

ITU-T

TELECOMMUNICATION
STANDARDIZATION SECTOR
OF ITU

J.1033

(08/2020)

SERIES J: CABLE NETWORKS AND TRANSMISSION
OF TELEVISION, SOUND PROGRAMME AND OTHER
MULTIMEDIA SIGNALS

Conditional access and protection – Downloadable
conditional access system for bidirectional networks

Downloadable conditional access system for bidirectional networks – The terminal

Recommendation ITU-T J.1033

Recommendation ITU-T J.1033

Downloadable conditional access system for bidirectional networks – The terminal

Summary

Recommendation ITU-T J.1033 specifies a terminal for the two-way downloadable conditional access system (DCAS) for bidirectional networks. A two-way DCAS protects broadcast content/services and controls consumer entitlements in the same way as what traditional conditional access (CA) systems do, and enables a two-way terminal device, such as a set-top-box (STB), to adapt to a new CA system by downloading and installing a new CA system's client software without changing hardware. In particular, a two-way DCAS can work in bidirectional cable television networks and other bidirectional networks such as broadband cable networks.

History

Edition	Recommendation	Approval	Study Group	Unique ID*
1.0	ITU-T J.1033	2020-08-13	9	11.1002/1000/14356

Keywords

Bidirectional network, CA, DCAS, downloadable.

* To access the Recommendation, type the URL <http://handle.itu.int/> in the address field of your web browser, followed by the Recommendation's unique ID. For example, <http://handle.itu.int/11.1002/1000/1114356>.

FOREWORD

The International Telecommunication Union (ITU) is the United Nations specialized agency in the field of telecommunications, information and communication technologies (ICTs). The ITU Telecommunication Standardization Sector (ITU-T) is a permanent organ of ITU. ITU-T is responsible for studying technical, operating and tariff questions and issuing Recommendations on them with a view to standardizing telecommunications on a worldwide basis.

The World Telecommunication Standardization Assembly (WTSA), which meets every four years, establishes the topics for study by the ITU-T study groups which, in turn, produce Recommendations on these topics.

The approval of ITU-T Recommendations is covered by the procedure laid down in WTSA Resolution 1.

In some areas of information technology which fall within ITU-T's purview, the necessary standards are prepared on a collaborative basis with ISO and IEC.

NOTE

In this Recommendation, the expression "Administration" is used for conciseness to indicate both a telecommunication administration and a recognized operating agency.

Compliance with this Recommendation is voluntary. However, the Recommendation may contain certain mandatory provisions (to ensure, e.g., interoperability or applicability) and compliance with the Recommendation is achieved when all of these mandatory provisions are met. The words "shall" or some other obligatory language such as "must" and the negative equivalents are used to express requirements. The use of such words does not suggest that compliance with the Recommendation is required of any party.

INTELLECTUAL PROPERTY RIGHTS

ITU draws attention to the possibility that the practice or implementation of this Recommendation may involve the use of a claimed Intellectual Property Right. ITU takes no position concerning the evidence, validity or applicability of claimed Intellectual Property Rights, whether asserted by ITU members or others outside of the Recommendation development process.

As of the date of approval of this Recommendation, ITU had not received notice of intellectual property, protected by patents, which may be required to implement this Recommendation. However, implementers are cautioned that this may not represent the latest information and are therefore strongly urged to consult the TSB patent database at <http://www.itu.int/ITU-T/ipr/>.

© ITU 2020

All rights reserved. No part of this publication may be reproduced, by any means whatsoever, without the prior written permission of ITU.

Table of Contents

		Page
1	Scope	1
2	References.....	1
3	Definitions	2
	3.1 Terms defined elsewhere	2
	3.2 Terms defined in this Recommendation.....	2
4	Abbreviations and acronyms	3
5	Conventions	4
6	Overview of the two-way DCAS terminal	5
7	Architecture of the two-way DCAS terminal	5
8	Two-way DCAS APIs	6
9	Terminal security chipset.....	6
	9.1 Terminal security chipset workflow	6
	9.2 The root key derivation module	8
	9.3 The key ladder module	9
	9.4 Security implementation mechanism (SIM).....	11
Annex A – Security mechanism of two-way DCAS client software downloading and bootloading		13
	A.1 Basic principles of chain of trust	13
	A.2 Bootup signature verification	13
	A.3 Downloading and replacing the DCAS client software	14
	A.4 Key management	15
	A.5 Security requirements of the bootloader.....	15
	A.6 Performance requirements of bootloader and terminal security chipset	16
Annex B – Two-way DCAS APIs		17
	B.1 Java APIs	17
	B.2 Javascript APIs	48
	B.3 Other GP extension APIs.....	68
	B.4 Security chipset key ladder driver APIs	72
Bibliography.....		76

Introduction

This Recommendation is part 3 of a multi-part deliverable covering the terminal for the two-way DCAS specification, as identified below:

Part 1 [ITU-T J.1031]: "Requirements";

Part 2 [ITU-T J.1032]: "System Architecture";

Part 3 (ITU-T J.1033): "The Terminal".

Recommendation ITU-T J.1033

Downloadable conditional access system for bidirectional networks – The terminal

1 Scope

The object of this Recommendation is to specify the terminal of the two-way DCAS for the bidirectional network, including the terminal security chipset, the two-way DCAS client software and the related application programming interfaces (APIs). This Recommendation is one in a series of Recommendations, specifying the whole two-way DCAS for the bidirectional network. The other parts of two-way DCAS Recommendations include the requirement for the two-way DCAS as defined in [ITU-T J.1031], and the specification of system architecture and related security mechanism for the two-way DCAS as defined in [ITU-T J.1032].

2 References

The following ITU-T Recommendations and other references contain provisions which, through reference in this text, constitute provisions of this Recommendation. At the time of publication, the editions indicated were valid. All Recommendations and other references are subject to revision; users of this Recommendation are therefore encouraged to investigate the possibility of applying the most recent edition of the Recommendations and other references listed below. A list of the currently valid ITU-T Recommendations is regularly published. The reference to a document within this Recommendation does not give it, as a stand-alone document, the status of a Recommendation.

- [ITU-T J.1031] Recommendation ITU-T J.1031 (2020), *Downloadable conditional access system for bidirectional networks – Requirements*.
- [ITU-T J.1032] Recommendation ITU-T J.1032 (2020), *Downloadable conditional access system for bidirectional networks – System architecture*.
- [ETSI ETR 289] ETSI ETR 289 (1996), *Digital Video Broadcasting (DVB); Support for use of scrambling and Conditional Access (CA) within digital broadcasting systems*.
- [NIST FIPS 197] NIST U.S. FIPS PUB 197 (FIPS 197) (2001), *Advanced Encryption Standard (AES)*.
- [NIST 3DES] National Institute of Standards and Technology (2017), *Recommendation for the Triple Data Encryption Algorithm (TDEA) Block Cipher (Revision 2)*, NIST Special Publication 800-67, November.
- [ISO/IEC 10118-3] ISO/IEC 10118-3:2018, *Information technology – Security techniques – Hash-functions – Part 3: Dedicated hash-functions*.
- [ISO/IEC 13818-1] ISO/IEC 13818-1:2019, *Information technology – Generic coding of moving pictures and associated audio information – Part 1: Systems*.
- [ISO/IEC 14888-3] ISO/IEC 14888-3:2018, *Information technology – Security techniques – Digital signatures with appendix – Part 3: Discrete logarithm based mechanisms*.

3 Definitions

3.1 Terms defined elsewhere

This Recommendation uses the following terms defined elsewhere:

3.1.1 descrambling [b-ITU-T J.93]: The processes of reversing the scrambling functions (see "scrambling") to yield usable pictures, sound and data services.

3.1.2 entitlement control messages (ECMs) [b-ITU-T J.290]: An ECM is an encrypted message that contains access criteria to various service tiers and a control word (CW).

3.1.3 entitlement management messages (EMMs) [b-ITU-T J.290]: The EMM contains the actual authorization data and shall be sent in a secure method to each CPE device.

3.1.4 scrambling [b-ITU-T J.93]: The process of using an encryption function to render television and data signals unusable to unauthorized parties.

3.1.5 bootloader [b-ITU-T J.1026]: The program for initiating hardware and loading software after a receiver boots up.

3.1.6 downloadable conditional access system (DCAS) [b-ITU-T J.1026]: A conditional access (CA) system that supports all the features of a legacy conditional access and provides CA-neutral mechanism to securely download CA client image and switch CA terminals without changing hardware through either a broadcasting or two-way network.

3.1.7 key ladder (KLAD) [b-ITU-T J.1026]: A structured multi-level key mechanism that ensures secure transport of control word.

3.1.8 root key [b-ITU-T J.1026]: The key used for the first level of a key ladder.

3.1.9 secure data management platform (SDMP) [b-ITU-T J.1026]: A platform that generates and manages some basic and root information, such as keys and IDs used in DCAS, including information to DCAS headend and to terminal security chipset.

3.1.10 security chipset key de-obfuscation [b-ITU-T J.1026]: Algorithm used to de-obfuscate encrypted security chipset key.

3.1.11 terminal security chipset [b-ITU-T J.1026]: A stream processing chipset with security functions such as secure key deriving and key ladder processing, etc.

3.1.12 terminal software platform [b-ITU-T J.1026]: A software platform running on a terminal, integrated with various hardware drivers, having various terminal application APIs, capable of downloading and running terminal applications according to specified security requirements, and providing a secure execution environment for terminal application.

3.1.13 transport stream filtering [b-ITU-T J.1026]: By using certain filtering mechanism to extract data matching filter rules from transport stream.

3.2 Terms defined in this Recommendation

This Recommendation defines the following terms:

3.2.1 two-way DCAS: A downloadable conditional access system (DCAS) operated especially in a two-way network.

3.2.2 two-way DCAS App: A two-way DCAS application running on the terminal software platform. After a terminal device is deployed in the field, this application can be upgraded or replaced through online downloading.

3.2.3 two-way DCAS trusted App: A two-way DCAS trusted application running in the trusted execution environment of a terminal device. After a terminal device is deployed in the field, this application can be upgraded or replaced through online downloading.

3.2.4 two-way DCAS manager: The terminal software platform's software component responsible for registering two-way DCAS client software, supporting information exchange between the two-way DCAS App and the two-way DCAS trusted App, as well as receiving and forwarding two-way DCAS entitlement control and management messages.

3.2.5 two-way DCAS client software: A terminal application composed of a two-way DCAS App and a two-way DCAS trusted App through the joint work with the support of the DCAS manager embedded in the terminal software platform.

3.2.6 two-way DCAS client software data: Data to be saved or updated, which include conditional access (CA) authorization information, CA private data, etc., when the two-way DCAS client software runs.

4 Abbreviations and acronyms

This Recommendation uses the following abbreviations and acronyms:

API	Application Programming Interface
BOSS	Business Operations Support System
CA	Conditional Access
CAT	Conditional Access Table
CAS	Conditional Access System
CBC	Cipher Blocker Chaining
CCI	Copy Control Information
ChipID	Chipset Identification
CPU	Central Processing Unit
CSA	Common Scrambling Algorithm
CW	Control Word
CRC	Cyclic Redundancy Check
DCAS	Downloadable Conditional Access System
DVB	Digital Video Broadcasting
ECB	Electronic Code Book
ECM	Entitlement Control Message
EMM	Entitlement Management Message
EPG	Electronic Programme Guide
ESCK	Encrypted Security Chipset Key
GP	Global Platform
IP	Internet Protocol
IPTV	TV using the Internet Protocol (IP)
IPC	Inter-Process Communication

JS	JavaScript
KLAD	Key Ladder
OSD	On-Screen Display
OTP	One Time Programmable
PID	Packet Identification
PIN	Personal Identification Number
PMT	Program Map Table
RAM	Random Access Memory
SCK	Security chipset Key
SCKv	Security chipset Key vendor
SDMP	Secure Data Management Platform
Seedv	Seed vendor
SIM	Security Implementation Mechanism
SMK	Secret Mask Key
SoC	System on Chip
TA	Trusted Application
TDES	Triple-DES
TEE	Trusted Execution Environment
TVOS	Smart TV Operating System
UDP	User Datagram Protocol
UI	User Interface
UUID	Universally Unique Identifier
Vendor_SysID	Vendor System Identification

5 Conventions

In this Recommendation:

The keywords **"is required to"** indicate a requirement which must be strictly followed and from which no deviation is permitted if conformance to this document is to be claimed.

The keywords **"is recommended"** indicate a requirement which is recommended but which is not absolutely required. Thus this requirement need not be present to claim conformance.

The keywords **"is prohibited from"** indicate a requirement which must be strictly followed and from which no deviation is permitted if conformance to this document is to be claimed.

The keywords **"can optionally"** indicate an optional requirement which is permissible, without implying any sense of being recommended. This term is not intended to imply that the vendor's implementation must provide the option and the feature can be optionally enabled by the network operator/service provider. Rather, it means the vendor may optionally provide the feature and still claim conformance with the specification.

In the body of this Recommendation and its annexes, the words *shall*, *shall not*, *should*, and *may* sometimes appear, in which case they are to be interpreted, respectively, as *is required to*, *is*

prohibited from, is recommended, and can optionally. The appearance of such phrases or keywords in an appendix or in material explicitly marked as *informative* are to be interpreted as having no normative intent.

6 **Overview of the two-way DCAS terminal**

The two-way DCAS terminal includes the terminal security chipset, the DCAS client software and the terminal software platform. The two-way DCAS terminal is an essential part of the two-way DCAS, whose architecture can be referred to Figure 1 in part 2 [ITU-T J.1032].

The two-way DCAS terminal validates the user's entitlement and descrambles protected services to implement conditional access of services. The terminal software platform can securely download, update and replace the two-way DCAS client software.

This Recommendation mainly focuses on specifying the terminal security chipset, two-way DCAS APIs embedded in the two-way DCAS terminal software platform, and two-way DCAS manager through the definition of the two-way DCAS APIs.

7 **Architecture of the two-way DCAS terminal**

The architecture of the two-way DCAS terminal is shown in the figure 1.

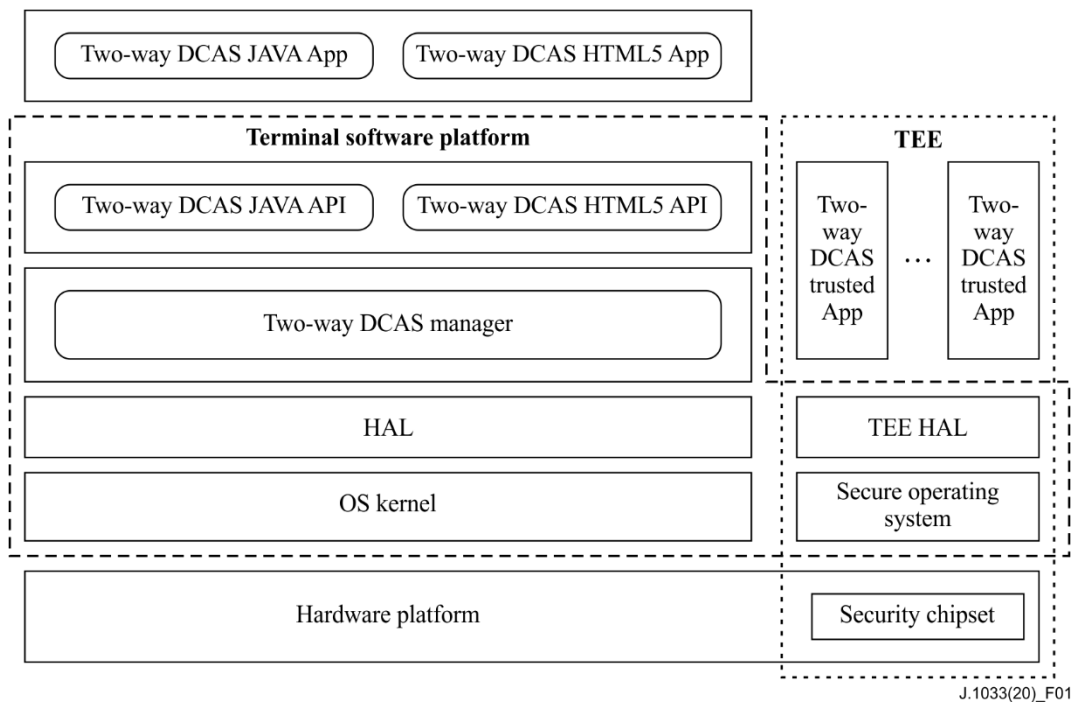


Figure 1 –Architecture of the two-way DCAS terminal

The terminal security chipset provides key ladder module and root key derivation module, to ensure the secure transport of the terminal data and the independence of the CA system. Clause 9 gives more details.

The two-way DCAS APIs embedded in the two-way terminal software platform support the joint work of two-way DCAS App and the two-way DCAS TApp with the assistance of the two-way DCAS manager embedded in the two-way terminal software platform.

The two-way DCAS manager uses its functions such as registration, cancellation and paring to manage the two-way DCAS App and the respective TApp.

The terminal software platform shall support trusted execution environment (TEE). It can either be a smart TV operating system (TVOS) or a middleware based on operating systems such as Linux and secureOS.

Two-way DCAS APIs supports DCAS manager to manage the two-way DCAS client software, and supports two-way DCAS client software to process the CA data such as ECM/EMM and data exchange with other applications such as electronic programme guide (EPG).

The two-way DCAS client software implements the functions of analysing and processing data such as ECM and EMM, and the processing of the secure information. The processing of the ECM and EMM shall be implemented within the TEE by DCAS TApp.

Functions of the two-way DCAS client software are achieved through the joint work of the two-way DCAS App and the respective two-way DCAS trusted App with support of the two-way DCAS manager and related APIs.

The two-way DCAS client software can be downloaded to terminal software platform, and runs in parallel with other applications on the same terminal software platform.

8 Two-way DCAS APIs

Two-way DCAS APIs are used to perform data exchange between two-way DCAS client software and terminal software platform.

There are two kinds of two-way DCAS APIs. One kind of API is the general API such as Java API and JavaScript API. The other kind of API is the two-way DCAS API for the two-way DCAS TApp. The detailed specification of the two-way DCAS APIs is given in Annex B.

The general APIs include:

- a) Filtering APIs. Two-way DCAS client software invokes the filtering APIs to receive ECM, EMM and CAT.
- b) Two-way DCAS management APIs. Two-way DCAS client software uses DCAS management APIs to register itself to the terminal software platform and receive descrambling requests from DCAS manager in the terminal software platform.

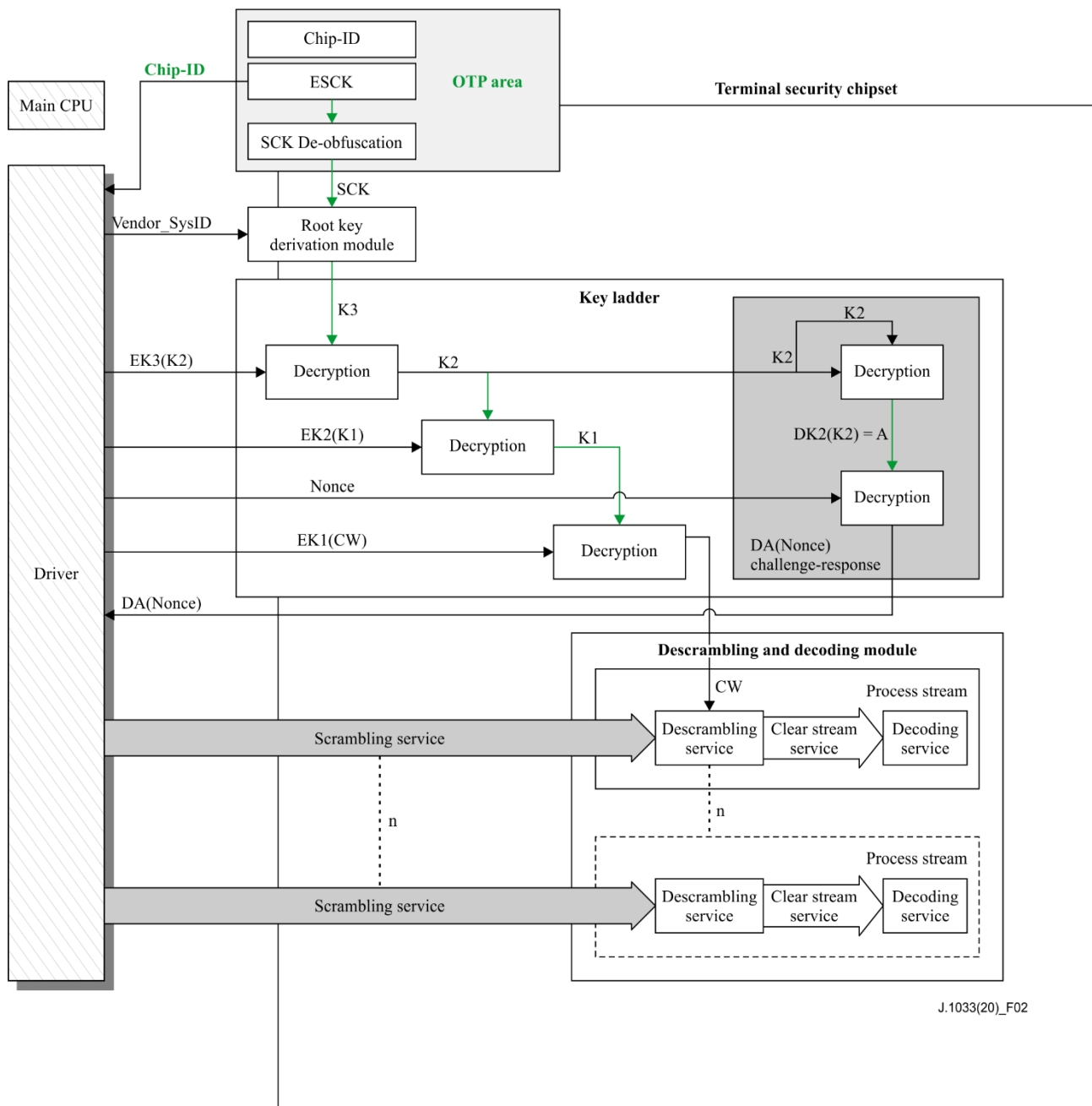
The two-way DCAS APIs for the two-way DCAS TApp include:

- a) Key ladder APIs. Two-way DCAS client software invokes the key ladder APIs to load encrypted keys to the terminal security chipset to descramble transport streams.
- b) Other global platform (GP) extension APIs. Two-way DCAS client software uses this APIs for encryption/decryption, signature verification, memory management, debug printing, etc.

9 Terminal security chipset

9.1 Terminal security chipset workflow

Figure 2 shows the functional diagram of the terminal security chipset also defined in [b-GY/T 255-2012], which includes modules of OTP, root key derivation, key ladder, descrambling and decoding.



J.1033(20)_F02

Figure 2 – Functional diagram of the terminal security chipset

The terminal security chipset uses root key derivation module to generate root key K3, and uses key ladder module to ensure the secure transport of the control word and other secret keys as well as the validity of the terminal security chipset.

The terminal security chipset's functionalities shall not be implemented by the primary central processing unit (CPU).

The workflow of the terminal security chipset is described as follows:

After a terminal security chipset is powered on, the OTP module of the terminal security chipset uses built-in encrypted security chipset key (ESCK) and security chipset key (SCK) de-obfuscation to generate SCK, which is fed to the root key derivation module to generate root key K3.

The key ladder module receives the root key K_3 and uses it for decryption of keys and challenge-response. The key ladder module contains functions such as: decrypting keys level by level with the input encrypted keys; processing challenge information (Nonce) and generating response (DA (Nonce)).

Finally, the decrypted control word (CW) is sent to the descrambling and decoding module for descrambling and decoding services. The challenge-response is a function in the two-way system for the headend to certificate the system on chip (SoC) chip.

9.2 The root key derivation module

The root key derivation module consists of a group of hardware logical modules. It uses an embedded derivation mechanism together with input parameters for the derivation of the root key. All CA vendors use this derivation mechanism to generate different root keys. This method can circumvent the vulnerability of using a singular root key.

The functions of the root key derivation module include SCK preliminary manipulation function, vendor separation function and final root key derivation function. The root key derivation module can derive a specific root key for a two-way DCAS vendor according to the input SCK and Vendor_SysID. Figure 3 shows the functional diagram of the root derivation module also defined in [b-GY/T 255-2012] and [b-ETSI TS 103 162].

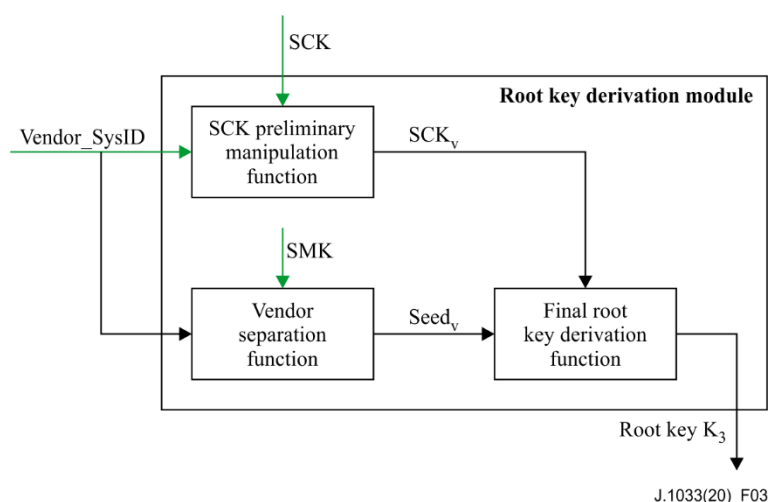


Figure 3 – The functional diagram of the root derivation module

The preliminary SCK manipulation function generates security chipset key vendor (SCK_v) based on SCK and input Vendor_SysID. The function must use a cipher algorithm with certain level of security strength which ensures the SCK cannot be retrieved when Vendor_SysID and SCK_v are known. Additionally, the function shall be provided by the terminal security chipset vendor and certified by SDMP.

The vendor separation function generates $Seed_v$ by using Vendor_SysID and secret mask key (SMK) as input. The function must use a cipher algorithm with certain level of security strength which ensures the SMK cannot be retrieved when Vendor_SysID and $Seed_v$ are known. Additionally, the function shall be provided by the terminal security chipset vendor and certified by SDMP.

The final root key derivation function derives root key K_3 based on the input of SCK_v and $Seed_v$. In the execution process, this function shall use a two-way function to ensure the other input parameter cannot be retrieved when K_3 and any other input parameter are known. For example, $Seed_v$

cannot be retrieved when K3 and SCKv are known. This function shall be provided by the terminal security chipset vendor and certified by the SDMP.

- Vendor_SysID: 2 bytes, used for identifying CA system and assigned by SDMP.
- SCKv: 16 bytes
- SMK: 16 bytes, gate-level data provided by chipset vendor
- Seedv: 16 bytes
- Terminal security chipset can support various final root key derivation functions at the same time.

9.3 The key ladder module

9.3.1 3-level key mechanism

Figure 4 shows the functional diagram of the key ladder module also defined in [b-GY/T 255-2012].

The terminal security chipset specified by this Recommendation shall support the 3-level key ladder mechanism. The security requirements and technical details for the key ladder mechanism with more levels are out of the scope of this Recommendation.

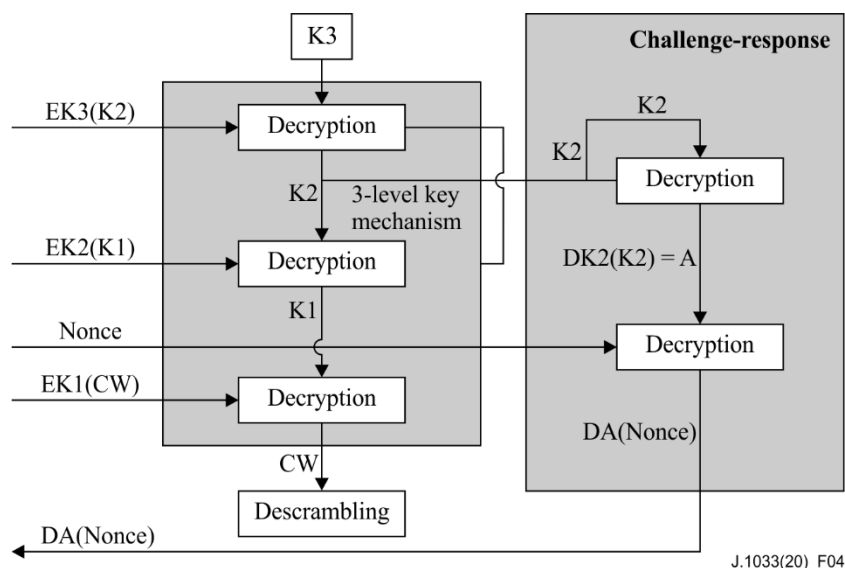


Figure 4 – The functional diagram of the key ladder module

The 3-level key ladder mechanism ensures secure transport of CW within the terminal.

The 3-level key ladder mechanism uses root key K3 obtained from root key derivation module, to decrypt EK3 (K2), EK2 (K1), EK1 (CW) to obtain the CW required by descrambling; at the same time, K2 works with challenge information (Nonce) to generate response(DA (Nonce)).

The terminal security chipset shall decrypt scrambled services with the following procedure:

- a) Shall receive encrypted EK3(K2), use K3 to decrypt it and generate K2;
- b) Shall receive EK2(K1), use K2 to decrypt it and generate K1;
- c) Shall receive EK1(CW), use K1 to decrypt it and generate CW;
- d) CW is used for decrypt scrambled services.

EK3(K2) represents key K2 encrypted with key K3.

EK2(K1) represents key K1 encrypted with key K2.

EK1(CW) represents CW encrypted with key K1.

K3 is derived root key, 16 bytes.

K2 is the key used to decrypt K1, 16 bytes.

K1 is the key used to decrypt CW, 16 bytes.

CW is key used to descramble services, 8 or 16 bytes

The key ladder mechanism may support SM4 algorithm defined in [b-GB/T 32907] with 128-bits keys and data blocks in electronic code book (ECB) mode.

9.3.2 Challenge-response mechanism

The terminal security chipset shall support challenge-response mechanism, which can be used in some security functions.

The challenge-response mechanism shall comply with the following procedure:

- a) The terminal security chipset shall receive Vendor_SysID, EK3(K2) and Nonce by using the driver API.
- b) The terminal security chipset shall use derived K3 to decrypt EK3(K2) to get K2.
- c) The terminal security chipset shall decrypt K2 using K2 itself to generate DK2(K2), denoted as A.
- d) The terminal security chipset shall decrypt Nonce using A, to generate DA(Nonce).
- e) The terminal security chipset returns DA(Nonce) by using the driver API.

DK2(K2) represents the process of decrypting K2 using K2.

A denotes the result of DK2(K2), 16 bytes.

Nonce denotes challenge data, 16 bytes.

DA(Nonce) denotes the result from decrypting Nonce with A as the key.

9.3.3 The OTP area

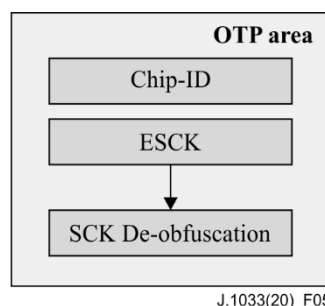


Figure 5 – OTP area functional diagram

Figure 5 shows the functional diagram of the OTP area also defined in [b-GY/T 255-2012], which is used to store information such as chipset identification (ChipID), ESCK and SCK de-obfuscation. The logical circuit of SCK de-obfuscation reads ESCK from the OTP area, de-obfuscates ESCK to SCK and provides SCK to root key derivation module.

SCK is the security information required for deriving root key. SCK should not be stored in the OTP area as plain text.

- a) SCK is terminal security chipset key, which is unique per chipset and is generated by SDMP, 16 bytes.
- b) ESCK is the encrypted SCK provided by SDMP. It is stored in OTP area with the same length as SCK.
- c) ChipID is an 8-byte public identifier of terminal security chipset. It includes information such as chipset vendor, type, as well as a 4-byte unique global identifier assigned by secure data management platform. The ChipID format is described in Table 1.

Table 1 – ChipID data format

Data name	Length (Bit)	Data type
Chip Vendor ID	8	Uimbsf
Chip Type	12	Uimbsf
Reserved	12	Uimbsf
Chipset SN	32	Uimbsf

Chip Vendor ID: unique ID for chipset vendor, 8 bits.

Chip Type: ID for a chipset model manufactured by a chipset vendor. Assigned by SDMP, 12 bits.

Reserved: 12 bits.

Chipset SN, a 32-bit serial number of a chipset manufactured by a chipset vendor. It is globally unique for any chipset regardless of whether the chipset vendor or type is the same.

9.4 Security implementation mechanism (SIM)

9.4.1 SIM of the terminal security chipset

9.4.1.1 OTP area

The OTP area of terminal security chipset is used to store information such as ChipID, ESCK and BL_KEY0, etc. It shall comply with the following requirements:

- a) The content written in OTP area cannot be changed;
- b) SCK is necessary security information required for root key derivation, clear SCK should not be directly stored in the OTP area;
- c) The security information in the OTP area shall be able to be tested and verified by the terminal security chipset to see if it has been tampered. If the security information is tampered, two-way DCAS function modules shall stop working immediately.

9.4.1.2 The SCK De-obfuscation function

The SCK de-obfuscation function of terminal security chipset shall comply with the following requirements:

- a) It shall be a cryptographically secure enough two-way function.
- b) It shall be implemented by internal hardware logical circuits; external software cannot intercept SCK or ESCK.
- c) The SCK de-obfuscation function shall only be used by the root key derivation module.

9.4.1.3 Root key derivation module

The root key derivation module of terminal security chipset consists of sub modules such as SCK preliminary processing function, vendor separation function and final root key derivation function. The root key derivation module shall comply with the following requirements:

- a) The root key derivation module shall be implemented by using hardware logical circuits or independent secure operation unit, working independently with dedicated calculation and storage resources. Any other units cannot interfere with the logic, execution, and result of the derivation module;
- b) Any intermediate result from running of root key derivation module shall not be output or read to external modules.
- c) SCK preliminary processing function shall use a two-way function with certain level of security strength, such as at least AES128 or SM4, to ensure the SCK cannot be de-obfuscated in case Vendor_SysID and SCKv are known. AES is defined in [NIST FIPS 197].
- d) The vendor separation function shall use a two-way function with certain level of security strength to ensure SMK cannot be de-obfuscated in case Vendor_SysID and Seedv are known.
- e) The final root key derivation function shall use a two-way function with a certain level of security strength so that when root key K3 and any input parameter are known, another input parameter cannot be retrieved. For example, Seedv cannot be retrieved when K3 and SCKv are known.
- f) SMK shall not be read or tampered by external modules, and its length shall be at least 128 bits;
- g) Terminal security chipset can support a variety of final root key derivation functions at the same time.

9.4.1.4 Key Ladder module

The key ladder module of the terminal security chipset consists of multi-level key mechanism and challenge-response mechanism. The key ladder module shall comply with the following requirements:

- a) It must be implemented using hardware logical circuits or independent secure operation unit, and it must work independently with dedicated calculation and storage resources.
- b) The logic, execution and results of the key ladder module cannot be interfered with or used by any other modules except the descrambling module can get CW from key ladder module.
- c) Any intermediate result from running the key ladder shall not be output to or read out by any external module.
- d) Any key in key ladder cannot be exposed in the challenge-response process.

Annex A

Security mechanism of two-way DCAS client software downloading and bootloading

(This annex forms an integral part of this Recommendation.)

A.1 Basic principles of chain of trust

The security mechanism of DCAS client software bootloading is based on a bottom-to-top chain of trust.

This chain of trust is established by using the digital signature technique. From bottom to top, it includes: terminal security chipset, bootloader, terminal software platform and DCAS client software.

The security mechanism of DCAS client software bootloading requires every link in the chain of trust must perform signature verification in a bottom-to-top order. Only if the signature verification at current link passes, can the security verification of next link be started. Only if the signature verification at all links passes, can DCAS client software be launched.

In the entire chain of trust:

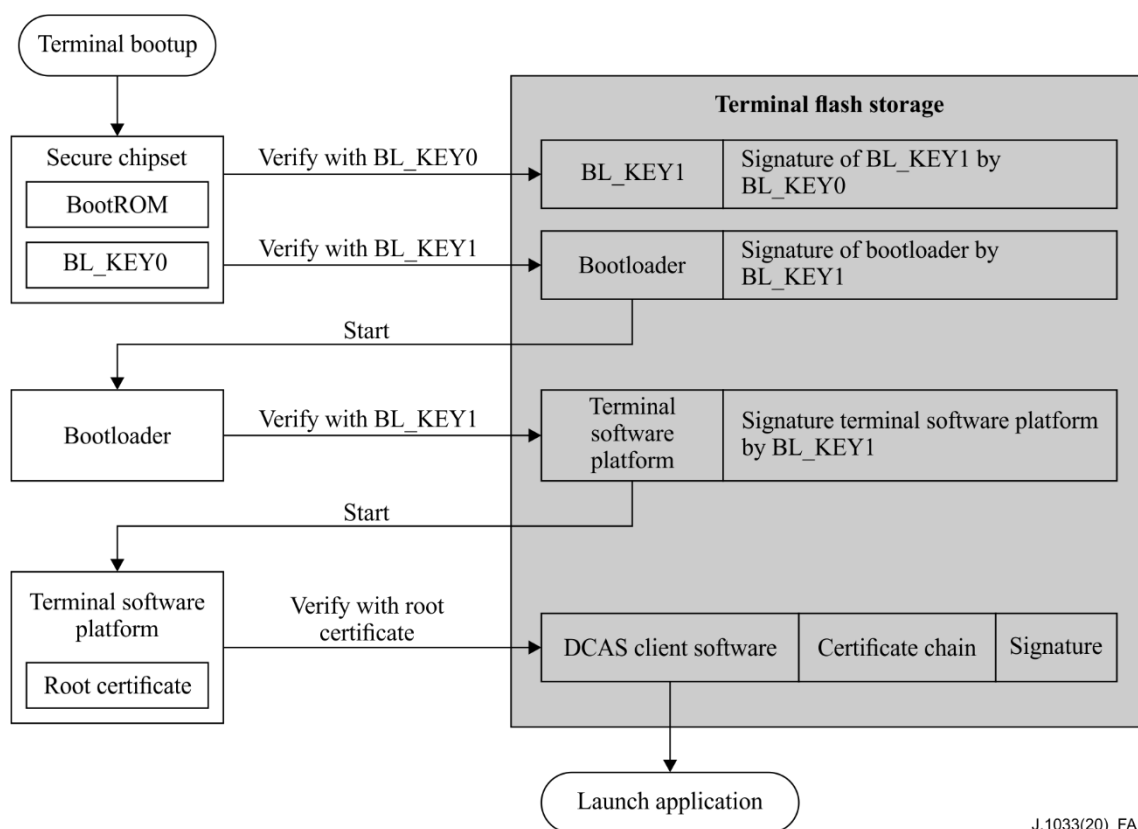
- a) The OTP area of terminal security chipset shall be preset with a verification key for bootloader.
- b) Every link in the chain of trust shall be preset with a verification key for verifying software of the next link.
- c) Software of every link of the chain of trust shall be digitally signed by a private key corresponding to the verification key.
- d) Software of every link shall have its digital signature;
- e) Software of the current link shall complete the security verification of the software of the next link first, before the software of the next link can be launched.

The downloading, loading and running of DCAS client software shall comply with the mechanism of chain of trust described above at any time.

A.2 Bootup signature verification

The CPU shall execute certain secure codes to check the bootloader in the Flash that is already signed. Bootloader needs to verify the signature of the terminal software platform, and terminal software platform needs to verify signed DCAS client software.

See Figure A.1 for the verification process of the bootloader.



J.1033(20)_FA.1

Figure A.1 – Verification of Bootloader

A public key assigned by SDMP, denoted as BL_KEY0, shall be embedded in the secure area of the terminal security chipset during SoC production. SDMP shall under no circumstances leak BL_KEY0's private key. The BL_KEY0 in Figure A.1 is the public key.

During bootup, the code of the BootRom in the secure area of terminal security chipset executes to read an additional public key (denoted by BL_KEY1) and signature of BL_KEY1 from terminal flash storage, and use BL_KEY0 to verify BL_KEY1's signature. The BL_KEY1 in Figure A.1 is the public key.

After BL_KEY1 is successfully verified, BootRom shall verify bootloader using BL_KEY1.

Bootloader shall use BL_KEY1 to verify the signature of terminal software platform. The terminal software platform shall use its embedded root certificate to verify the DCAS client software.

The algorithms used by all the signatures mentioned above include SM3 hash and SM2 public key cryptographic algorithm. SM3 algorithm is defined in [ISO/IEC 10118-3] and SM2 algorithm is defined in [ISO/IEC 14888-3].

A.3 Downloading and replacing the DCAS client software

The downloading and replacing of the DCAS client software, including DCAS App and DCAS trusted App, are implemented by the loader function of bootloader or the application manager in terminal software platform. The bootloader is used for the downloading and replacing of the whole terminal software image including terminal software platform and applications software, and the application manager is only used for the download and replacement of the applications software. The download and replacement procedure is as follows:

- a) The headend system sends information of starting DCAS download and replacement to the terminal;
- b) The bootloader or application manager downloads the image or DCAS client software;
- c) The bootloader or application manager performs signature verification over the downloaded image or DCAS client software;
- e) The bootloader or application manager replaces the image or DCAS client software with the verified new one;
- e) The terminal reboots;
- f) The new DCAS client software shall be able to work with SoC. Thus, the download and replacement of the DCAS client software are complete.

A.4 Key management

See Table A.1 for a description of keys for bootloader.

Table A.1 – Description of keys for bootloader

Key name	Key owner	Signed by	Used to sign	Note
BL_KEY0	SDMP	N/A	BL_KEY1	Public key is embedded to chipset
BL_KEY1	Operator	BL_KEY0	Bootloader Terminal software platform.	

For BL_KEY0:

SDMP shall manage its internal database for BL_KEY0, and shall be responsible for sending chipset vendors BL_KEY0, which will be embedded to designated chipset.

For BL_KEY1:

BL_KEY1 is owned by the operator. The key could be managed by either the operator or a third-party trusted authority, who signs the terminal software.

SDMP shall provide the method and procedure for signing BL_KEY1 with BL_KEY0. The owner of BL_KEY1 shall provide detailed information to SDMP, including chipset model and public key of BL_KEY1. SDMP shall sign BL_KEY1's public key with BL_KEY0's private key, and return the result together with BL_KEY1's public key to BL_KEY1's owner, so that BL_KEY1's public key and signed result can be preset into the terminal device.

A.5 Security requirements of the bootloader

The security requirements of the bootloader focus on the bootup and download process. Bootloader in flash should be copied to random access memory (RAM) before boot, and boot from RAM (BFR). The bootloader shall comply with the chain of trust mechanism to verify software signature during boot and loader.

For the bootup process:

The bootloader shall not launch subsequent component software until their signatures are verified. The bootloader shall verify the signature again every time the subsequent software restarts. Only the bootloader stored in the terminal's flash can be executed. The signature verifications and executions of all software shall be performed in RAM.

For the download:

The bootloader shall write downloaded software and its signature to flash only after the signature of the downloaded software is verified in RAM. After the software is successfully stored, the terminal shall perform a complete reboot. If the downloaded software exceeds the maximum size of the flash allocated by bootloader, the bootloader shall reject the software and reboot. Downgrades of upgradable software should be avoided.

A.6 Performance requirements of bootloader and terminal security chipset

To ensure terminal user experience, a terminal security chipset shall be used to provide hardware acceleration for hash algorithm.

Annex B

Two-way DCAS APIs

(This annex forms an integral part of this Recommendation.)

B.1 Java APIs

The standard Java virtual machine solution has been widely used in the industry for downloading and executing applications. Figure B.1 illustrates the runtime environment of the DCAS client software.

DCAS client software is an Xlet application running on terminal software platform that supports Java runtime environment.

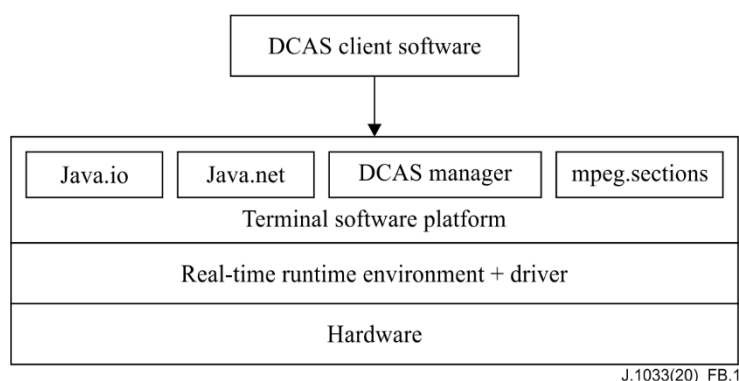


Figure B.1 – Diagram of DCAS JAVA APIs

B.1.1 APIs type

B.1.1.1 APIs for DCAS Manager

B.1.1.1.1 A DCAS module manager (DCAS Manager) has been defined for the terminal software platform to manage requests for descrambling services. DCAS Manager includes an upper layer API and bottom layer API of terminal software platform, and extension application APIs. The upper layer APIs of terminal software platform.

A CA module manager is defined by the upper-level APIs of terminal software platform, to manage requests for descrambling services (which means to descramble video/audio streams). A DCAS client software must register the CA module in CA module manager in order to receive descramble request from the terminal software platform on a terminal device.

The DCAS client software requires terminal software platform to implement the upper-layer APIs of the DCAS terminal software platform.

B.1.1.1.2 The bottom-layer APIs of the terminal software platform

Except for the existing Java APIs, DCAS client software requires a collection of Java APIs that are implemented on the terminal software platform, including the extension APIs required for accessing the terminal security chipset

B.1.1.1.3 Extension application APIs

By extending DCAS APIs, the conditional access system (CAS) management module on a terminal software platform can perform basic CA information communication with Java Apps without being limited by using IXC between Java application and DCAS application.

DCAS client software requires DCAS extension APIs to be implemented by terminal software platform for DCAS applications.

B.1.1.1.4 Detachable security device APIs

DCAS applications can communicate with detachable security devices via this APIs.

B.1.1.2 Network APIs

DCAS client software can use Java network APIs to access network resources, such as interconnection with headend CA server.

DCAS client software requires terminal software platform to implement the existing Java network API according to the definition in Java.net.

B.1.1.3 MPEG section filter APIs

DCAS client software uses MPEG Section Filter API to load MPEG Section for CA. CA related data includes ECM, EMM and conditional access table (CAT).

DCAS client software requires terminal software platform to implement MPEG Section Filter APIs according to definitions in `org.davic.mpeg.sections`, `org.davic.mpeg.TransportStream` and `org.davic.net.tuning.NetworkInterface`.

B.1.1.4 Non-volatile storage APIs

DCAS client software can use the existing Java APIs to access terminal storage, including storing data in non-volatile storage.

DCAS client software requires terminal software platform to implement non-volatile storage APIs.

DCAS client software can store data by using designated directory in the file system of a non-volatile storage. Terminal software platform needs to provide proper functions to read the name of root path of the file system.

B.1.2 APIs invoking sequence

This clause describes two scenarios where DCAS API is used: CASModule registration and channel switching. See Figure B.2 for CASModule registration and Figure B.3 for channel switching.

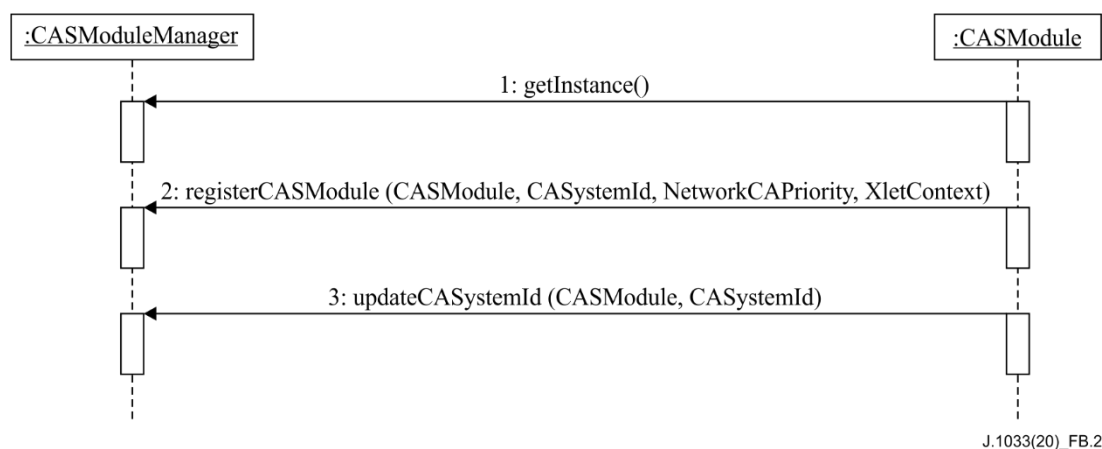


Figure B.2 – CA Module Registration in CASModuleManager

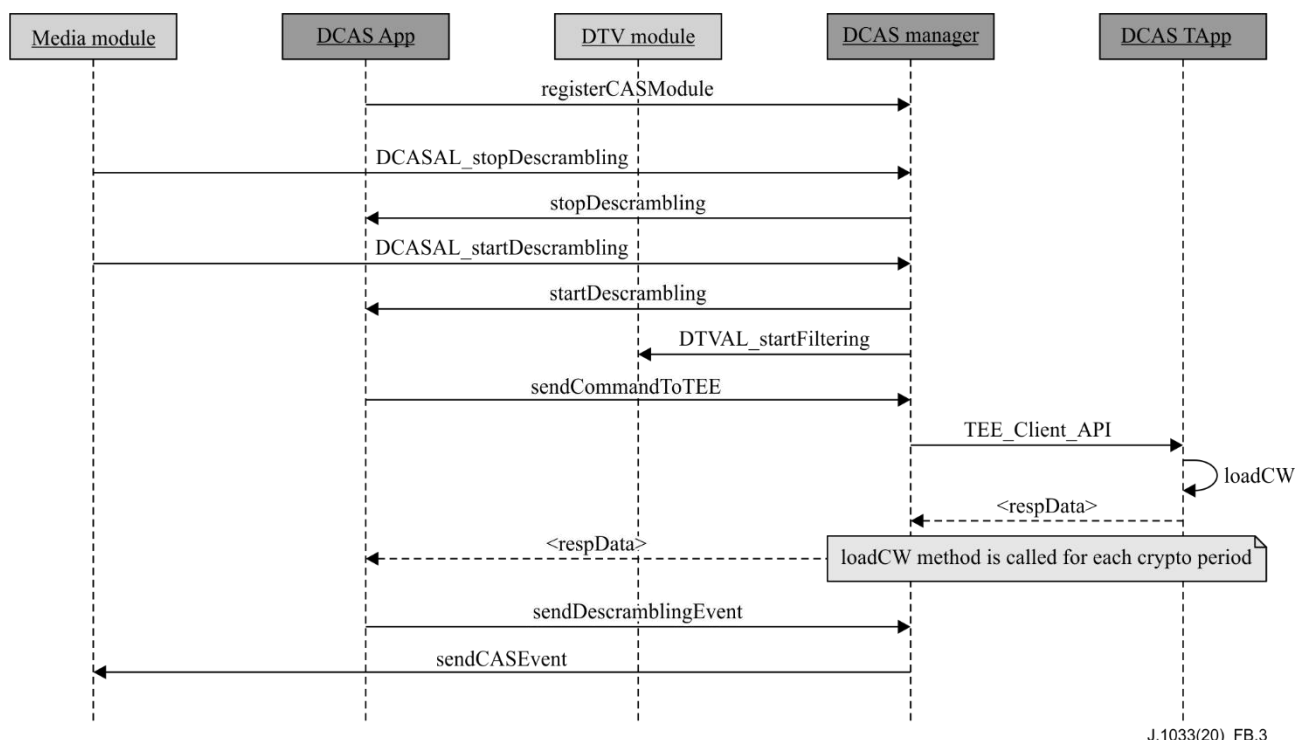


Figure B.3 – Channel Selection

B.1.3 APIs description

See Table B.1 for the names of the APIs.

Table B.1 – APIs

API name	Package name
Terminal software platform upper layer API	org.ngb.net.cas.module
Terminal software platform bottom layer API	org.ngb.net.cas.controller
Extension Application API	org.ngb.net.cas.event
Detachable Security Device API	org.ngb.net.cas.detachable
Network API	java.net
Section Filter API	org.davic.mpeg.sections
Non-volatile Storage API	java.io

B.1.4 Package org.ngb.net.cas.module

Package org.ngb.net.cas.module provides the upper layer APIs of DCAS terminal software platform, which needs to be implemented in TVOS.

Refer to Table B.2 for the overview of Package org.ngb.net.cas.module

Table B.2 – Overview of package org.ngb.net.cas.module

Interface	
CASModule	To denote the CASModule object that requests descrambling of a group of elementary streams.
CASDataUtils	To obtain and set CA information, as well as read and write DCAS data.
CADescriptor	Provide information of CA descriptor. The program map table (PMT) of a given service may provide CA descriptors. CAT may also have CA descriptors.
CASServiceComponentInfo	For extracting component information of a CA service, for example, ECM packet identification (PID) and DescramblerContext for loading CW.
CASPacketListener	DCAS application uses this interface to receive out of band CAS Packets (e.g., EMMs).
CASSession	Provide information about a CAS session.
CASStatus	DCAS application sends CASStatus every time the descrambling status in DescramblerContext changes. This status is used to indicate if descrambling is successful. If any component fails to be descrambled, this status must report the failure of the request on descrambling the entire service. When receiving a new CASStatus, terminal software platform shall use the CAS event described in this clause to notify other applications.
CATListener	Required by DCAS application to filter in-band EMM using CA descriptor in CAT.
CATNotifier	Used by DCAS application to register the listener for receiving CAT update notification.
Class	
CASModuleManager	Used to register all CASModules implemented by all DCAS applications.
CASPermission	Any DCAS application must get CASPermission to access CASModuleManager. This mechanism ensures only network operator authorised DCAS can use DCAS APIs.

B.1.4.1 Interface org.ngb.net.cas.module.CASModule

B.1.4.1.1 Methods

B.1.4.1.1.1 startDescrambling

Prototype:

```
public void startDescrambling (CASSession,
                               CASServiceComponentInfo[]casci )
```

Description:

This method is invoked by terminal software platform to request CASModule to descramble a group of elementary streams in a given session.

DCAS application gets the relevant Network Interface object from CAS session, and gets the TransportStream object from the NetworkInterface object, which will be used for ECM section filtering via org.davic.mpeg.sectionsAPI.

Parameters:

casSession: Session for the descrambling request.