INTERNATIONAL STANDARD

ISO/IEC 23465-1

First edition
2023-02

Card and security devices for personal identification — Programming interface for security devices —

Part 1: Introduction and architecture description

Cartes et dispositifs de sécurité pour l'identification personnelle — Interface de programmation pour dispositifs de sécurité —

Partie 1: Introduction et description de l'architecture

Reference number
ISO/IEC 23465-1:2023(E)

Contents

Page

Foreword

ISO (the International Organization for Standardization) and IEC (the International Electrotechnical Commission) form the specialized system for worldwide standardization. National bodies that are members of ISO or IEC participate in the development of International Standards through technical committees established by the respective organization to deal with particular fields of technical activity. ISO and IEC technical committees collaborate in fields of mutual interest. Other international organizations, governmental and non-governmental, in liaison with ISO and IEC, also take part in the work.

The procedures used to develop this document and those intended for its further maintenance are described in the ISO/IEC Directives, Part 1. In particular, the different approval criteria needed for the different types of document should be noted. This document was drafted in accordance with the editorial rules of the ISO/IEC Directives, Part 2 (see www.iso.org/directives or www.iec.ch/members_experts/refdocs).

Attention is drawn to the possibility that some of the elements of this document may be the subject of patent rights. ISO and IEC shall not be held responsible for identifying any or all such patent rights. Details of any patent rights identified during the development of the document will be in the Introduction and/or on the ISO list of patent declarations received (see www.iso.org/patents) or the IEC list of patent declarations received (see https://patents.iec.ch).

Any trade name used in this document is information given for the convenience of users and does not constitute an endorsement.

For an explanation of the voluntary nature of standards, the meaning of ISO specific terms and expressions related to conformity assessment, as well as information about ISO's adherence to the World Trade Organization (WTO) principles in the Technical Barriers to Trade (TBT) see www.iso.org/iso/foreword.html. In the IEC, see www.iec.ch/understanding-standards.

This document was prepared by Joint Technical Committee ISO/IEC JTC 1, *Information technology*, Subcommittee SC 17, *Cards and security devices for personal identification*.

A list of all parts in the ISO/IEC 23465 series can be found on the ISO and IEC websites.

Any feedback or questions on this document should be directed to the user's national standards body. A complete listing of these bodies can be found at www.iso.org/members.html and www.iec.ch/national-committees.

Introduction

Integrated circuit card (ICC) technologies and solutions are widely deployed around the world, but systems for identity tokens and credentials are quickly changing. In this context, the application protocol data unit (APDU) protocol defined in the ISO/IEC 7816 series is becoming, in some cases, a hindrance to the integration of integrated circuits (ICs) (as security devices) in environments such as mobile phones, handheld devices, connected devices (e.g. M2M, IoT) or other applications using security devices.

Several stakeholders are not familiar with, or not very fond of the APDU protocol because of its complexity. They will often circumvent its constraints by requesting an abstraction layer hiding IC specifics. Although the security mechanisms of security devices are well defined in ISO/IEC 7816-4 their implementation and application differ from vendor to vendor and the complexity overstrains most of the application developers.

In software development, a common way to simplify the usage of complex systems is the definition and application of application programming interface (API) functions to access the IC within the devices. Specific knowledge of APDU protocols and details of the IC implementation is not necessary anymore. Also, the complexity and details of the implementation of the security model and the security policy can be shifted from pure application development into system design of the electronic device and its related software.

Therefore, this document is geared towards software (SW)-architects, application programmers or specification developers developing software applications using and addressing ICs as security devices within operating systems or their components.

The projected applications can run on different software and hardware environments. Generalisation of the API definition is key and the dependencies on specific runtime environments and equipment are kept out in principle.

Existing runtime environments already support the access to IC as security devices using different specific APIs, e.g. OpenMobileAPI,[10] PKCS#11,[12] but they always implement a proprietary interface and middleware, which is not commonly applicable. However, even solutions based on those kinds of middleware are perceived as cumbersome in some systems. The market looks for a middleware memory footprint to be as low as possible. This document also aims to overcome or mitigate those issues by proposing a new approach that would preserve ICC functionality and allows for a seamless ICC portability onto new systems.

Since the system is designed for easy support by mobile operation systems, mobile operating system (OS) designers/ implementers are encouraged to support these standardized APIs to access any embedded secure element (eSE) within the mobile device.

In the context of mobile devices, there is a necessity for trusted computing, e.g. by dedicated security hardware. The proposed API helps the application implementer with a standardized common interface to such trusted IC.

The ISO/IEC 23465 series focuses on a solution by designing an API and a system with the following characteristics.

— It offers a set of API calls related to multi-sectorial ICC functionality, derived from the ISO/IEC 7816 series and other ICC related standards.

— It defines the sub-system to perform the conversion from the API function to the interface of the security device (e.g. APDU-interface), called Proxy.

— It results in a description of solutions with no middleware or very little middleware memory footprint (i.e. simplified drivers).

— It defines the simplified ICC capabilities, the discoverability (i.e. with significantly less complexity than ISO/IEC 24727) and examples of usages.

The ISO/IEC 23465 series is comprised of three parts each focusing on a specific topic:

— ISO/IEC 23465-1 (this document): provides an introduction to the series and a short overview of the architecture;

— ISO/IEC TS 23465-2: defines the API for client applications allowing incorporation and usage of security devices;

— ISO/IEC TS 23465-3[1]: describes the software called Proxy which provides different services e.g. to convert the API calls into serialized messages to be sent to the security device.

1) Under preparation. Stage at the time of publication: ISO/IEC DTS 23465-3.

Card and security devices for personal identification — Programming interface for security devices —

Part 1: Introduction and architecture description

1 Scope

This document introduces and describes the concept of the application programming interface (API) to security devices with the intention to simplify the usage of commands and mechanisms defined by the ISO/IEC 7816 series.

This document gives guidelines on:

— the system overview and description of the system of the programming interface;

— the architecture description;

— the data model in general, used by the API;

— the use cases and the usage model of the API.

2 Normative references

There are no normative references in this document.

3 Terms and definitions

For the purposes of this document, the following terms and definitions apply.

ISO and IEC maintain terminology databases for use in standardization at the following addresses:

— ISO Online browsing platform: available at https://www.iso.org/obp

— IEC Electropedia: available at https://www.electropedia.org/

3.1
client
any type of entity requesting services from a *security device* (3.7)

3.2
ISO/IEC 23465 API
software interface defined in ISO/IEC TS 23465-2

3.3
middleware
software (SW) component allowing two systems from different or similar operating systems interconnection (OSI) layers to communicate with each other

3.4
operating systems interconnection model
OSI model
conceptual model that characterizes and standardizes the communication functions of a network or computing system without regard to its underlying internal structure and technology

3.5
secure digital memory card
SD-card
secure storage media using non-volatile memory

3.6
proxy
sub-system to perform conversion from the application programming interface (API) function to the interface of the *security device* (3.7)

3.7
security device
tamper-resistant secure hardware component which is used in a device to provide the security, confidentiality, and multiple application environment required to support various business models

Note 1 to entry: It may exist in any form factor, e.g. embedded or integrated SE, SIM/UICC, smart card, SD-card.

3.8
serialization
translation of data structures or object state into an octet string for transmitting or storing

3.9
trusted execution environment
TEE
aspect of the mobile device comprising hardware and/or software which provides security services to the mobile device computing environment, protects data against general software attacks and isolates hardware and software security resources from the operating system

[SOURCE: ISO 12812-1:2017, 3.60]

3.10
use case
list of actions or event steps typically defining the interactions between a role and a system to achieve a goal

Note 1 to entry: A role is known as an actor in the Unified Modelling Language.

4 Symbols and abbreviated terms

APDU	application protocol data unit
API	application programming interface
BLE	bluetooth low energy
CBOR	concise binary object representation
DF	dedicated files
eSE	embedded secure element
eSIM	embedded subscriber identity module
IC	integrated circuit
ICC	integrated circuit card
IDL	interface description language
iSIM	integrated subscriber identity module
I^2C	inter-integrated circuit
JSON	java script object notation
NFC	near field communication
PCB	printed circuit board

PC/SC	personal computer/smart card
OSI	open systems interconnection
OTA	over the air
SD	secure digital (memory card)
SE	secure element
SIM	subscriber identity module
SoC	system on chip
SPI	serial peripheral interface
SW	software
TEE	trusted execution environment
UICC	universal integrated circuit card
USB	universal serial bus
WIFI	wireless communication technology, defined by the Wifi consortium

5 System overview

5.1 Conditions of use

The utilisation of an API for security devices defined in the ISO/IEC 23465 series of standards is useful in any client application software which needs services of a security device. The application software may run on any electronic devices, e.g. personal computers, terminals or mobile devices. The electronic device contains or is connected to a security device and allows its access by client applications. It is assumed that any software running on the electronic device is separated into several logical or functional layers. Such layers may be designed as a middleware between the client application software and the related security device or may be provided by the device's operating system. API is herein defined as generalised function calls from the client application software to the additional layers in the system.

The API allows the logical access to any available security device, independent from the physical form factor, the technology, the used connectivity and the applied transmission protocol. It hides the physical layer, the data link layer and the network layers of OSI model to the application. The API offers standardized methods and functions to security device services and builds either an abstraction or a subset of the underlying security device interface, e.g. the APDU interface defined by ISO/IEC 7816-4, or both. In this way, the transport, session and presentation layers are also hidden from the client application.

The API is a representation of the application layer to a security device. The application implementer does not need further details to contact, address, select and use a connected security device. But the set of the API functions still allows the application to work with the security device by retrieving all relevant information, functionality and services.

Systems implementing this API facilitate the access to the security device. Dedicated function calls and specific knowledge about the structure and architecture of the security device's application is not needed by the client application programmer. The API generalises function calls, offers less complexity and reduces the need of knowledge about details of the security device.

The API functions are resolved within an additional software, which handles the access to security devices. These additional software components are provided by the manufacturer of the electronic device. Since the number of security devices in a system is not limited, the additional software components have to be enabled to handle the different involved security devices.

Conversely, several client applications can use a single security device. It is possible for each client application to use a different application inside the security device. Furthermore, applications inside a security device are separated from each other from a security point of view.

5.2 Simple system configuration

The simplest possible configuration is outlined Figure 1. This kind of system consists (possibly among other components) of just one software environment running any application and incorporates the security device. Such configuration is, e.g. a SoC device proposed for mobile devices. In this case, the mobile device environment contains the complete client application and the security device for which the implementation is running on the same system and within the same runtime environment as the client application itself.

API calls from the client application directly lead to function calls in the security device. The security device implements the resolution of the API and handles the processing of the requested operation on the security device. Thus, from a client application programmer's point of view, the API calls from the client application just call the corresponding method implementation of the security device.

In this simple situation, there is neither any other layer in between nor is there any kind of message serialization. The API calls and the resolution of the API calls are done in the common system. To achieve this, the security device programmer normally provides a library to the system manufacturer or application.

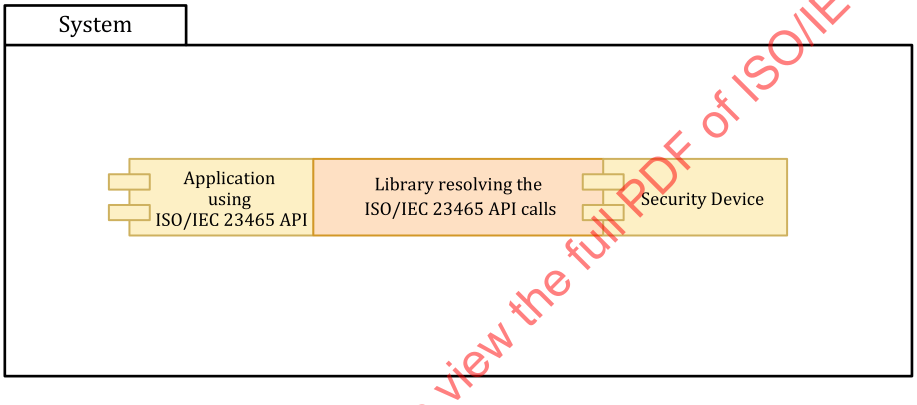
System
Application using ISO/IEC 23465 API
Library resolving the ISO/IEC 23465 API calls
Security Device

Figure 1 — Simple system configuration

5.3 Complex system configuration

Compared to the simple configuration in Figure 1, the API defined in the ISO/IEC 23465 series of standards applies also to configurations where a system contains more than one client application or more than one security device, as outlined in Figure 2. The API supports the possibility that more than one security device is available. It contains methods to select a specific security device among the available ones. This kind of selection requires another component, called Proxy, mediating between client applications and security devices. The client application calls API methods which trigger the Proxy and address the appropriate requested functionality in the selected security device.

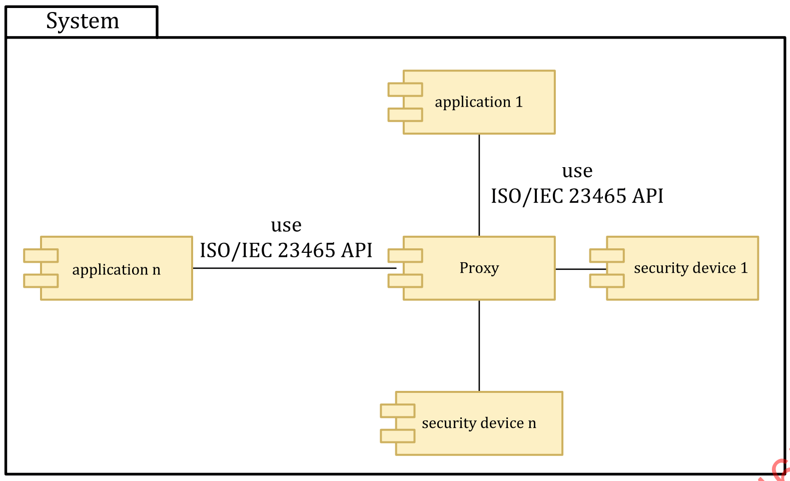

Figure 2 — Complex system configuration

Even in complex system client applications, security devices and the Proxy may run in the same runtime environment. Typically, a security device is deployed as dedicated hardware. If so, client applications and (part of) the Proxy on one hand runs in a runtime environment which differs from the security device.

5.4 Generic examples using the different configurations

5.4.1 SoC — Example of simple system configuration

The simplest configuration of a system using security devices is a system running client applications and security devices in the same runtime environment (see 5.3). A library within the runtime environment is supposed to act as the Proxy between the client application and the security device. This library performs the API call resolution and the security device access.

5.4.2 ID-systems

A slightly more complex system runs client applications and security devices in different runtime environments with separated software layers. Examples are ID-systems in general (e.g. border control, banking, health) typically holding a client application as a part of a host connected to a card reader. Security devices in this configuration are ID-cards temporarily connected to the ID-system. To simplify the development of such systems, it would be appropriate not only to standardize the interface between the client applications and the Proxy, but also to standardize the communication between the Proxy and the security devices.

Figure 3 depicts a generic simple system showing the conditions of usage with an client application using the API defined in ISO/IEC TS 23465-2 within a single runtime environment. The Proxy can be outlined as a library or as a module in a more complex reader operating system.

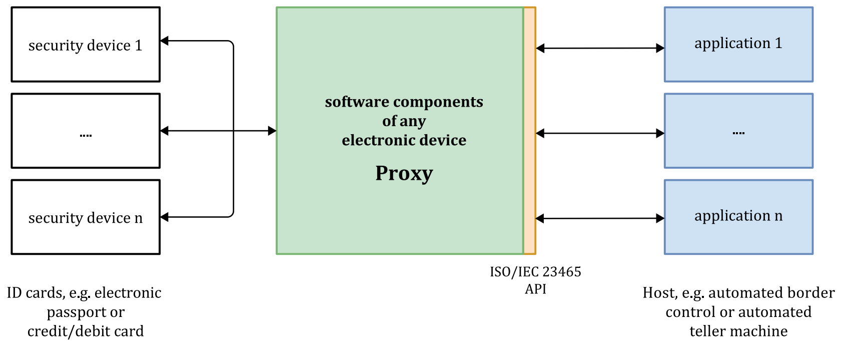
security device 1
....
security device n
ID cards, e.g. electronic passport or credit/debit card
software components of any electronic device
Proxy
ISO/IEC 23465 API
application 1
....
application n
Host, e.g. automated border control or automated teller machine

Figure 3 — ID-system with host and card reader

5.4.3 Mobile devices with multiple security devices

Figure 4 shows an example of a complex system configuration using different security devices within a mobile device (e.g. mobile phones or tablet computers). The mobile app, downloaded from an app store, requests, in the course of the client application, functionality from security devices by API function calls. The software system beneath the mobile app, introduced as Proxy in the ISO/IEC 23465 series, handles the function calls and transforms or serializes them into a data stream understandable by the security devices. Depending on the implementation of the security devices, the function calls are operated by the addressed security device and the results/responses are transferred and transformed adequately to the calling mobile app. Within mobile devices, usually a fixed number of security devices are permanently connected, e.g. USIM, ISIM, eSIM, eSE, which can be used by different mobile apps loaded on the mobile phone.

The Proxy can use additional APIs to achieve the physical access to the security device, e.g. provided by the mobile operating system. Applications running on personal computers or similar devices can use additional existing SW drivers, e.g. PC/SC, which can facilitate the access. Security device form factors of mobile devices beside SIMs are, e.g. soldered ICs on the PCB, connected USB-devices, SD-cards or connected ID cards via card reader interfaces (NFC, BLE).

ISO/IEC 7816-4 defined APDUs or other representation of messages are transmitted by the lower level transmission protocols determined by the technology used in the addressed security device. Examples for this protocol types are, e.g. USB, I^{2}C, SPI or serial communication according to ISO/IEC 7816-3, representing the physical layer in the OSI model.

The API hides all these security device details completely from the client applications. ISO/IEC TS 23465-3 gives examples on how a Proxy implements the different layers.

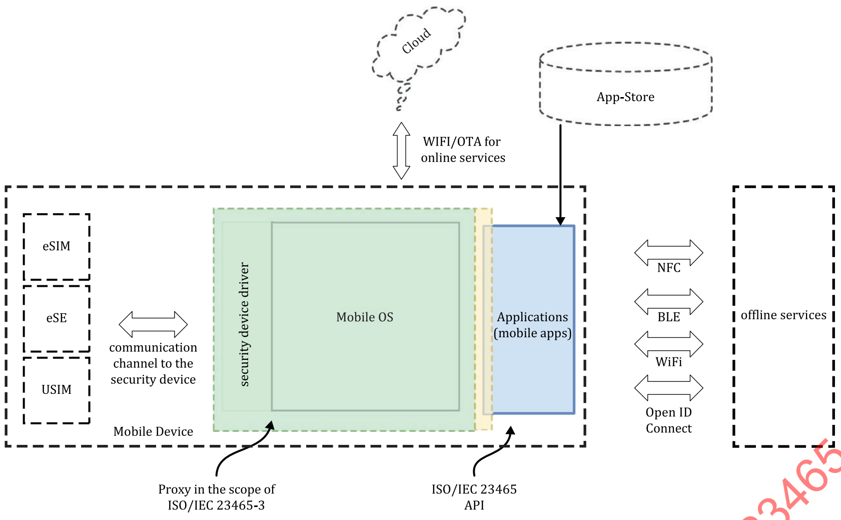
Cloud
App-Store
WIFI/OTA for online services
eSIM
eSE
USIM
communication channel to the security device
security device driver
Mobile OS
Applications (mobile apps)
Mobile Device
NFC
BLE
WiFi
Open ID Connect
offline services
Proxy in the scope of ISO/IEC 23465-3
ISO/IEC 23465 API

Figure 4 — Mobile environment using security devices

5.4.4 Mobile devices with a single security device

If a mobile device contains just one security device, which is used by several mobile apps, then the security device can be configured such that each mobile app uses a different application in the security device. Furthermore, the security device can be configured such that its applications are physically and logically separated from each other such that no interference occurs. The security device controls access to the assets in its applications. The Proxy controls that function calls from the mobile apps are directed to the corresponding application inside the security device. Figure 5 depicts this situation.

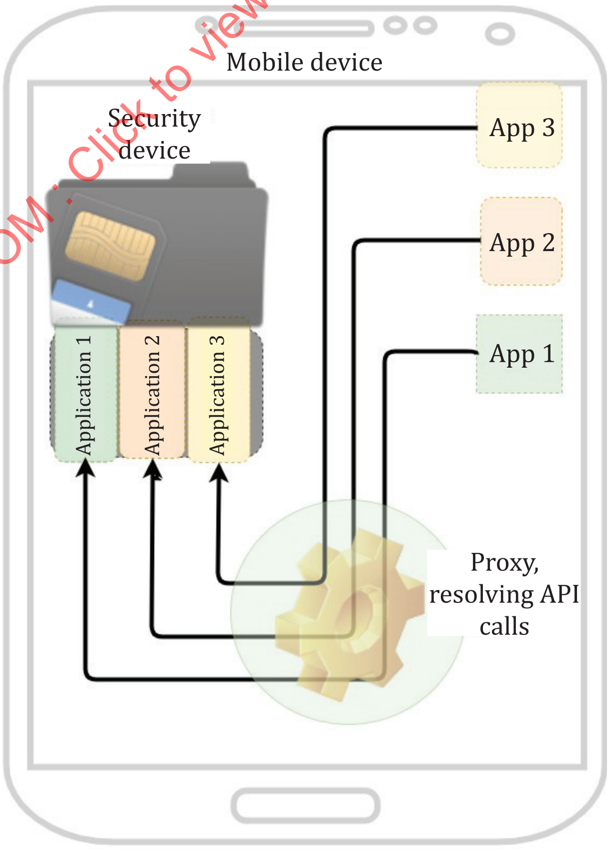

Figure 5 — Mobile device with only one security device

5.4.5 Use cases

Client applications use the API to access the security device. The API facilitates the access to, and the usage of, the security device.

Annex B outlines some use case examples.

6 Architecture model

6.1 Components

6.1.1 Client application

The client application in the context of the ISO/IEC 23465 series is a software module which fulfills a specific functionality on or within a system. The ISO/IEC 23465 series of standards does not specify which programming language is used for that software module. Furthermore, that software module can have several interfaces to internal or external entities. The ISO/IEC 23465 series of standards becomes relevant for such a software module if one of those interfaces is in conformance with the API described in the ISO/IEC TS 23465-2. The client application related to this context uses the ISO/IEC 23465 API to interact with the security device in the system.

6.1.2 Proxy

The Proxy is the software module in the system which resolves the ISO/IEC 23465 API calls from the client application and handles all operational steps to perform the required security device activity derived from these calls.

The Proxy acts in this context as a middleware between the client application and security device.

6.1.3 Security device

The security device is a separated module in the system, with the ability to receive and send messages. Typically, a message received by the security device is a command to be performed. In this case the security device calculates a corresponding response message which is then sent as a reply to the command message.

6.2 System using the API

Derived from the different examples from 5.4, the basic system which incorporates an API for security device access is outlined in Figure 6.

The different entities within the chain of the software layers correspond to specific levels of the OSI reference model (ISO/IEC 7498-1). The client application with API calls for security device access represents the application layer. Each access to a security device is managed by a set of API calls.

The API resolving software parts, the runtime environment of a mobile operating system or firmware possibly using additional APIs, and manufacturer specific middlewares for security device access are dedicated to OSI layers 1 to 6. This document compresses these parts in the entity Proxy, which is detailed in ISO/IEC TS 23465-3.

Any API call is resolved in the Proxy. An essential portion of the software is programming language related. The ISO/IEC 23465 series of standards is programming language agnostic, i.e. it is developed with no particular programming language in mind. Additional layers are necessary for performing the identification and administration, controlling the connectivity, and most importantly, resolving the references to the real objects.

A session layer related to the complex architecture, outlined in 5.3 handles the access from different client applications to multiple security devices. Depending on the system, a set of software parts, e.g.

mobile operating system environments, firmware implementations, libraries and/or middlewares, represent the presentation layer and transport layers in the OSI reference model.

Security device specific drivers, mostly provided by the manufacture of the security device, enable the generation of security device related messages, which are related to the functionality of the security device. These software parts implement the network and the data link layer.

Finally, the physical layer deals with the low-level protocols and the connection to the security devices. Well known protocols and techniques for security devices are defined in ISO/IEC 7816-3 with T=0 or T=1 protocols. There is no limitation to use other logical and physical protocols depending on the security device's technology today or in the future. Details are out of scope of this document.

The API definition is independent from any used programming language. Besides the obligatory programming language binding parts, the Proxy separates API calls into a set of basic security device related commands and translates them into security device related messages.

On the administration side, it sets-up the session between the client application and the addressed security device, handles any required secure messaging and supports the connectivity and transmission over the security device interface. An exception handling mechanism translates the security device error messages into exceptions thrown by the Proxy.

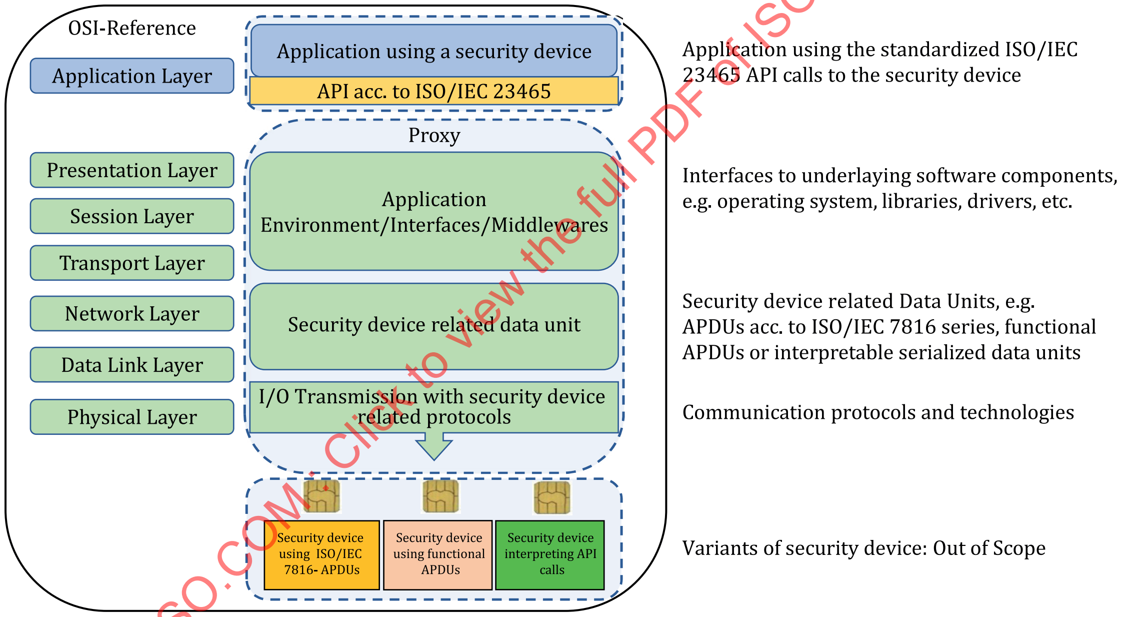
OSI-Reference
Application Layer
Presentation Layer
Session Layer
Transport Layer
Network Layer
Data Link Layer
Physical Layer
Application using a security device
API acc. to ISO/IEC 23465
Proxy
Application Environment/Interfaces/Middlewares
Security device related data unit
I/O Transmission with security device related protocols
Security device using ISO/IEC 7816- APDUs
Security device using functional APDUs
Security device interpreting API calls
Application using the standardized ISO/IEC 23465 API calls to the security device
Interfaces to underlaying software components, e.g. operating system, libraries, drivers, etc.
Security device related Data Units, e.g. APDUs acc. to ISO/IEC 7816 series, functional APDUs or interpretable serialized data units
Communication protocols and technologies
Variants of security device: Out of Scope

Figure 6 — Architecture model for a complex configuration

6.3 Data model of client application

The environments outlined in 5.4 describe several situations where client applications are combined with security devices. Their usage and the related data stored in the security device are defined in general by the requirements of the client application.

The following situations can be anticipated.

— The security device is structured according to ISO/IEC 7816-4, that implies, the security device applications are separated in DFs in the file structure of ISO/IEC 7816-4. Any client application accesses its counterpart DF on the security device, e.g. by selecting the relevant DF. When the client application uses its counterpart DF by requesting services from that DF, the Proxy sends appropriate ISO/IEC 7816 related commands, which use e.g. data and secrets within this counterpart DF.

— The security device is structured as a Java Card®[2] and contains application specific Java Card applets. This scenario allows a client application specific functionality with its own commands and data, implemented in the java card applet. Depending on the implementation, a complete security application can be processed on the security device, e.g. an ID-application or a payment application. Each security application may have its own scale of functionality and data content.

— The security device works only as a trust anchor for a client application. In this case a separation of security device application is not intended. The whole security device is related to a main application and serves, e.g. as a secure key or data storage, possibly with no further functionality.

— The security device is primarily used in another application context. Existing systems already contain security devices for their primary usage, e.g. UICC, eSIM, in mobile phones, which can be also used for other applications. The primary security device may host separated additional security device applications for new client applications. The format for the additional security device application is mandated by the system. It is in the responsibility of the Proxy to enable the access to such extended security devices.

6.4 Data management

The client application needs a set of data and information to identify and to deal with its dedicated security device.

From a high-level point of view, the generic data model of the client application requires:

— the logical interface between the client application and security device application;

— the addressing means between the client application and the security device;

— an abstracted access of secure device data by the client application by object references;

— a definition of the relationship between client application data and security device application data;

— a definition of the relationship between security policy of the client application according to the access rules of the security device application;

— a simplified usage of the security device's security policy (with support of Proxy);

— a defined security policy of the client application, e.g. access rules known by the client application, supported by the security device;

— a client application specific set of functionality, e.g. an API name space;

— a set of data related to the client application used by the client application in the course of an API function call.

The design and the definition of the client application shall take the required data management elements into account.

Figure 7 depicts the situation of a system with a data model supporting distinguished data elements, credentials and security requirements. Each element has a counterpart in the security device. The API methods and the underlaying class model support the data management.

NOTE 1 Electronic device applications, e.g. in mobile phones, can have a distributed data model with data stored in a different location in the system, but belonging together.

A generic data model allows any relation between the security device(s) with its security device applications and the corresponding client application. The client application needs to open a logical

2) Java Card is a registered trademark of Oracle and/or its affiliates. This information is given for the convenience of users of this document and does not constitute an endorsement by ISO or IEC of the product named.

link to the appropriate security device(s) and its related security device application by applying the addressing methods of the ISO/IEC 23465 API.

NOTE 2 The number of security devices are not restricted. For simplification, only a single security device is depicted in Figure 7.

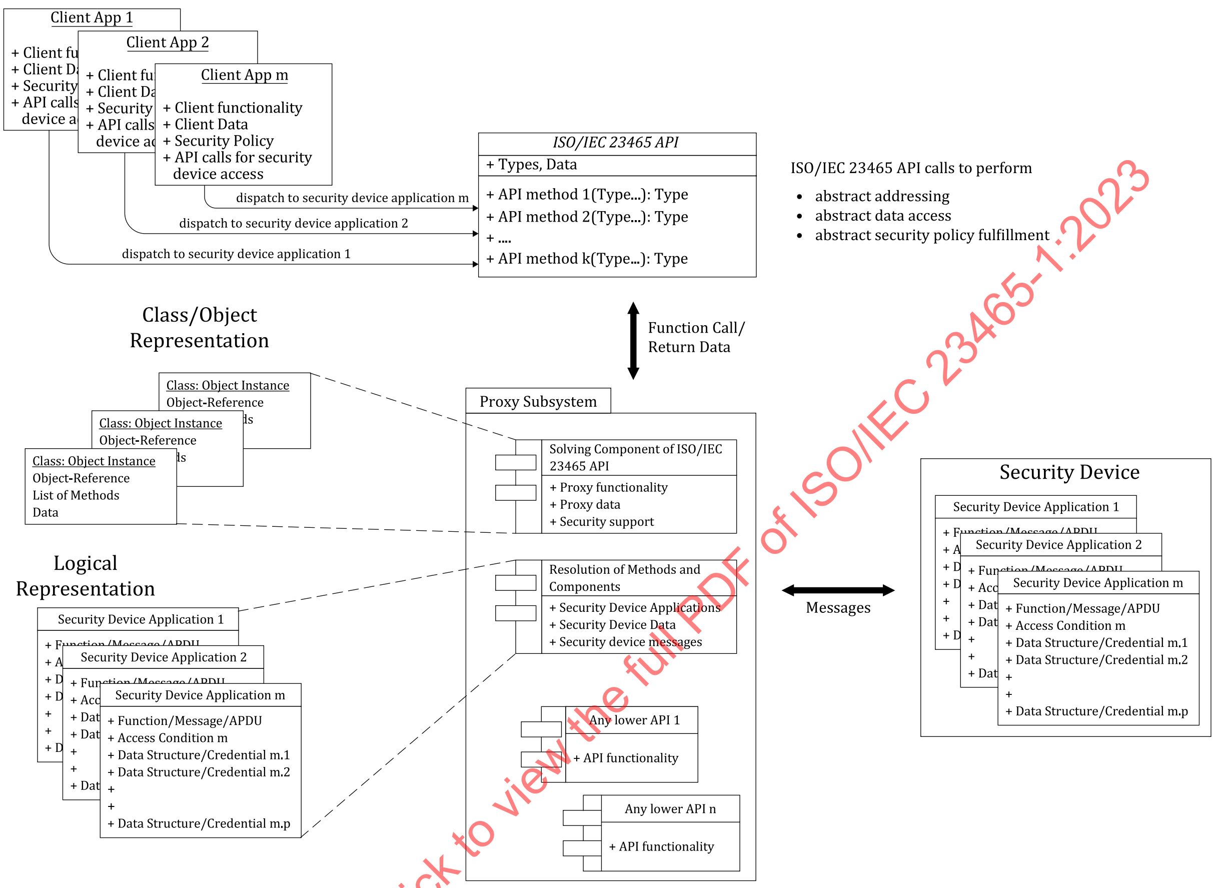
Client App 1
Client App 2
Client App m
+ Client functionality
+ Client Data
+ Security Policy
+ API calls for security device access
dispatch to security device application m
dispatch to security device application 2
dispatch to security device application 1
ISO/IEC 23465 API
+ Types, Data
+ API method 1(Type...): Type
+ API method 2(Type...): Type
+ ....
+ API method k(Type...): Type
ISO/IEC 23465 API calls to perform
• abstract addressing
• abstract data access
• abstract security policy fulfillment
Function Call/ Return Data
Class/Object Representation
Class: Object Instance
Object-Reference
List of Methods
Data
Logical Representation
Security Device Application 1
Security Device Application 2
Security Device Application m
+ Function/Message/APDU
+ Access Condition m
+ Data Structure/Credential m.1
+ Data Structure/Credential m.2
+
+
+ Data Structure/Credential m.p
Proxy Subsystem
Solving Component of ISO/IEC 23465 API
+ Proxy functionality
+ Proxy data
+ Security support
Resolution of Methods and Components
+ Security Device Applications
+ Security Device Data
+ Security device messages
Any lower API 1
+ API functionality
Any lower API n
+ API functionality
Messages
Security Device

Figure 7 — Data storage model

6.5 Security architecture

The security policy of a client application defines the requirements of specific mechanisms to be supported by the electronic system. Some of them are fulfilled inherently, some can be located in the electronic system software, but this document focusses on security methods and means of security devices and its applications.

A client application requires therefore:

— a security policy related to the access rules of the security device application;

— a simplified method and translation of the security policy into security device related commands;

— simplified means to perform complex security protocols without being involved in details (e.g. multistep authentication protocols).

The design of the client‘s security architecture has direct impact on the security conditions of a security device application. Provisioning a client application always requires an adequate provisioning of the security device mechanisms and applications.

Some of the fulfillments of these requirements are achieved with the support of the Proxy.

7 API

7.1 General

An API is part of a program provided by a software system connected and linked to other software system. This kind of programming interface is a junction on source code level.

The APIs defined in the ISO/IEC 23465 series of standards are designed to be used in software of electronic systems, e.g. readers, terminals or mobile devices. The client applications in these systems use security device services. The definition of the API is independent of any programming language. Binding of the API to a used programming language is out of scope of this document. The API implementation is not independent from the rest of the software system since the calls from the client application have to be resolved within a related software or library. Typically, the library is part of the Proxy.

To be independent from any programming language with its specific features, a standardized IDL[11] for the description of the API is used. The API is described in detail in ISO/IEC 23465-2.

7.2 Requirements for a programming interface

The API design is based on, but not limited to, the following principles:

— offers stability over a long period of time;

— provides capacity for adding additional software to the system;

— reduces complexity;

— focuses on high acceptance by a majority of the audience;

— provides an independent description which is easily adopted by any programming language;

— allows a dedicated abstraction level to be used in any electronic systems with a security device;

— offers extendability in the future when new functionalities or features are needed.

Annex A describes some design rules for a generic API definition.

7.3 API Implementation

Depending on the software system and the underlaying operating system, the characteristics of the electronic system and the used security device have to consider some boundaries for the implementation of the API resolving software parts.

These aspects also influence the characteristics and the complexity of the Proxy configuration and the architecture of the electronic system. Some of these aspects are treated in ISO/IEC TS 23465-2 and ISO/IEC TS 23465-3.

8 Proxy

8.1 General

A proxy is generally defined as a network component accepting requests from a client and dispatching the requests to other parts of the network. It serves as a broker that receives and processes the requests sends the processed data to the next partner in the network. It acts as a filter in both directions, conditionally changes the protocols and hides details of the partners. With the definition of the API and its usage in an application of the electronic system, the API call has to be resolved in assigned libraries provided by the rest of the system software. In case of an (mobile) operating system, the application layer is supported by the next layers according to the OSI reference model (see 6.2).

The term Proxy in the ISO/IEC 23465 series of standards is used for any kind of software between the client application and the security device. The Proxy acts as a broker, a translator, a filter and an interface to the security device and has the ability and knowledge to communicate finally on the physical layer with the security device.

If mobile applications use this API then a framework needs to be provided by the mobile operating system implementing the appropriate functionality. These frameworks can call other system components which possibly utilize secure hardware. The interface to the security device can use either duplex or half duplex communication and support multi-channel handling, or both.

8.2 Proxy characteristics

The Proxy consists of different software parts. As depicted in Figure 6, the Proxy aligns with these OSI layers which are relevant for the handling of the API functions and the translation into (a set of) suitable security device commands. The same has to be applied for the responses of these commands. The re-translation of the response data unit into the API requested format has to be performed by the different instances of the Proxy.

The first API-call-resolving software part in the Proxy is generally a library, which handles the programming language bounded glue code. Depending on the system software, additional software layers have to be applied until the final communication layer to the security device is used.

To be future proofed the ISO/IEC 23465 series of standards allows synchronous and asynchronous message transfer. Adequate mechanisms describe both cases, e.g. with command response pairs or with the usage of call-back functions. A general message handling structure allows the independent handling of both communication directions. The Proxy facilitates the implementation of the client application, together with the help of the API, by handling the complexity and the access to the security device. Routing, selection, addressing, channel and access handling is performed by the Proxy inherently.

The same applies for cryptographic support, e.g. application of multi-step protocols, performing secure messaging for data protection.

The Proxy itself, its mechanisms and behaviour are described in ISO/IEC TS 23465-3.

NOTE Annex C outlines different logical and physical interfaces to current and future eSE within a mobile infrastructure. Annex D explains the coexistence of the ISO/IEC 23465 API and other APIs or libraries, already used in the electronic systems, e.g. the Open Mobile API or a PKCS#11 library which can be called within the Proxy.

9 Evolution of security device types

Figure 6 depicts different variants of security devices. Security devices according to ISO/IEC 7816 series are well-known and are used in ICCs and also in electronic systems today.

Other types of security devices are defined by other specifications and standards. These security devices do not implement all functions of ISO/IEC 7816-4 but extend industry-specific functional APDUs in addition. Function sets are reduced, limited to specific functionality, e.g. for authentication, key storage and/or key application in combination with cryptographic functionality.

A well-known technique from software standards, e.g. JAVA®, uses language specific serialization of data structures for transmission of byte streams between the software modules. Formats for such mechanisms are, e.g. JSON or CBOR. Instead of a serialization into a specific command format, e.g. ISO/IEC 7816-4 defined APDUs, a transformation in the automated language related serialized format is sufficient. The result can be transferred directly to the security device. A de-serialization within the security device allows the re-build of the original data structure, which can be directly interpreted.

Annex A
(informative)

Design rules

Any definition of an API call in the ISO/IEC 23465 series of standards obeys the following design rules: Keep it small and simple:

For a client programmer it should be simple to use services and functions provided by a security device, especially for client programmers with little expertise in embedded devices.

— Avoid exceptions to normative statements: It should be avoided to restrict requirements containing shall or should.

— Simplicity over performance: Simplicity should be more important than performance.

— Single thread, synchronous security device: A security device should be treated as a single threaded device with synchronous communication. This implies that the communication model to a security device is half duplex.

— Multi thread Proxy: A Proxy should be able to communicate simultaneously with several clients. Because all security devices are treated as single threaded, it follows that the Proxy schedules procedure calls.

— Object-oriented programming: From a client's perspective, the API should provide (instantiations of) objects. Then, depending on the type of object, a client is able to call procedures defined for a particular object type in order to request a corresponding service or function from a security device.

— A class diagram should show the available type of objects and procedures supported by a class.

— For each procedure there should be an inverse operation, i.e. protagonist versus antagonist, e.g. get – set, read – write, create – delete, append – truncate.

— Procedures should be protected by access rules. Thus, during instantiation, it is possible to define which procedure is usable under which condition.

— Exception handling over exit status: Procedures should not be defined with an exit status (return code). Unexpected situations (e.g. errors) are handled via exceptions.

— Instantiations of objects in a security device should be organized in a hierarchical structure.

— It should be possible to select each instantiation of an object in a security device. By calling a select-procedure a client receives an instantiation of an object. Afterwards the client is able to use that instantiation to call procedures defined for that instantiation. Thus, the modus operandi from a client's perspective is to first select an object then work with it.

— Objects should provide getters and setters for their properties. These getters and setters are also procedures and thus protected access rules.

Annex B
(informative)

Use cases

B.1 General

From the perspective of a client application, this annex lists use cases supported by the API between client and Proxy. Thus, the main actor for all of the following use cases is the client application.

Use cases at level “very high summary” are contained in B.2. Use cases at level “user goal” are contained in B.3. Use cases at level “subfunction” are contained in B.4.

B.2 Use case level “very high summary”

B.2.1 Request services for identification, authentication and signing (IAS)

Primary actor: Application.

Scope: Services for IAS.

Level: Very high summary.

Antagonist: This use case is identical to its antagonist.

The application requests a cryptographic service for IAS. From the security device perspective several uses cases from level “user goal” are involved, e.g.:

a) open a connection to a specific security device, see B.3.2;

b) sign an artefact, see B.3.4;

c) decipher an artefact, see B.3.7;

d) verification of a signature.

B.3 Use case level “user goal”

B.3.1 Request a list of available security devices

Primary actor: Application.

Scope: Managing connections to security devices.

Level: User goal.

Antagonist: None.

The application retrieves a list of available security devices or (equivalent) a list of smart card readers with a security device inserted in order to select a security device and open a connection to that SD (see B.3.2).

B.3.2 Open a connection to a specific security device

Primary actor: Application.

Scope: Managing connections to security devices.

Level: User goal.

Antagonist: Close an open connection.

The application opens a connection to a specific security device in order to establish a communication channel to that security device and afterwards use it.

B.3.3 Close an open connection

Primary actor: Application.

Scope: Managing connections to security devices.

Level: User goal.

Antagonist: Open a connection to a specific security device.

The application closes an open connection in order to signal that it doesn't want to use this connection any longer.

B.3.4 Sign an artefact

Primary actor: Application.

Scope: Cryptographic operation.

Level: User goal.

Antagonist: Verify signature.

The application signs an artefact in order to prove its authenticity and integrity. From the security device perspective, several subfunctions are involved:

a) read certificate, see B.4.1;

b) authenticate user, see B.4.3;

c) sign artefact.

B.3.5 Verify a signature

Primary actor: Application.

Scope: Cryptographic operation.

Level: User goal.

Antagonist: Sign an artefact.

The application verifies the signature of an artefact in order to prove either its integrity or authenticity, or both. Depending on the implementation, it is possible that the security device is not involved in this use case, because typically none of the necessary information for verifying the signature (e.g. artefact, signature and public signature verification key) is stored inside a security device.

B.3.6 Encipher an artefact

Primary actor: Application.

Scope: Cryptographic operation.

Level: User goal.

Antagonist: Decipher an artefact.

The application enciphers an artefact in order to prevent disclosure of sensitive information. Depending on the implementation, it is possible that the security device is not involved in this use case, because typically none of the necessary information for enciphering (e.g. content of artefact and public key of the receiver) is stored inside a security device.

B.3.7 Decipher an artefact

Primary actor: Application.

Scope: Cryptographic operation.

Level: User goal.

Antagonist: Encipher an artefact.

The application deciphers an artefact in order to gain access to the plain text. From the security device perspective, several subfunctions are involved:

a) authenticate user, see B.4.3;

b) decipher a ciphertext, see B.4.10.

B.4 Use case level “subfunction”

B.4.1 Read arbitrary data

Primary actor: Application.

Scope: Data management.

Level: Subfunction.

Antagonist: Write arbitrary data.

The application reads data stored in a security device in order to retrieve information.

B.4.2 Write arbitrary data

Primary actor: Application.

Scope: Data management.

Level: Subfunction.

Antagonist: Read arbitrary data.

The application writes data to a security device in order to store information.

B.4.3 Verify user by password

Primary actor: Application.

Scope: Security management.

Level: Subfunction.

Antagonist: Clear user verification, Unblock password

The application presents a password to the security device in order to gain access to protected functionality. If the application presents a wrong password too often then this use case is blocked.

B.4.4 Clear user verification

Primary actor: Application.

Scope: Security management.

Level: Subfunction.

Antagonist: Verify user by password.

The application clears the security status of a password in order to protect functionality of a security device from unauthorized access.

B.4.5 Change password

Primary actor: Application.

Scope: Security management.

Level: Subfunction.

Antagonist: None.

The application sends a new value for a password in order to change it.

B.4.6 Unblock password

Primary actor: Application.

Scope: Security Management.

Level: Subfunction.

Antagonist: Verify user by password.

The application unblocks a password in order to enable the use case Verify user by password.

B.4.7 Generate asymmetric key pair

Primary actor: Application.

Scope: Cryptographic operation.

Level: Subfunction.

Antagonist: None.

The application triggers the generation of an asymmetric key pair in order to use the generated keys later.

B.4.8 Read public key

Primary actor: Application.

Scope: Cryptographic operation.

Level: Subfunction.

Antagonist: None.

The application reads a public key in order to use it, e.g.:

a) create a corresponding certificate;