

Digital Video Broadcasting (DVB) CBMS Mobile Device Security Framework Specification

European Broadcasting Union



Union Européenne de Radio-Télévision

THIS IS A PROVISIONAL DVB DOCUMENT. IT MAY BE CHANGED BEFORE FINAL ADOPTION BY DVB. THIS PROVISIONAL DOCUMENT IS FOR DISCUSSION PURPOSES ONLY. IMPLEMENTERS ARE NOT ENTITLED TO RELY ON THIS PROVISIONAL DOCUMENT. WHERE POSSIBLE, ITEMS FOR WHICH CONSENSUS HAS NOT BEEN REACHED ARE SUITABLY MARKED, FOR EXAMPLE BY SQUARE BRACKETS. IMPLEMENTERS SHOULD ALSO NOTE THAT ONLY FINAL SPECIFICATIONS ADOPTED BY DVB ARE (SUBJECT TO THE "NEGATIVE DISCLOSURE" RIGHTS OF MEMBERS) ENTITLED TO THE IPR LICENSING TERMS OF DVB'S MEMORANDUM OF UNDERSTANDING.



Reference

REN/JTC-DVB-102

Keywords

broadcasting, digital, DVB, MPEG, service, TV,
video**ETSI**

650 Route des Lucioles
F-06921 Sophia Antipolis Cedex - FRANCE

Tel.: +33 4 92 94 42 00 Fax: +33 4 93 65 47 16

Siret N° 348 623 562 00017 - NAF 742 C
Association à but non lucratif enregistrée à la
Sous-Préfecture de Grasse (06) N° 7803/88

Important notice

Individual copies of the present document can be downloaded from:

<http://www.etsi.org>

The present document may be made available in more than one electronic version or in print. In any case of existing or perceived difference in contents between such versions, the reference version is the Portable Document Format (PDF).

In case of dispute, the reference shall be the printing on ETSI printers of the PDF version kept on a specific network drive within ETSI Secretariat.

Users of the present document should be aware that the document may be subject to revision or change of status.

Information on the current status of this and other ETSI documents is available at <http://www.etsi.org/tb/status/>

If you find errors in the present document, send your comment to:
editor@etsi.fr

Copyright Notification

No part may be reproduced except as authorized by written permission.
The copyright and the foregoing restriction extend to reproduction in all media.

© European Telecommunications Standards Institute 2000.
© European Broadcasting Union 2000.
All rights reserved.

Contents

Contents	3
Intellectual Property Rights.....	4
Foreword.....	4
1 Scope.....	5
2 References.....	5
3 Definitions and abbreviations.....	5
3.1 Definitions	5
3.2 Abbreviations.....	6
4 Introduction.....	7
5 Mobile Device Security Architecture	7
5.1 Content and Service Protection Architecture.....	7
5.2 Viewer Device Segment	8
5.2.1 Key Management System Device Agent (KDA).....	8
5.2.2 Descrambler.....	8
5.2.3 Session Setup.....	8
5.2.4 UICC.....	8
6 Key Management System Device Agent	9
6.1 Overview	9
6.1.1 J2ME Mobile Information Device Profile (MIDP).....	9
6.2 KDA Platform.....	9
6.2.1 Interaction Channel.....	11
6.3 Security Background	11
6.4 KDA APIs.....	11
6.4.1 UICC.....	11
6.4.2 Generic Connection Framework	11
6.4.3 Descrambler.....	12
6.4.3.1 Descrambler interface	12
6.4.3.2 DescramblerContext interface	13
7 The KDA Life Cycle.....	13
7.1 Loading a new KDA.....	14
8 UICC	14
8.1 Application IDentifier (AID)	14
8.2 Application and Logical Structure of the Card.....	14
8.3 EF _{KMSID} (KMS Identifier).....	16
8.4 KMS application selection	16
9 Secure Authenticated Channel Protocol.....	16
9.1 High level description of the SAC.....	16
9.2 The cryptographic keys and parameters.....	17
9.2.1 The Descrambler's keys.....	17
9.3 The SAC protocol.....	17
9.3.1 Session key establishment.....	17
9.3.2 Secure key exchange.....	17
10 Bibliography.....	18
11 History.....	18

Intellectual Property Rights

IPRs essential or potentially essential to the present document may have been declared to ETSI. The information pertaining to these essential IPRs, if any, is publicly available for **ETSI members and non-members**, and can be found in ETSI SR 000 314: *"Intellectual Property Rights (IPRs); Essential, or potentially Essential, IPRs notified to ETSI in respect of ETSI standards"*, which is available from the ETSI Secretariat. Latest updates are available on the ETSI Web server (<http://www.etsi.org/ipr>).

Pursuant to the ETSI IPR Policy, no investigation, including IPR searches, has been carried out by ETSI. No guarantee can be given as to the existence of other IPRs not referenced in ETSI SR 000 314 (or the updates on the ETSI Web server) which are, or may be, or may become, essential to the present document.

Foreword

This European Standard (Telecommunications series) has been produced by the Joint Technical Committee (JTC) Broadcast of the European Broadcasting Union (EBU), Comité Européen de Normalisation ELECtrotechnique (CENELEC) and the European Telecommunications Standards Institute (ETSI).

NOTE: The EBU/ETSI JTC Broadcast was established in 1990 to co-ordinate the drafting of standards in the specific field of broadcasting and related fields. Since 1995 the JTC Broadcast became a tripartite body by including in the Memorandum of Understanding also CENELEC, which is responsible for the standardization of radio and television receivers. The EBU is a professional association of broadcasting organizations whose work includes the co-ordination of its members' activities in the technical, legal, programme-making and programme-exchange domains. The EBU has active members in about 60 countries in the European broadcasting area; its headquarters is in Geneva.

European Broadcasting Union
CH-1218 GRAND SACONNEX (Geneva)
Switzerland
Tel: +41 22 717 21 11
Fax: +41 22 717 24 81

Founded in September 1993, the DVB Project is a market-led consortium of public and private sector organizations in the television industry. Its aim is to establish the framework for the introduction of MPEG-2 based digital television services. Now comprising over 200 organizations from more than 25 countries around the world, DVB fosters market-led systems, which meet the real needs, and economic circumstances, of the consumer electronics and the broadcast industry.

National transposition dates
Date of adoption of this EN:
Date of latest announcement of this EN (doa):
Date of latest publication of new National Standard or endorsement of this EN (dop/e):
Date of withdrawal of any conflicting National Standard (dow):

1 Scope

This document specifies the mobile device security framework required for decrypting content broadcasted with the DVB-CBMS system.

This specification should be read in conjunction with:

- ETSI EN xxx xx1 DVB CBMS Service Purchase and Protection Open Framework, System Architecture Specification.
- ETSI EN xxx xx2 DVB CBMS Service Purchase and Protection Open Framework, Content Encryption Specification.

2 References

- [1] ISO/IEC 13818-1 - "Information technology - Generic coding of moving pictures and associated audio information - part 1: Systems"- 200-12-01
- [2] IETF RFC 2396 – "Uniform Resource Identifier"
- [3] ETSI EN xxx xx1 DVB CBMS Service Purchase and Protection Open Framework, System Architecture Specification.
- [4] ETSI EN xxx xx2 DVB CBMS Service Purchase and Protection Open Framework, Content Encryption Specification.
- [5] JSR 118: Mobile Information Device Profile 2.0 <http://jcp.org/en/jsr/detail?id=118>
- [6] JSR 177: Security and Trust Services API for J2ME™ <http://jcp.org/en/jsr/detail?id=177>
- [7] ETR 162 Digital broadcasting systems for television, sound and data services; Allocation of Service Information (SI) codes for Digital Video Broadcasting (DVB) systems
- [8] ETSI TS 102 221: Smart-cards; UICC-Terminal interface; Physical and logical characteristics
- [9] ETSI TS 101 220: Smart-cards; ETSI numbering system for telecommunication application providers
- [10] ETSI TS 31 101: 3GPP; Technical Specification Group Terminals; UICC-terminal interface; Physical and logical characteristics (release6)
- [11] ETSI TS 31 102: 3GPP; Technical Specification Group Terminals; Characteristics of the USIM application
- [12] ISO 7816-4 Smart Card Standard: Part 4: Interindustry Commands for Interchange
- [13] Handbook of applied cryptography by Alfred J. Menezes, Paul C. van Oorschot and Scott A. Vanstone.

3 Definitions and abbreviations

3.1 Definitions

For the purposes of the present document, the following terms and definitions apply:

Mobile TV Application – the main device application responsible for accessing the Mobile TV Services. Sometimes referred as the Player Application.

3.2 Abbreviations

For the purposes of the present document, the following abbreviations apply:

ADF	Application Dedicated File
AID	Application Identifier
APDU	Application Protocol Data Units
API	Application Program Interface
ASCII	American Standard Code for Information Interchange
BCD	Binary Coded Decimal
Bslif	Bit string left bit first
CA	Conditional Access
CAS	Conditional Access System
CAS_ID	CA System Identifier
Device	Mobile Device or Terminal
DVB	Digital Video Broadcasting
DVB-H	DVB-Handheld
DVB-T	DVB-Terrestrial
ESG	Electronic Service Guide
GCF	Generic Connection Framework
IP	Internet Protocol
IPDC	IP DataCast
ISMA	Internet Streaming Media Alliance
ISO	International Standards Organization
J2ME	Java 2 Micro Edition
JVM	Java Virtual Machine
KDA	Key Management System Device Agent
KMS	Key Management System
MF	Master File
MHP	Multimedia Home Platform
MPE	Multi-Protocol Encapsulation
MPEG	Moving Pictures Experts Group
PIX	Proprietary application Identifier eXtension
PSS	Probabilistic Signature Scheme
RID	Registered application provider Identifier
RTP	Real-Time Transport Protocol
SC	Smart Card
SDP	Session Description Protocol
SHA	Standard Hash Algorithm
SHW	Secure Hardware
SIM	Subscriber Identity Module
Terminal	A consumer device that can receive, descramble, and decode mobile DVB-H services.
UI	User Interface
UICC	Universal Integrated Circuit Card
Uimbsf	Unsigned integer, most significant bit first
UMTS	Universal Mobile Telecommunications System
URL	Uniform Resource Locator

4 Introduction

This document provides a description of the mobile device security framework for DVB-CBMS systems. This document complements the CBMS Service Purchase and Protection Open Framework, specifying a set of standard security services for supporting varied Key Management Systems. The Mobile Device Security Framework allows for horizontal deployment of devices, supports Key Management System interoperability, and facilitates secure mobile device implementations.

The basic concept governing this specification is the CBMS Service Purchase and Protection Open Framework approach. According to this approach, a number of basic building blocks are standardised to provide interoperability. Additional non-standardized blocks may be used where required for added value. These additional non-standardized blocks may be hooked into the system using a number of standard interoperability points.

5 Mobile Device Security Architecture

5.1 Content and Service Protection Architecture

An overview of the Content and Service Protection architecture is depicted in Figure 1: -

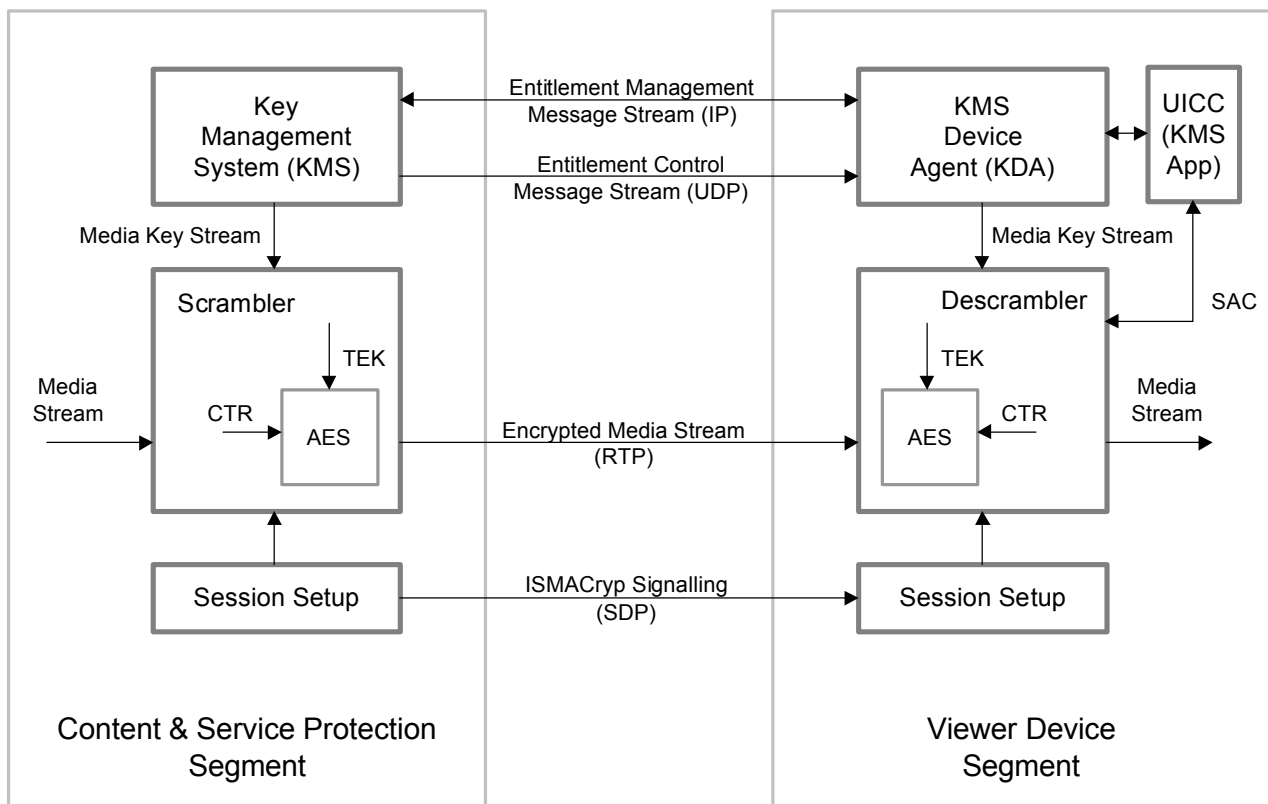


Figure 1: Content and Service Protection Architecture

5.2 Viewer Device Segment

The security mechanisms employed in the device are summarised as follows: -

1. Reception and processing of Key Management authorisation/rights stream – Entitlement Management Messages (EMMs).
2. Service access control – decryption of the Encrypted Media Key Stream – Entitlement Control Messages (ECMs)
3. Decryption of the encrypted media stream based on ISMACryp 1.0: AES-128 in Counter Mode - Session Setup based on SDP

5.2.1 Key Management System Device Agent (KDA)

The Key Management System Device Agent (KDA) contains vendor-specific logic required to control the descrambling process for a specific Key Management System.

The KDA performs the following security functions: -

1. Reception of EMMs from the KMS by means of the broadcast or interactive networks.
2. Secure generation of device authorisations from the received EMMs.
3. Reception of the ECM stream for the selected service from the broadcast network.
4. Secure generation of the Media Key Stream (TEKs) from the received ECM stream.
5. Applying the Media Key Stream to the descrambler, controlling the decryption of the Encrypted Media Stream.

The KDA typically communicates with a smart card to perform secure generation of the TEKs from the received ECMs. The KDA also establishes a Secure Authenticated Channel (SAC) between the smart card and the descrambler to control the secure exchange of TEKs.

5.2.2 Descrambler

The media stream is decrypted with the TEKs using ISMACryp 1.0 AES-128 Counter Mode. The details of the Content Decryption scheme can be found in Reference [7].

5.2.3 Session Setup

The Session Information used by the KDA for controlling access to the Service Layer is carried in its Session Layer using SDP.

5.2.4 UICC

Security-conscious operators will encourage the use of viewer devices with some secure hardware installed, specifically a Universal Integrated Circuit Card (UICC). A KMS application shall reside on the UICC.

If used, the UICC card shall be compliant with 3GPP 43.019, 51.014, 51.011 (SIM) and 3GPP 31.101, 31.102, 31.111, ETS 102.221 & 102.223 (USIM). Any secure KMS device functions will be contained as an application on this multi-application UICC.

The interface between the device and the UICC is specified by ISO 7816-4. The content of the messages exchanged on this communication link is outside the scope of this specification.

The KMS application residing on the UICC performs the following security functions under control of the KDA:

1. Secure generation of device authorizations from the received EMMs.
2. Secure generation of the Media Key Stream (TEKs) from the received ECM stream.
3. The KDA and the UICC may optionally establish a Secure Authenticated Channel (SAC) depending on the lifespan of the keys and the commercial value of the content.

6 Key Management System Device Agent

This section describes a Java Platform required to support the KDA. The purpose of the Java platform is to allow different KDAs access to the security services of the device using standard Application Program Interfaces (APIs). The KDA Platform is specified as an extension to the widely deployed MIDP 2.0 profile of J2ME.

6.1 Overview

The KDA is implemented as a Java application, which runs on the 'KDA platform'. A KDA is required by the Key Management System to operate with the Mobile TV application on the device to protect access to Mobile TV services and support a wide range of purchase models, such as subscription, pay-per-view, rental etc, according to the operator defined business rules.

The KDA interacts with the descrambler and the KMS application on the UICC to realize the KMS-specific security functions. The KDA allows a Secure Authenticated Channel (SAC) to be established between the KMS application and the descrambler for secure delivery of the Media Key Stream within the device. As such, the KDA Platform supports interfaces to the KDA to allow communication with the descrambler and the UICC. Additionally, the KDA application may access the interaction channel for direct communication with the KMS.

The interface that the KDA application exposes to the Mobile TV application is KMS-specific. A standard KDA interface, whilst beyond the scope of this document, is not precluded. Such an interface could be defined to support additional interoperability between the Mobile TV and KDA applications (JSR211 defines a content handler API that may be suitable for this purpose). However, any such API will naturally restrict the functionality of the KDA that implements it and as such is not recommended for most applications.

The KDA platform supports the KDA application being deployed according to the following scenarios: -

1. A library component integrated within a single Mobile TV Java application
2. A standalone Java application that runs concurrently with a Mobile TV Java application
3. A standalone Java application that runs concurrently with a Mobile TV native application

Additionally, this specification does not preclude device manufacturers from deploying native KDA implementations in addition to supporting KDAs via the KDA platform.

6.1.1 J2ME Mobile Information Device Profile (MIDP)

J2ME, the Java 2 platform Micro Edition, is the Java runtime that is optimized for portable devices. It has become commonplace in almost all mobile phones and PDAs, and is thus a natural choice for the KDA Platform, allowing terminal vendors to reuse the existing J2ME platform already present on their devices.

J2ME is available in the following configurations which detail the basic functionality of the virtual machine:

- CDC (Connected Device Configuration)
- CLDC (Connected Limited Device Configuration)

CLDC is the more prevalent configuration for portable devices.

J2ME defines several profiles, which specify the higher level API and behavior of the J2ME VM environment. The Mobile Information Device Profile – version 2 is the common profile found in portable devices today, and is the profile that the KDA Platform is built on.

6.2 KDA Platform

The KDA shall be implemented as a MIDlet according to this profile. The mobile device shall provide a J2ME MIDP2.0 platform for running KDA applications.

Figure 2 shows the KDA Platform.

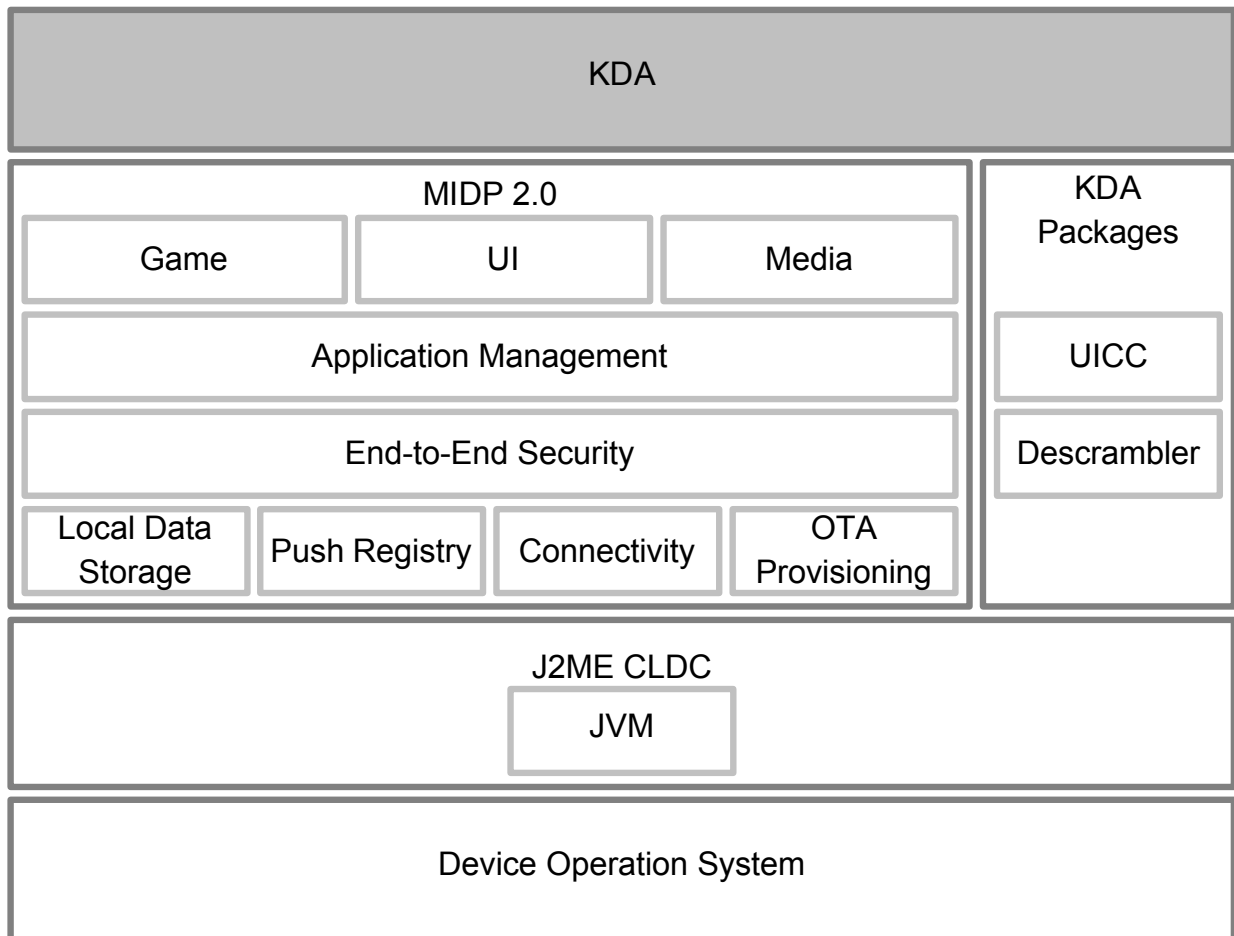


Figure 2: KDA Platform

The KDA Platform extends the MIDP 2.0 profile (JSR-118) with several extension APIs to give access to the various system components that may be required by the KDA:

- The Descrambler API is provided as a means for controlling the Encrypted Media Descrambler component.
- The standard SATSA-APDU (JSR-177) is provided as a means of communication with Secure Hardware (SHW) such as a smart card.

KDAs may use other MIDP 2.0 resources. These will typically include:

- HTTP network access by means of the Generic Connection Framework (GCF) where available. It is important to note that certain devices may have limited or no access to a two-way network connection. Therefore, implementers of KDAs are encouraged to enable the KDAs to support operation in a one-way broadcast environment.
- RMS storage for persistent data. RMS storage is typically not secure, and should be used for storing insensitive data only.
- MIDP 2.0 User Interface controls for OSDs and other user interaction.

The KDA Platform utilizes the J2ME over-the-air download capability for KDA provisioning, and extends this to support one-way delivery of KDAs over a broadcast network. All downloaded KDAs must be properly signed and provisioned with the necessary security permissions to access the KDA Platform extensions.

KDAs may appear in the normal J2ME application menu to allow users to delete them. KDAs may provide a user interface, which may be invoked from the normal startApp() method of the MIDlet.

6.2.1 Interaction Channel

Where available, the MIDP 2.0 profile provides HTTP and HTTPS access for communicating via the interaction channel over the two-way network. As such the KDA Platform provides the KDA with access to these protocols. Some devices may not have an available back channel. KDAs should handle the exceptions generated by the Generic Connection Framework in such cases gracefully.

6.3 Security Background

The J2ME MIDP2.0 platform offers a restricted execution environment. Access to restricted APIs is granted according to the security policy on the device. This policy defines protection domains, where each domain has a set of rules granting access to some or all of the restricted APIs. Each protection domain is typically associated with a root signing key-pair and certificate.

To provide a MIDlet with access to restricted functionality, the MIDlet must be put into the necessary protection domain. Putting a MIDlet into the protection domain is accomplished by signing the MIDlet with a private key for which the signer has a corresponding certificate chain whose root is the aforementioned protection domain root certificate. The signature and certificate chain are included with the MIDlet descriptor. When the MIDlet is installed the signature and the certificate chain are verified. If valid, the MIDlet is put into the protection domain associated with the root of the certificate chain.

Although KDAs may not be running in a completely secure environment, some security is necessary to prevent downloading of unauthorized KDAs. Unauthorized KDAs could enable denial of service attacks by stopping communication to the Key Management System or by not forwarding an EMM and an ECM to the SHW. The KDA has an interface for communicating with the SIM/WIM/USIM. Protecting the elements that interfaces with the SIM/WIM/USIM enables increasing the security of the Key Management System's secrets.

The KDA Platform APIs encapsulate access to all resources in the device necessary for running a KDA MIDlet. All KDA specific extension APIs must be restricted to a KDA protection domain. All KDA specific extension APIs must be completely disallowed in ALL other protection domains. A user must not be prompted if a MIDlet in any other protection domain attempts to access KDA extensions.

The KDA protection domain root will be used to provide signed KDAs. This domain must also grant access to necessary standard restricted interfaces, such as SATSA (JSR-177), HTTP and HTTPS.

The J2ME MIDP2.0 platform provides for authentication of a KDA MIDlet during download. The KDA MIDlet shall be stored as a digitally signed JAR file (the format used to download them). Thus the same security model will be used when the KDAlets are downloaded or retrieved from the local storage. The security model is described in JSR-118 RSP.

The KDA platform may support the option to receive encrypted KDAs. This can be useful where some semi-sensitive aspects of the KMS are contained in the KDA.

6.4 KDA APIs

This section specifies the APIs provided to the KDA by the KDA Platform.

6.4.1 UICC

The KDA Platform provides an implementation of SATSA-APDU (JSR-177) package for communication with the KMS application residing on a UICC. The KDA utilizes the SATSA-APDU for communication with ISO-7816-4 compliant UICCs.

This specification does not define the specific commands that the KDA may send the UICC. These commands are key management system specific.

6.4.2 Generic Connection Framework

The Generic Connection Framework (GCF), is a J2ME API that provides a straightforward hierarchy of interfaces and classes to create connections (such as HTTP, datagram, or streams) and perform I/O.

As the name implies, the GCF provides a generic approach to connectivity. It is generic because it provides a common foundation API for all the basic connection types - for packet-based (data blocks) and stream-based (contiguous or sequence of data) input and output.

Even though MIDP 2.0 defines a number of connection types, HTTP/HTTPS are the only connection type vendors must support. To ensure interoperability of KDAs, the following additional connection types are mandated by the KDA platform: -

1. javax.microedition.io.SocketConnection,
2. javax.microedition.io.UDPDatagramConnection,

These APIs may be used by the KDA for accessing ECM and EMM streams as well as for communication via the interaction channel.

6.4.3 Descrambler

The Descrambler APIs provides methods for the KDA to securely load TEKs in the descrambler engine. The Descrambler APIs provides methods for negotiating an authenticated secure channel with either the KDA or the SHW. In most cases where SHW is present, the SHW will give out TEKs only over a secure channel with the Descrambler. These encrypted messages are delivered from the SHW to the Descrambler by the KDA, but the KDA cannot access or modify the keys.

The APIs define two java interfaces: the Descrambler interface and the DescramblerContext interface.

The first one represents the underlying descrambler and is used to establish a secure channel with it. As a rule, a platform will usually provide a single instance of a class implementing this interface and provide this instance to the KDA as part of the startup sequence of the CBMS platform.

The second represents a particular descrambler context, into which decryption keys will be loaded by the KDA using the secure channel previously established through the Descrambler interface. This allows for multiple DescramblerContext sharing the same secure channel to be instantiated in order to handle multiple descrambling sessions, each using different keys. One or more objects implementing the DescramblerContext interface are provided to the KDA by the platform, as part of the process of tuning to a protected service, and/or as the repository of the TEK contained in an ECM.

6.4.3.1 Descrambler interface

The Descrambler interface provides methods for establishing a secure connection with the underlying descrambler. This connection must be established before the descrambler may be used. The cryptographic protocol underlying the following API is described in chapter 9.

```
package dvb.cbms;

/**
 * Represents the descrambler engine, with which a secure connection must
 * be established before it may be used.
 */
public interface Descrambler
{
    /** Retrieves the Descrambler's unique identifier, which can, in turn
     * be used to retrieve the descrambler's public key from a trusted
     * source.
     * @return A byte array containing the public ID
     */
    public byte[] getPublicID();

    /** Loads a new session key in the Descrambler
     * @param sessionKey is the session key seed encrypted with the
     * Descrambler's public key.
     */
    public void loadSessionKey(byte[] sessionKey);
}
```

The `getPublicID()` method returns the unique identifier of the descrambler. This identifier is assigned by the descrambler manufacturer and may be used by the KDA to retrieve the public key of the descrambler from a trusted source. The manner of this retrieval is outside the scope of this specification, but it must be noted that the establishment of the trust, by the KMS, in the descrambler resides entirely in the security of this operation and is the responsibility of the KMS.

The `loadSessionKey()` method is used to establish a session key between the descrambler and the KDA. The `sessionKey` parameter is the value $g^x \bmod p$ as defined in 9.3.1. Once this method has been called at least once, the secure connection is considered as established and the `DescramblerContext` interface may be used to load keys in the descrambler. This method may be called at any time to establish a new session key.

6.4.3.2 DescramblerContext interface

The `DescramblerContext` interface provides methods for loading decryption keys in the descrambler.

The platform **MUST** support instantiating at least one `DescramblerContext`, and **MAY** allow instantiating more in order to support the descrambling of multiple streams.

Each `DescramblerContext` **MUST** support simultaneous loading of two keys with their associated Key Indicators in two separate memory slots. To allow the KDA to decide which key it needs to replace when loading a new key, the targeted slot is signaled in the `loadKey()` method.

```
package dvb.cbms;

/**
 * Represents a descrambler context.
 * Multiple descrambler contexts may be instantiated to handle decryption
 * of multiple streams with different keys.
 * Multiple instances of DescramblerContext may be related to the same
 * Descrambler instance, thereby sharing the same session key.
 */
public interface DescramblerContext
{
    /** Loads a TEK into the Descrambler. The Descrambler must support the
     * simultaneous loading of a minimum of two keys to allow synchronized
     * switching of one key to the next.
     * Throws ArrayIndexOutOfBoundsException if the specified slot is not
     * available in this descrambler. Slots 0 and 1 MUST be supported.
     */
    @param slot          The slot in which this key is stored
    @param keyIndicator  The key indicator identifying this key
    @param encKey        The TEK, encrypted with the session key
    /**
     public void loadKey(byte slot, long keyIndicator, byte[] encKey)
         throws ArrayIndexOutOfBoundsException;
    */
}
```

The `encKey` parameter of the `loadKey()` method is the decryption key, encrypted with the current session key: the value m as defined in 9.3.2.

Note: the Key Indicator uniquely identifies, for one particular `DescramblerContext`, the key to use to decrypt a block of data. This key indicator is signaled with the encrypted data, according to [4]. The same key indicator can be used in another `DescramblerContext` with a different key. The key indicator is not necessarily globally unique.

7 The KDA Life Cycle

The following methods of upgrading/replacing the KDA may be supported:

- A user may initiate a KDA upgrade from a device menu.
- The KDA Platform may initiate a KDA download. This would happen, for example, when the platform determines that the KDA necessary for accessing requested content is not installed.

- The KDA may be replaced by firmware upgrade or application download via a PC (exactly the same as any other MIDlet).

7.1 Loading a new KDA

The KDA is a replaceable and upgradeable security component. This approach allows an operator to provide an OTA update for a compromised KDA or replace an active KDA with a KDA from another KMS vendor. Replacement of a KDA can be done by means of the broadcast stream or the return path.

When accessible, the return path can be used for KDA installation/upgrade. An HTTP URL is provided to the KDA platform, which must return a JAD (Java Application Descriptor) that can be used to download the KDA using the standard OTA protocols.

The KDA may be delivered by means of the broadcast channel. This is the only way that one-way devices can receive KDAs. A socket URL, such as socket://225.0.0.1:8001, must be provided by the KMS. The KDA Platform will then receive the new KDA MIDlet from the data carousel at the specified address.

Once the new KDA is downloaded, the KDA platform invokes the standard Java Application Manager to install the KDA, as specified in section 6.3. During KDA upgrade, if the KDA name and protection domain has not changed, the Java Application Manager will leave all KDA persistent data stored in the RMS intact.

8 UICC

8.1 Application IDentifier (AID)

All UICC offering the KMS functionality must be recognizable as such. An Application IDentifier (AID) needs to be defined and registered at the ETSI-SCP offices.

The structure of the AID is given in the TS 101 220 [9].

The AID consists of a Registered application provider IDentifier (RID) of 5 bytes and a proprietary application IDentifier eXtension (PIX) of up to 11 bytes.

Registered application provider Identifier (RID)	Proprietary application Identifier eXtension (PIX)
5 bytes	≤ 11 bytes

RID: “A000000009” (ETSI)

PIX:

Digits 1 to 4: Application code: KMS (to be assigned by ETSI)

Digits 5 to 8: Country code

Digits 9 to 14: Application provider code. This contains the CAS_ID (2 bytes) encoded as 5 BCD digits.

Digits 15 to 22: Application provider field optional (up to 8 digits): reserved for the KMS provider's use

An UICC may contain several applications. Typically the UICC would contain an USIM application and one or several KMS applications. Each application will have a specific AID.

8.2 Application and Logical Structure of the Card

The application and logical structure for the UICC is defined in ETSI TS 102 221 [8].

For the UICC containing a USIM application, the ETSI TS 31.102 [11] gives the definition of files in the DF telecom and ADF USIM.

All UICC contains in the EF_{DIR} file at the MF level, the list of AID installed in the card. For the UICC offering the KMS functionality, the EF_{DIR} file contains a record with an AID corresponding to the KMS application (ie with an application code equal to KMS and the application provider code of the KMS provider).

The Application DF (ADF) of the KMS application contains all the DF and EF of the application. At least one EF is present in the ADF of a KMS. This file is the EF_{KMSID} file containing the identifier of the KMS application.

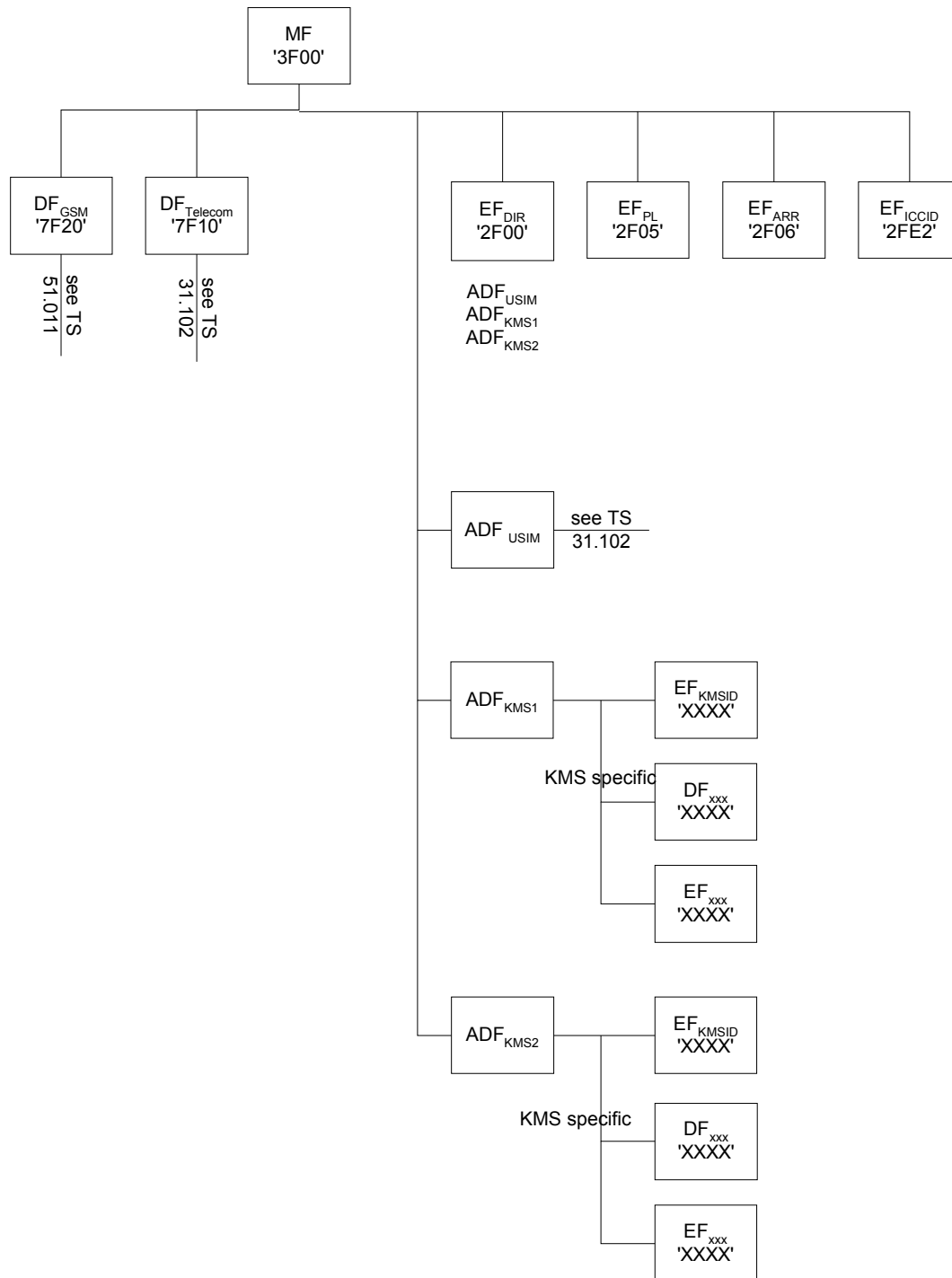


Figure 2: UICC File Structure

8.3 EF_{KMSID} (KMS Identifier)

This EF contains the KMS Identifier commonly known as CAS_ID:

Identifier: 'TBD'		Structure: transparent		Mandatory
SFI: '01'				
File size: 2n bytes			Update activity: low	
Access Conditions:				
READ		ALW		
UPDATE		ADM		
DEACTIVATE		ADM		
ACTIVATE		ADM		
Bytes	Description		M/O	Length
1 to 2	CAS_ID		M	2 bytes
3 to 2n	Other proprietary records		O	2n-2 bytes

CAS_ID: This 16 bits field identifies the KMS (uimbsf). Allocation of the values for this field is found in ETR 162 [7].

The file may contain additional proprietary records.

8.4 KMS application selection

After UICC activation (see TS 31.101 [10]), the KDA application creates an APDU connection to communicate with the KMS smart-card application using the AID of the application as a parameter. The logical channel management is then handled by the JSR177 API implementation, which requests the UICC to allocate an unused logical channel.

The JSR177 API will perform the following commands when the APDUConnection method is invoked:

- **MANAGE CHANNEL:** There can be one active selectable application session on a given logical channel. Therefore in order to activate the new KMS application session in parallel with the USIM application selected on the channel 0, a new channel has to be opened using the MANAGE CHANNEL command.
- **SELECT:** On this new channel, the KMS application may be selected. The selection of the application is performed using the SELECT function with the AID of the selected KMS application as a parameter.

An APDUConnection supports exchange of APDU commands encoded in the format that conforms to ISO7816-4. Each APDU connection has a logical channel reserved exclusively for it. Channel 0 is reserved for the USIM application."

9 Secure Authenticated Channel Protocol

This section describes the mechanisms used to establish a Secure Authenticated Channel (SAC) between the KMS application residing on the UICC and the Descrambler.

The SAC ensures the secrecy of the communication between the KMS and the descrambler and prevents TEK extraction by an eavesdropper. The SAC also provides authentication of the descrambler by the KMS, thus preventing unauthorized devices, such as a PC, from extracting TEKs from the KMS. The SAC does not, however, provide authentication of the KMS by the descrambler, as protection of the descrambler against misuse is not the primary goal of the SAC. The authentication of the KMS application may be done by the KDA.

9.1 High level description of the SAC

The SAC is based on asymmetric cryptography. Each descrambler is assigned a unique triplet:

- A Unique Identifier
- A Private Key, kept secret in the descrambler
- A Public Key, made available to the KMS

The descrambler makes its Unique Identifier available to the KDA through a public interface. This Unique ID may then be used by the KMS to request from its Head-end (server) the corresponding Public Key of the descrambler. The manner in which the KMS communicates with its Head-end and retrieves the Public Key of the descrambler is out of scope of this document, but it may involve using the interaction channel or, for unconnected devices, it may be based on the user contacting the service provider (Call center, Web, etc) to communicate the descrambler Unique ID and request that the public key be sent to its KDA.

Once in possession of the Public Key of the descrambler, the KMS generates a random session key seed, encrypts it with the descrambler's public key and provides the result to the descrambler. Both the KMS and the Descrambler then derive a session key from the session key seed.

Once both parties share this session key, all TEKs are encrypted by the KMS before they are passed to the descrambler, which decrypts them before use.

9.2 The cryptographic keys and parameters

The cryptographic protocol is based on the El Gamal Key Agreement (half-certified Diffie-Helman), specified in ISO 11770-3 and described in [13], chapter 12.51.

9.2.1 The Descrambler's keys

For each Descrambler, the following values are generated:

p - a 1536-bit prime modulus

q - a 160-bit prime divisor of $p-1$

g - a generator of order q

x - a randomly or pseudo-randomly generated integer with $0 < x < q$

The public key is composed of $(p, q, g, g^x \bmod p)$, and the secret key is x .

9.3 The SAC protocol

9.3.1 Session key establishment

The KMS picks a random $0 < y < q$, sends $g^y \bmod p$ to the descrambler and computes $s = (g^y)^x \bmod p$

The descrambler computes $s' = (g^x)^y \bmod p = (g^y)^x \bmod p = s$

At the end of this protocol the KMS and the descrambler share s , a 1536 bit value. This is hashed into a 160 bit value and the 128 high order bits of the hash are extracted to obtain the session key k :

$$k = [SHA-1(s)]_{MSB-128}$$

9.3.2 Secure key exchange

Following the session key establishment, the KMS may load keys in the descrambler, encrypting them with the session key k :

$$m = AES(TEK, k)$$

In this version of the specification, both the TEK and the session key are 128 bit keys and AES is used directly on the TEK in ECB mode.

10 Bibliography

The following material, though not specifically referenced in the body of the present document (or not publicly available), gives supporting information.

- ETSI ETR 289: "Digital Video Broadcasting (DVB); Support for use of scrambling and Conditional Access (CA) within digital broadcasting systems".
 - Implementation guidelines for use of telecommunications interfaces in the Digital Broadcasting systems (DVB Project Office).

11 History