided into (n-1) 8-byte blocks, one partial block p bytes in length, and one final 8-byte block. The (n-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.

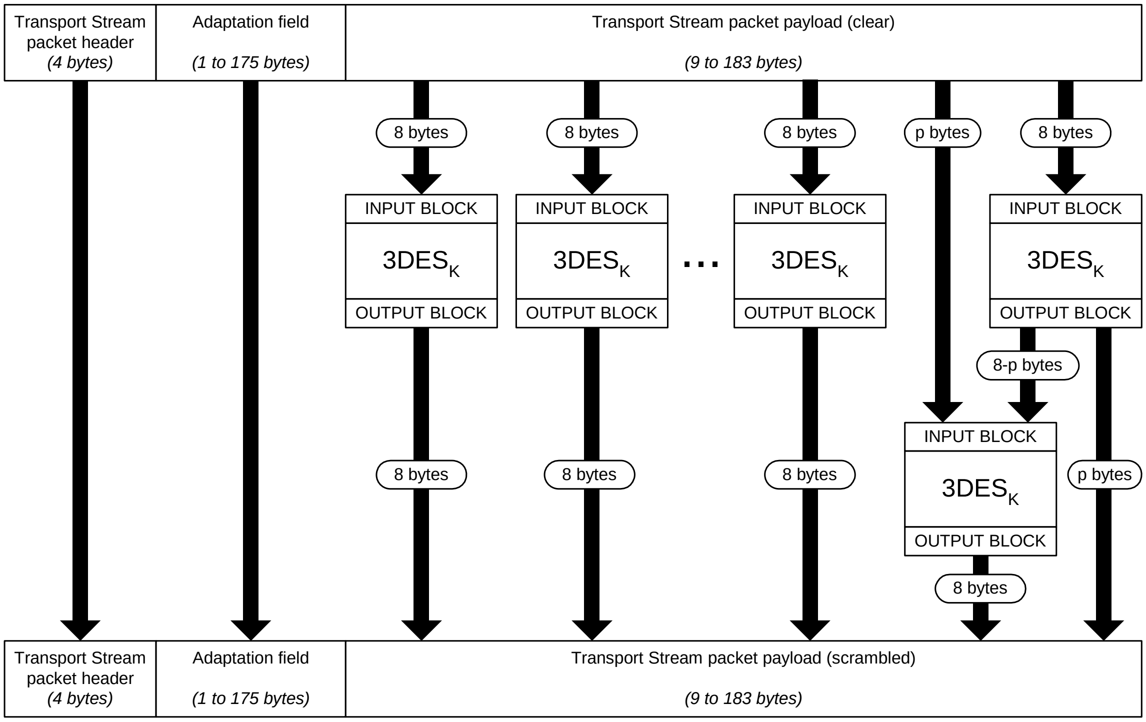

Transport Stream packet header (4 bytes)
Adaptation field (1 to 175 bytes)
Transport Stream packet payload (clear) (9 to 183 bytes)
8 bytes
p bytes
INPUT BLOCK
3DESK
OUTPUT BLOCK
...
8-p bytes
8 bytes
p bytes
Transport Stream packet header (4 bytes)
Adaptation field (1 to 175 bytes)
Transport Stream packet payload (scrambled) (9 to 183 bytes)

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.

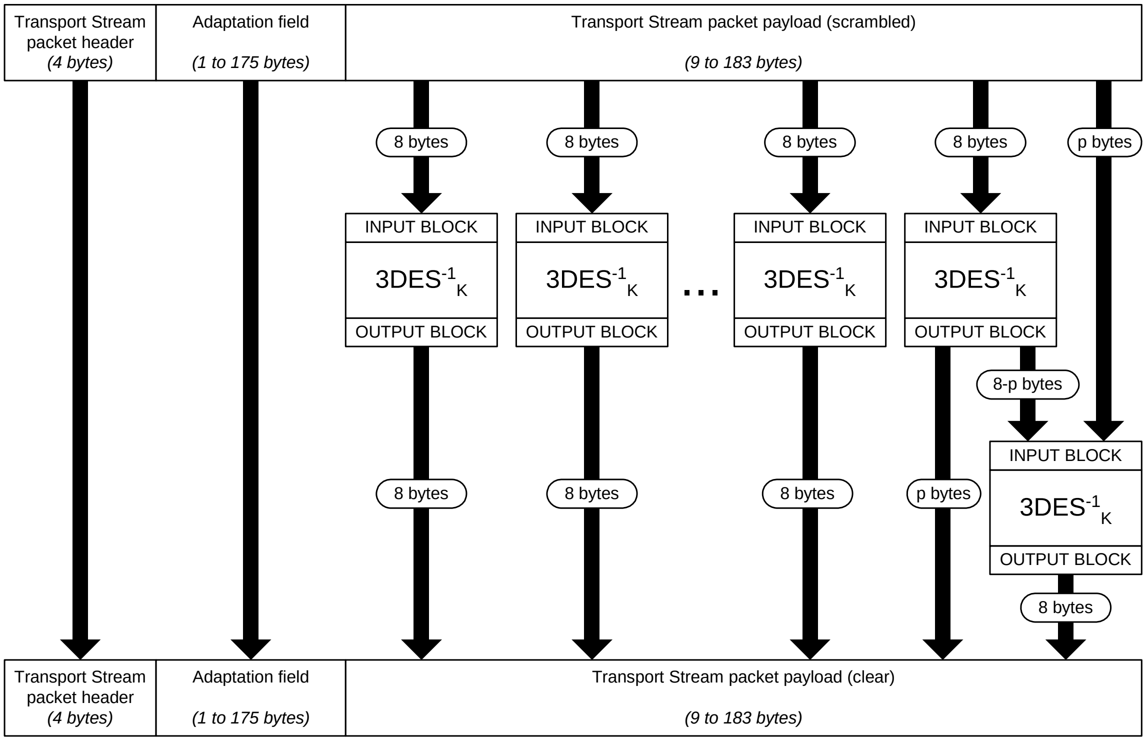


Figure B.3–7 - MPEG Transport Packet Descrambling, Case 2

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

N = p for $1 \le p \le 7$

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]} = [8989ababcdcdefef, 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 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:

```
47 00 40 30 b2 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 01 02 03 04 05
```

Encrypted:

```
47 00 40 b0 b2 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
```

```
00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 01 02 03 04 05
```

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