INTERNATIONAL STANDARD

ISO/IEC 24727-3

First edition
2008-12-01

Identification cards — Integrated circuit card programming interfaces —

Part 3: Application interface

Cartes d'identification — Interfaces programmables de cartes à puce —

Partie 3: Interface d'application

Reference number
ISO/IEC 24727-3:2008(E)

ISO copyright office
Case postale 56 • CH-1211 Geneva 20
Tel. + 41 22 749 01 11
Fax + 41 22 749 09 47
E-mail copyright@iso.org
Web www.iso.org

Published in Switzerland

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. In the field of information technology, ISO and IEC have established a joint technical committee, ISO/IEC JTC 1.

International Standards are drafted in accordance with the rules given in the ISO/IEC Directives, Part 2.

The main task of the joint technical committee is to prepare International Standards. Draft International Standards adopted by the joint technical committee are circulated to national bodies for voting. Publication as an International Standard requires approval by at least 75 % of the national bodies casting a vote.

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.

ISO/IEC 24727-3 was prepared by Joint Technical Committee ISO/IEC JTC 1, *Information technology*, Subcommittee SC 17, *Cards and personal identification*.

ISO/IEC 24727 consists of the following parts, under the general title *Identification cards — Integrated circuit card programming interfaces*:

— *Part 1: Architecture*

— *Part 2: Generic card interface*

— *Part 3: Application interface*

— *Part 4: Application programming interface (API) administration*

— *Part 5: Testing*

— *Part 6: Registration authority procedures for the authentication protocols for interoperability*

Introduction

ISO/IEC 24727 is a set of programming interfaces for interactions between integrated circuit cards (ICCs) and external applications to include generic services for multi-sector use. The organization and the operation of the ICC conform to ISO/IEC 7816-4.

ISO/IEC 24727 is relevant to ICC applications desiring interoperability among diverse application domains. This part of ISO/IEC 24727 specifies a language-independent and implementation-independent application level interface that allows information and transaction interchange with a card. ISO/IEC 7498-1 is used as the layered architecture of the application interface. That is, the application interface assumes that there is a protocol stack through which it will exchange information and transactions among cards using commands conveyed through the message structures defined in ISO/IEC 7816. The semantics of commands accessed by the application interface refers to application protocol data units (APDUs) as characterized in ISO/IEC 24727-2, and in the following standards:

— ISO/IEC 7816-4, *Identification cards — Integrated circuit cards — Part 4: Organization, security and commands for interchange*

— ISO/IEC 7816-8, *Identification cards — Integrated circuit cards — Part 8: Commands for security operations*

— ISO/IEC 7816-9, *Identification cards — Integrated circuit cards — Part 9: Commands for card management*

The goal of this part of ISO/IEC 24727 is to maximize the applicability and solution space of software tools that provide application interface support to card-aware applications. This effort includes supporting the evolution of card systems as the cards become more powerful, peer-level partners with existing and future applications while minimizing the impact to existing solutions conforming to this part of ISO/IEC 24727.

Identification cards — Integrated circuit card programming interfaces —

Part 3: Application interface

1 Scope

This part of ISO/IEC 24727 defines services as representations of action requests and action responses to be supported at the client-application service interface. The services are described in a programming-language-independent way.

This part of ISO/IEC 24727 is the application interface of the Open Systems Interconnection Reference Model defined in ISO/IEC 7498-1. It provides a high-level interface for a client-application making use of information storage and processing operations of a card-application as viewed on the generic card interface.

This part of ISO/IEC 24727 does not mandate a specific implementation methodology for this interface.

2 Normative references

The following referenced documents are indispensable for the application of this document. For dated references, only the edition cited applies. For undated references, the latest edition of the referenced document (including any amendments) applies.

ISO/IEC 7816-11, *Identification cards — Integrated circuit cards — Part 11: Personal verification through biometric methods*

ISO/IEC 24727-1, *Identification cards — Integrated circuit card programming interfaces — Part 1: Architecture*

ISO/IEC 24727-2, *Identification cards — Integrated circuit card programming interfaces — Part 2: Generic card interface*

IETF RFC 2141, *URN Syntax,* May 1997

3 Terms and definitions

For the purposes of this document, the terms and definitions given in ISO/IEC 24727-1, ISO/IEC 24727-2 and the following apply.

3.1
access control list
set of access rules

3.2
access permission
granted capability to perform an action

3.3
access rule
association of an action and a security condition in the context of a card-application

3.4
action
operation that a client-application can request of a card-application at the ISO/IEC 24727-3 application interface

3.5
alpha card-application
administrative card-application, specific to ISO/IEC 24727, providing card and application discoverability and administrative services

3.6
card-application-path
ordered set of protocol termination points in the network that connects the client-application to the card-application

3.7
card-application-service
set of actions

3.8
channel
physical pathway allowing movement of bits of information between a client-application and a card-application

3.9
client-application
software component running on a platform that uses data storage and computational services offered by a card-application

3.10
connection
logically referenced channel

3.11
global differential-identity
differential-identity recognized in all card-applications that are managed by the same alpha card-application

3.12
local differential-identity
differential-identity recognized only within a specific card-application in which it is defined

3.13
parameter
information required to define or effect an action

3.14
return code
information, including status, returned as a consequence of an action

3.15
security condition
boolean expression in terms of differential-identity authentication states

3.16
session
connection used under a particular security context

3.17
target
persistent entity that shall be manipulated through the actions of a card-application-service

4 Abbreviated terms

For the purposes of this document, the abbreviated terms given in ISO/IEC 24727-1 and the following apply.

ACL	access control list
AR	access rule
DID	differential-identity
DSI	data structure for interoperability

5 Organization for interoperability

5.1 General

A client-application and a card-application comprise two peer-level entities that interact through well-defined transaction and communication mechanisms. The ISO/IEC 24727-3 application interface shall be the sole mechanism through which a client-application interacts with a card-application.

This clause defines the model of computation and persistent entities on the ISO/IEC 24727-3 application interface by which the client-application and the card-application interact. A security model governs the security mechanisms through which trust in this interaction is achieved.

Clause 5.2 introduces the conceptual entities on the ISO/IEC 24727-3 application interface and describes their operational behavior and relationships. Clause 5.3 specifies the entities and relationships presented on the ISO/IEC 24727-3 application interface. Clause 5.4 specifies the security model provided on the ISO/IEC 24727-3 application interface.

5.2 Computation model

Using the terminology defined in ISO/IEC 24727-1:

5.2.1 A *client-application* may know a *card-application-path* to a *card-application.* Using this card-application-path, a client-application can initiate a *connection* to a card-application. The card-application shall be known as the *current card-application* for the connection.

5.2.2 The ISO/IEC 24727-3 *application interface* is the set of *card-application-services* provided to the client-application. Each card-application-service shall be comprised of *actions* that may be *requested* by a client-application and *confirmations* returned by the SAL.

5.2.3 If the client-application does not know a card-application-path to a card-application, it may make a request at the ISO/IEC 24727-3 application interface to discover a card-application-path to the requested card-application.

5.2.4 A card-application shall be uniquely identified by an AID.

5.2.5 The *alpha card-application* provides the basis for trusted management of card-applications by the client-application.

5.2.6 A client-application may open more than one connection. A client-application may open more than one connection with the same card-application, each connection being referenced by a different *connection handle*.

5.2.7 Using an open connection, a client-application may initiate a *session* with a card-application.

5.2.8 A client-application may open more than one session. Only one session may be open within a connection.

5.2.9 The *current state* is the current state of a connection defined by the current card-application, current data-set, authentication state of recognized differential-identities, and the presence of a session on the connection.

5.2.10 A card-application contains one or more card-application-services.

5.2.11 A specific card-application-service, the *Named Data Service*, provides access to zero or more *data-sets*.

5.2.12 A data-set contains zero or more *data structures for interoperability* (DSI). A data-set shall provide naming scope and access rules for the DSIs it contains.

5.2.13 Every data-set shall be accessible according to the *access rules* governing the actions available from the *Named Data Service*.

5.2.14 The currently selected data-set in a card-application shall be known as the *current data-set*.

5.2.15 A single *action request* at the ISO/IEC 24727-3 application interface may translate into multiple generic requests at the ISO/IEC 24727-2 generic card interface.

5.2.16 A card-application may support multiple actions that may be requested by the client-application through the ISO/IEC 24727-3 application interface.

5.2.17 An action request shall produce an *action confirmation*.

5.2.18 An *access rule* consists of the name of an action and a *security condition*. The security condition is said to be associated with the action by the access rule.

5.2.19 An *access control list* is a collection of zero or more access rules. An access control list shall be associated with each *target*.

5.2.20 An action involving a target may be successfully performed if and only if the security condition associated with the action by an access rule in the access control list of the target evaluates to TRUE.

5.2.21 An *authentication protocol* is the process by which a *differential-identity* demonstrates possession of a *marker*.

5.2.22 *Authentication* is the successful execution of an authentication protocol. In this case the differential-identity is said to be authenticated.

5.2.23 The *authentication state* of a differential-identity shall be a boolean variable that is TRUE if the differential-identity is authenticated and FALSE otherwise.

5.2.24 A client-application may learn about information, status or services provided by a card-application through a discovery process effected through the various List, Get, and Describe actions offered at the 24727-3 application interface.

5.2.25 Access rules are discoverable at the ISO/IEC 24727-3 application interface.

5.2.26 In general, the degree of interoperability of a card-application with client-applications depends on the discoverable information presented by the card-application at the ISO/IEC 24727-3 application interface.

5.3 Entity relationships on the application interface

5.3.1 General

This clause describes the entities accessible through a card-application, specifically the alpha card-application and card-applications it manages. Entities more directly involved in the security model are described in Clause 5.4.

Figure 1 illustrates the client-application to card-application paradigm that underlies the ISO/IEC 24727 standard. The ISO/IEC 24727-3 application interface facilitates the connection and session mechanisms shown in this figure.

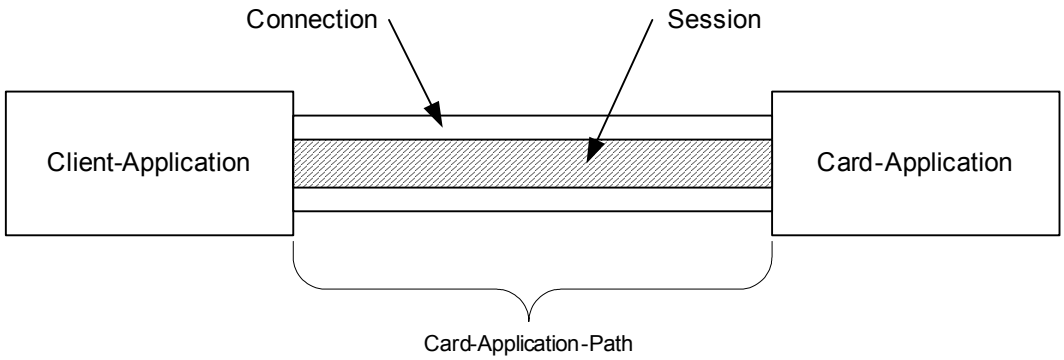

Figure 1 — Application Connection Paradigm

A card-application-path can be specified to allow a client-application to be connected to a specific card-application. Through such a connection, the client-application can access card-application-services using the ISO/IEC 24727-3 application interface. A session adds a security context to a connection in addition to its intrinsic security characteristics that can be used to protect the data flowing between the client-application and the card-application.

Figure 2 illustrates the entities defined in the model of computation and establishes the relationships among these entities. This entity-relationship diagram is elaborated below to describe individual card-application-services. The type of target applicable to each action is shown in the rounded boxes associated with each action.

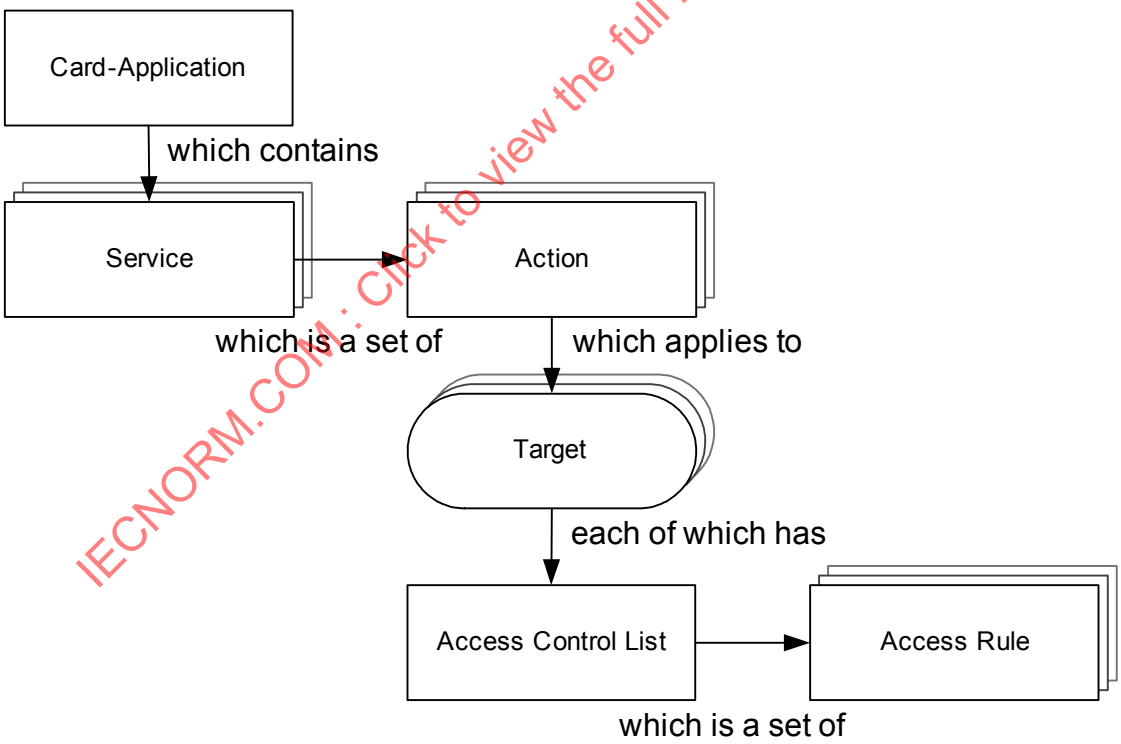

Figure 2 — Model of Computation Entities and Relationships

A card-application is a collection of targets and services. A service is a collection of actions. An action is an operation which is applied to a target. The actions and services of a card-application are accessed by a client-application using the ISO/IEC 24727-3 application interface.

5.3.2 Alpha Card-Application

The alpha-card-application shall be available at the application interface. It may either be present in the ICC or emulated by the SAL or GCAL implementation. In all cases, a connection by the client application to the alpha-card-application grants the availability of the card application list.

5.3.3 Accessing Card-Application-Services

A client-application uses the Initialize and CardApplicationPath entry points on the ISO/IEC 24727-3 application interface to gain access to card-application-services. A client-application uses the Terminate entry point to terminate access to card-application-services.

Before accessing card-application-services, a client-application must request the Initialize entry point on the ISO/IEC 24727-3 application interface. After the Initialize confirmation, the client-application may open connections and request actions with multiple card-applications, simultaneously or sequentially. The client-application should request the Terminate entry point when use of card-application-services is no longer needed.

5.3.4 Connection Service Interface

This card-application-service provides actions for the establishment of a connection between a client-application and a card-application. Once a connection is established, a session may be effected through this connection in order to enhance the security characteristics of the communication between the client-application and the card-application. Figure 3 illustrates the entity relationships of the Connection Service.

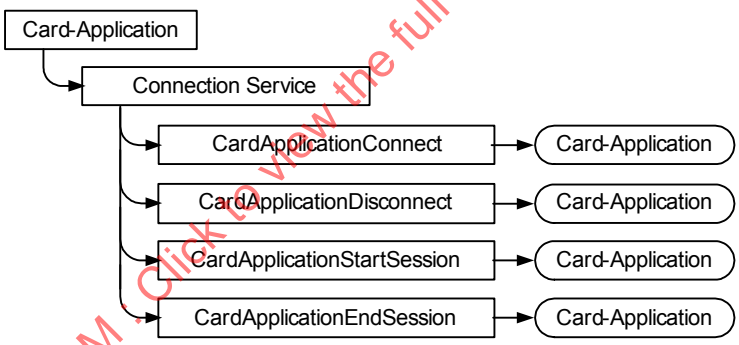

Figure 3 — Connection Service

The target of each of these actions, as specified in the diagram above, is the current card-application, or the card-application to which the client-application is attempting to connect.

5.3.5 Card-Application Service Interface

This card-application-service provides actions for the creation and manipulation of card-applications. Figure 4 illustrates the entity relationships of the Card-Application Service.

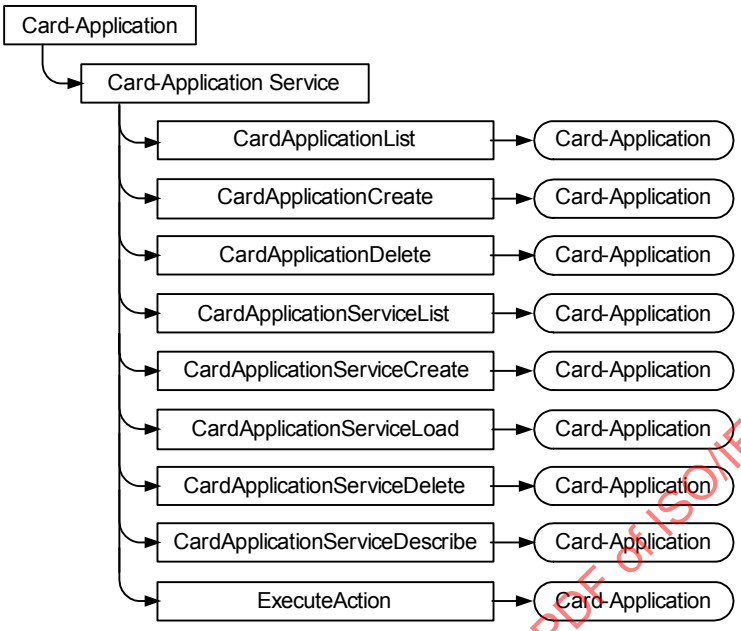

Figure 4 — Card-Application Service

The target of the CardApplicationCreate and CardApplicationDelete actions is the alpha card-application. The current card-application shall be the alpha card-application when requesting these actions.

The target of each of the remaining actions is the current card-application.

5.3.6 Named Data Service Interface

This card-application-service provides actions for the creation and manipulation of data-sets, a containment mechanism that allows for the establishment of common access rules for data contained in data structures for interoperability. Data-sets may contain any number of DSIs. Figure 5 illustrates the entity relationships of the Named Data Service.

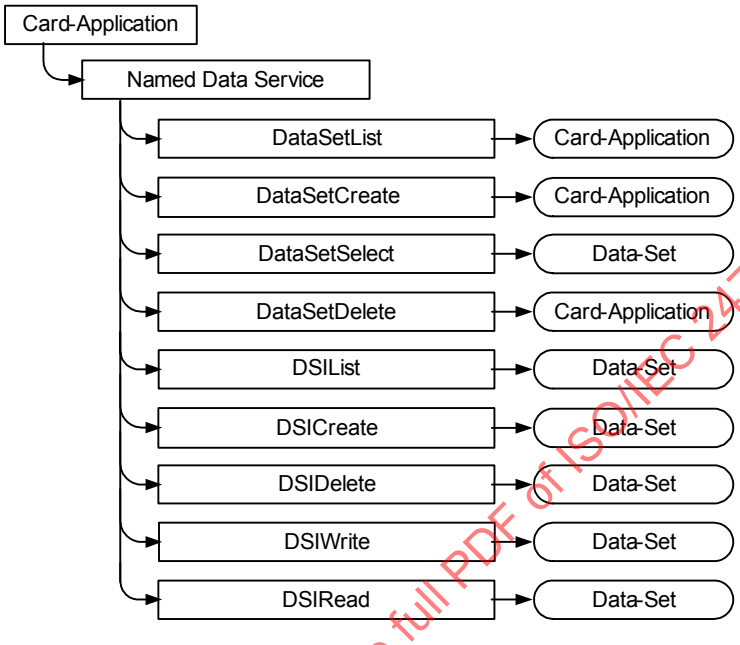

Figure 5 — Named Data Service

The target of each of these actions, as specified in the diagram above, is the current card-application, the current data-set, or the data-set which the client-application is attempting to select.

5.3.7 Cryptographic Service Interface

This card-application-service provides actions for a variety of cryptographic operations to be performed on or through the parameters contained within a differential-identity. Figure 6 illustrates the entity relationships of the Cryptographic Service.

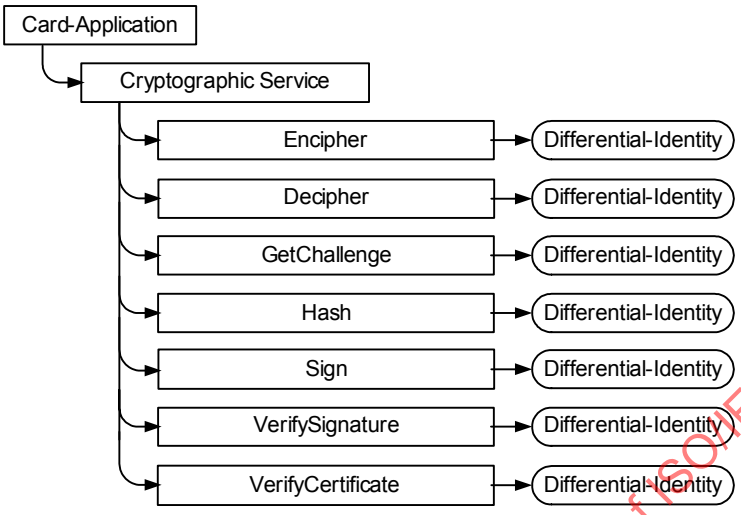

Figure 6 — Cryptographic Service

The target of each of these actions, as specified in the diagram above, is the differential-identity named in the request of the action.

5.3.8 Differential-Identity Service Interface

This card-application-service provides actions for the creation and manipulation of differential-identities. Figure 7 illustrates the entity relationships of the Differential-Identity Service.

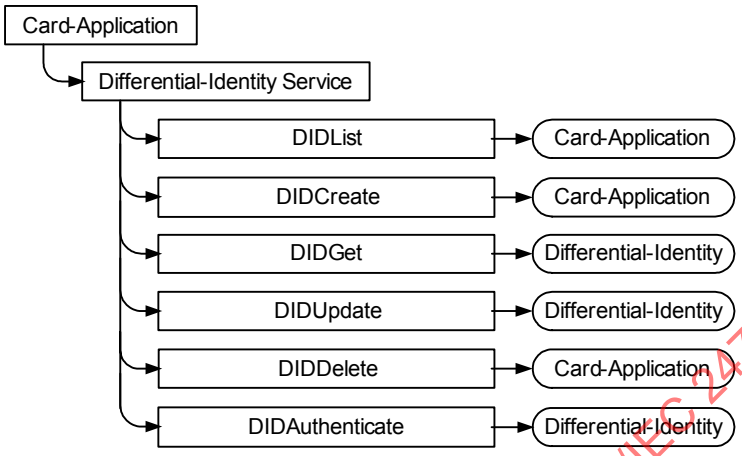

Figure 7 — Differential-Identity Service

The target of each of these actions, as specified in the diagram above, is either the current card-application or the differential-identity named in the request of the action.

5.3.9 Authorization Service Interface

This card-application-service provides actions for the manipulation of access control lists. Figure 8 illustrates the entity relationships of the Authorization Service.

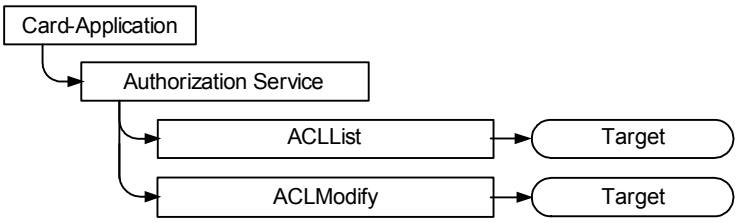

Figure 8 — Authorization Service

The target of these actions, as specified in the diagram above, can be any target. The target is specified in the request of the action.

5.4 Security model

5.4.1 General

A shared security state is established between a client-application and a card-application by successfully performing the required protocols between the client-application and the card-application. Establishing the required security state authorizes the performance of actions requested of a card-application by a client-application.

The mechanism for specifying the required security state for performing an action shall be the access rule. A security state is established by authenticating differential-identities.

5.4.2 Differential-Identity

In order to establish a security state encompassing a client-application and a card-application, the differential-identity authentication mechanism shall be used. The elements of each differential-identity are illustrated in Table 1. At the generic card interface, a differential-identity may be transmitted as defined by the ASN.1 in Annex C.

Table 1 — Elements of Differential-Identity

Element 1	Element 2	Element 3	Element 4	Element 5
DIDName	Authentication Protocol	Marker	Scope	Qualifier
Mandatory	Mandatory	Mandatory	Implicit	Optional

The DIDName element shall be specified by the client-application which creates the differential-identity at the time the differential-identity is created.

The Authentication Protocol element shall be specified as an object identifier (OID) and determines for which actions the differential-identity can be used. Only if the authentication protocol specified by this OID is defined for use in authentication as defined in Annex A shall its successful completion for the differential-identity set the authentication state of the differential-identity to TRUE within the card-application. A DIDName shall be associated with only one authentication protocol. The actions of the Cryptographic Service may make use of the information in the Marker element as defined in Annex A.

Note: The Authentication Protocol indicated by element 2 is used either for authentication or for other cryptographic services.

The Scope element is implicit. Differential-identities defined within the alpha card-application are global in scope and therefore are recognized within all card-applications managed by this alpha card-application. All other differential-identities are local in scope to the card-application in which they are defined. The scope of the corresponding authentication state shall be identical to the scope of the differential-identity.

The optional Qualifier element may provide details on the use of the differential-identity. For example, it may refer to an external specification by OID.

The Marker element is the information within the card-application, or a reference to it, that shall be used in executing the authentication protocol specified in the Authentication Protocol element.

Examples of markers include:

- a PIN
- a password

- a symmetric key
- an asymmetric key
- a digital certificate
- a biometric image or template
- a pair of symmetric keys; e.g., one for encryption and one for message authentication code (MAC) generation

As evidenced by the final entry in the previous list, the marker may be comprised of multiple parts; e.g., of multiple keys. This allows for complex authentication protocols that may need to use multiple keys in parallel or serially.

The Marker may often be implemented within a card-application as a key that is referred to by a key reference as specified in ISO/IEC 7816-4. One differential-identity and hence its marker may be related to multiple key references in some complex authentication protocols. The key reference may be coded as defined in ISO/IEC 7816-4 security environments. A specific key reference may appear in several differential-identities.

The association of the DIDName to a Marker and Authentication Protocol is the mechanism through which the differential-identity is associated with the marker. This mapping shall be maintained as part of the discovery information. This mapping allows the client-application and the card-application to recognize, exchange, and share information about a differential-identity.

5.4.3 Authentication Protocols

Authentication protocols shall be the mechanism by which the client-application sets an authentication state within the card-application for the differential-identity indicated by the DIDName. Authentication protocols are enumerated in Table 2, and described in Annex A.

Table 2 — Authentication Protocols

Simple Assertion	Modular Extended Access Control Protocol (M-EAC)
Asymmetric Internal Authenticate	Key Transport with mutual authentication based on RSA
Asymmetric External Authenticate	Age Attainment
Symmetric Internal Authenticate	Asymmetric Session Key Establishment
Symmetric External Authenticate	Secure PIN Compare
Compare	EC Key Agreement with Card-Application Authentication
PIN Compare	EC Key Agreement with Mutual Authentication
Biometric Compare	Simple EC-DH Key Agreement
Mutual Authentication with Key Establishment	GP Asymmetric Authentication
Client-Application Mutual Authentication with Key Establishment	GP Symmetric Authentication (Explicit Mode)
Client-Application Asymmetric External Authenticate	GP Symmetric Authentication (Implicit Mode)

To set the authentication state of a differential-identity to TRUE, the authentication protocol of that differential-identity shall be successfully completed using either the CardApplicationStartSession or DIDAuthenticate action.

Consecutive requests of the same action for the same connection handle and named differential-identity may be necessary to complete the authentication protocol as defined in Annex A. These represent individual steps in a multistep authentication protocol.

Differential-identity authentication states are valid for the connection handle specified during their authentication and shall be set to FALSE when the CardApplicationDisconnect action is requested for that connection handle or the connection handle otherwise becomes invalid. Authentication states established while connected to the alpha card-application shall be valid for all connections to card-applications managed by this alpha card-application. If a differential-identity was authenticated by requesting the CardApplicationStartSession action, a subsequent request of the CardApplicationEndSession action shall set its authentication state to FALSE.

The authentication state of a differential-identity shall be set to FALSE when the DIDAuthenticate or CardApplicationStartSession action is requested and set to TRUE only at the time of the action confirmation produced for the request that successfully completes the authentication protocol.

The authentication state of a differential-identity that is not recognized at the time its authentication state is evaluated shall evaluate to FALSE.

5.4.4 Session Keys

A protocol may involve the creation of cryptographic keys for use during a session.

The session keys shall become invalid when the CardApplicationDisconnect action is requested or the connection otherwise becomes invalid. Additionally, if the protocol was completed by requesting the CardApplicationStartSession action, the session keys shall become invalid when the CardApplicationEndSession action is requested.

5.4.5 Access Control Lists

An access control list (ACL) is a set of access rules (ARs) applicable to a target. An AR associates an action of a card-application-service with a security condition. A security condition is a boolean expression in terms of differential-identity authentication states. A target is a data-set, a differential-identity, or a card-application.

Authenticating a differential-identity shall set its authentication state to TRUE. Otherwise, its authentication state shall be FALSE. If the security condition evaluates to TRUE, the associated action is authorized. If the security condition evaluates to FALSE the associated action is not authorized.

An action involving a target shall only be performed if an AR exists in the ACL applicable to that target associating the action with a security condition that evaluates to TRUE with respect to the current authentication states of the differential-identities recognized within the current card-application. An AR is only applicable to requests of its action and only when the request involves the target to which its ACL is associated.

An ACL is a part of the target to which it applies and is not a distinct target itself. Therefore when an action of the Authorization Service appears in an AR, the AR controls whether that action may be performed on the ACL in which it exists. The ARs controlling access to an ACL exist within the ACL itself.

Through the inclusion of a differential-identity in a security condition, the authentication of that differential-identity may become necessary to authorize the action associated with it in an access rule. If the differential-identity is associated with an authentication protocol that generates session keys, its inclusion in the security condition may make necessary the establishment of a security context.

6 Card-application-service access

6.1 General

This clause defines entry points on the ISO/IEC 24727-3 application interface through which a client-application can locate a specific card-application and establish communication channels to that card-application. A client-application may connect to more than one card-application.

6.2 Initialize

6.2.1 Purpose

This entry point shall initialize the ISO/IEC 24727-3 layer including connectivity to other components of ISO/IEC 24727 protocol stack. Card-application-services shall be available on the ISO/IEC 24727-3 application interface to the client-application only after a successful call to this entry point.

The initialization state (variables, possible values, resources, supporting card-application-services) of the ISO/IEC 24727-3 application interface implementation is out of scope for ISO/IEC 24727-3.

6.2.2 Entry point

```
OUT       ReturnCode     Initialize(
                         );
```

6.2.3 Parameters

None

6.2.4 Prerequisites

None

6.2.5 Return codes

```
API_OK

API_COMMUNICATION_FAILURE
API_INCORRECT_PARAMETER
```

6.2.6 Impact on current state

Upon successful completion, the ISO/IEC 24727-3 layer shall be initialized.

6.3 Terminate

6.3.1 Purpose

This entry point shall terminate access to card-application-services offered on the ISO/IEC 24727-3 application interface. Any connection(s) currently in effect between the client-application and any card-application(s) shall be disconnected. The Initialize entry point shall be called before any card-application-services are available on the ISO/IEC 24727-3 application interface to the client-application.

6.3.2 Entry point

```
OUT         ReturnCode      Terminate(
                            );
```

6.3.3 Parameters

None

6.3.4 Prerequisites

None

6.3.5 Return codes

```
API_OK
API_WARNING_CONNECTION_DISCONNECTED
API_INCORRECT_PARAMETER

API_NOT_INITIALIZED
API_COMMUNICATION_FAILURE
```

6.3.6 Impact on current state

The return code of API_WARNING_CONNECTION_DISCONNECTED is a successful completion of the request with the side effect of disconnecting at least one active connection.

The return code of API_COMMUNICATION_FAILURE is an unsuccessful completion of the action. In this case, the state of existing connections is undefined.

6.4 CardApplicationPath

6.4.1 Purpose

This entry point shall determine card-application-paths from the client-application to a named card-application.

6.4.2 Entry point

```
OUT       ReturnCode                CardApplicationPath(
IN        CardApplicationPath               cardAppPathRequest,
OUT       CardApplicationPathSet            cardAppPathResultSet
                                    );
```

6.4.3 Parameters

cardAppPathRequest sequence of protocol termination points representing the final segment of a card-application-path from the client-application to the card-application, including its AID, a host or terminal identifier and optionally an IFD name or identifier. The specific syntax of this final segment is expressed by IETF RFC 2141.

cardAppPathResultSet set of card-application-paths between the client-application and the named card-application, each of which contains the value of **cardAppPathRequest** as its final segment.

6.4.4 Prerequisites

None

6.4.5 Return codes

```
API_OK

API_INCORRECT_PARAMETER

API_NOT_INITIALIZED
API_SECURITY_CONDITION_NOT_SATISFIED
API_TOO_MANY_RESULTS
API_COMMUNICATION_FAILURE
```

6.4.6 Impact on current state

None

7 Connection service

7.1 General

This clause defines actions through which a client-application can connect to or disconnect from a card-application and start or end a session based on a connection.

7.2 CardApplicationConnect

7.2.1 Purpose

This action shall establish an unauthenticated connection between the client-application and a card-application.

7.2.2 Action

```
OUT         ReturnCode              CardApplicationConnect(
IN          CardApplicationPath           cardApplicationPath,
IN          BOOLEAN                       exclusiveUse,
OUT         ConnectionHandle              connectionHandle
                                    );
```

7.2.3 Parameters

cardApplicationPath card-application-path from the client-application to the card-application

exclusiveUse if TRUE, this indicates to the ISO/IEC 24727-3 layer that the connection should be opened if no other connection currently exists with the the card-application and no other connection should be established to the card-application until this connection is closed; if FALSE, multiple simultaneous connections may be opened to this card-application

connectionHandle opaque reference for use at the ISO/IEC 24727-3 application interface in action requests

7.2.4 Prerequisites

None

7.2.5 Return codes

```
API_OK

API_INCORRECT_PARAMETER

API_NOT_INITIALIZED
API_EXCLUSIVE_NOT_AVAILABLE
API_SECURITY_CONDITION_NOT_SATISFIED
API_COMMUNICATION_FAILURE
```

7.2.6 Impact on current state

Upon successful completion, the card-application named in the card-application-path is the current card-application and the card-application associated with the connection handle.

7.3 CardApplicationDisconnect

7.3.1 Purpose

This action shall terminate a connection between the client-application and a card-application.

7.3.2 Action

```
OUT         ReturnCode          CardApplicationDisconnect(
IN          ConnectionHandle                connectionHandle,
IN          ReaderAction                    action
                                );
```

7.3.3 Parameters

connectionHandle connection handle

action optional argument indicating a requested IFD action to reset or unpower the IFD or eject or confiscate the ICC

7.3.4 Prerequisites

A valid `connectionHandle` is expected.

7.3.5 Return codes

```
API_OK
API_WARNING_SESSION_ENDED

API_INCORRECT_PARAMETER

API_NOT_INITIALIZED
API_SECURITY_CONDITION_NOT_SATISFIED
API_COMMUNICATION_FAILURE
```

7.3.6 Impact on current state

The return code of API_WARNING_SESSION_ENDED is a successful completion of the action with the side effect of ending at least one active session on the connection.

7.4 CardApplicationStartSession

7.4.1 Purpose

This action shall establish a session between a client-application and a card-application.

A connection handle to a second card-application, e.g. a security access module (SAM), may be provided to perform cryptographic operations on behalf of the client-application in establishing the session.

Consecutive requests of this action for the same connection handle and named differential-identity may be necessary to complete the protocol as defined in Annex A.

7.4.2 Action

```
OUT       ReturnCode                CardApplicationStartSession(
IN        ConnectionHandle                connectionHandle,
IN        DIDScope                        didScope,
IN        DIDName                         didName,
IN/OUT    DIDAuthenticationData           authenticationProtocolData,
IN        ConnectionHandle                samConnectionHandle
                                    );
```

7.4.3 Parameters

connectionHandle	connection handle
didScope	scope of the named differential-identity
didName	name of the differential-identity containing the authentication protocol to be executed to establish the session between the client-application and the card-application
authenticationProtocolData	data exchanged according to the protocol
samConnectionHandle	optional argument to a card-application that can be used by the ISO/IEC 24727-3 layer

7.4.4 Prerequisites

A valid `connectionHandle` is expected.

7.4.5 Return codes

```
API_OK
API_NEXT_REQUEST

API_INCORRECT_PARAMETER
API_NAMED_ENTITY_NOT_FOUND
API_PROTOCOL_NOT_RECOGNIZED
API_INAPPROPRIATE_PROTOCOL_FOR_ACTION

API_NOT_INITIALIZED
API_SECURITY_CONDITION_NOT_SATISFIED
API_INSUFFICIENT_RESOURCES
API_COMMUNICATION_FAILURE
```

7.4.6 Impact on current state

See Annex A.

7.5 CardApplicationEndSession

7.5.1 Purpose

This action shall end a session between a client-application and a card-application.

7.5.2 Action

```
OUT       ReturnCode          CardApplicationEndSession(
IN        ConnectionHandle          connectionHandle
                              );
```

7.5.3 Parameters

connectionHandle connection handle

7.5.4 Prerequisites

A valid `connectionHandle` is expected.

7.5.5 Return codes

`API_OK`

`API_INCORRECT_PARAMETER`
`API_NO_ACTIVE_SESSION`

`API_NOT_INITIALIZED`
`API_SECURITY_CONDITION_NOT_SATISFIED`
`API_COMMUNICATION_FAILURE`

7.5.6 Impact on current state

Upon successful completion, the security characteristics of the connection revert to the intrinsic security characteristics of the connection.

8 Card-application service

8.1 General

This clause defines actions through which card-applications can be created and deleted and through which services can be created and deleted within card-applications. The CardApplicationCreate() action provides administrative functionality to establish the existence of the card-application. Card-application executable code is loaded via the CardApplicationServiceLoad() action.

8.2 CardApplicationList

8.2.1 Purpose

This action shall return the names of the card-applications listed in the alpha card-application. The access rule applicable to this action shall be in the access control list of the current card-application.

8.2.2 Action

```
OUT       ReturnCode                 CardApplicationList(
IN        ConnectionHandle                  connectionHandle,
OUT       CardApplicationNameList           cardApplicationNameList
                                     );
```

8.2.3 Parameters

connectionHandle connection handle

cardApplicationNameList list of card-application names

8.2.4 Prerequisites

A valid `connectionHandle` is expected.

8.2.5 Return codes

`API_OK`

`API_INCORRECT_PARAMETER`

`API_NOT_INITIALIZED`
`API_SECURITY_CONDITION_NOT_SATISFIED`
`API_COMMUNICATION_FAILURE`

8.2.6 Impact on current state

None

8.3 CardApplicationCreate

8.3.1 Purpose

This action shall create a new card-application.

8.3.2 Action

```
OUT       ReturnCode              CardApplicationCreate(
IN        ConnectionHandle                  connectionHandle,
IN        CardApplicationName               cardApplicationName,
IN        AccessControlList                 cardApplicationACL
                                  );
```

8.3.3 Parameters

connectionHandle connection handle

cardApplicationName name of the card-application to be created

cardApplicationACL access control list for this new card-application

8.3.4 Prerequisites

A valid `connectionHandle` is expected. The current card-application shall be the alpha card-application.

8.3.5 Return codes

```
API_OK

API_INCORRECT_PARAMETER
API_NAME_EXISTS

API_NOT_INITIALIZED
API_SECURITY_CONDITION_NOT_SATISFIED
API_PREREQUISITE_NOT_SATISFIED
API_INSUFFICIENT_RESOURCES
API_COMMUNICATION_FAILURE
```

8.3.6 Impact on current state

None

8.4 CardApplicationDelete

8.4.1 Purpose

This action shall delete the named card-application including all of its services, data-sets, and differential-identities.

8.4.2 Action

```
OUT       ReturnCode              CardApplicationDelete(
IN        ConnectionHandle                  connectionHandle,
IN        CardApplicationName               cardApplicationName
                                  );
```

8.4.3 Parameters

connectionHandle connection handle

cardApplicationName name of the card-application to be deleted

8.4.4 Prerequisites

A valid `connectionHandle` is expected. The current card-application shall be the alpha card-application.

8.4.5 Return codes

```
API_OK
API_WARNING_CONNECTION_DISCONNECTED

API_INCORRECT_PARAMETER
API_NAMED_ENTITY_NOT_FOUND

API_NOT_INITIALIZED
API_SECURITY_CONDITION_NOT_SATISFIED
API_PREREQUISITE_NOT_SATISFIED
API_COMMUNICATION_FAILURE
```

8.4.6 Impact on current state

Upon successful completion, any connections to the named card-application shall be disconnected.

8.5 CardApplicationServiceList

8.5.1 Purpose

This action shall list the card-application-services in the current card-application.

8.5.2 Action

```
OUT     ReturnCode                          CardApplicationServiceList(
IN      ConnectionHandle                        connectionHandle,
OUT     CardApplicationServiceNameList          cardApplicationServiceNameList
                                            );
```

8.5.3 Parameters

connectionHandle	connection handle
cardApplicationServiceNameList	list of names of card-application-services in the current card-application

8.5.4 Prerequisites

A valid `connectionHandle` is expected.

8.5.5 Return codes

`API_OK`

`API_INCORRECT_PARAMETER`

`API_NOT_INITIALIZED`
`API_SECURITY_CONDITION_NOT_SATISFIED`
`API_COMMUNICATION_FAILURE`

8.5.6 Impact on current state

None

8.6 CardApplicationServiceCreate

8.6.1 Purpose

This action shall create a new card-application-service in the current card-application.

8.6.2 Action

```
OUT     ReturnCode                  CardApplicationServiceCreate(
IN      ConnectionHandle                    connectionHandle,
IN      CardApplicationServiceName          cardApplicationServiceName
                                    );
```

8.6.3 Parameters

connectionHandle connection handle

cardApplicationServiceName name of the card-application-service to be created

8.6.4 Prerequisites

A valid `connectionHandle` is expected.

8.6.5 Return codes

```
API_OK

API_INCORRECT_PARAMETER
API_NAME_EXISTS

API_NOT_INITIALIZED
API_SECURITY_CONDITION_NOT_SATISFIED
API_INSUFFICIENT_RESOURCES
API_COMMUNICATION_FAILURE
```

8.6.6 Impact on current state

None

8.7 CardApplicationServiceLoad

8.7.1 Purpose

This action shall load executable code that implements a card-application-service into the current card-application.

8.7.2 Action

```
OUT       ReturnCode                          CardApplicationServiceLoad(
IN        ConnectionHandle                        connectionHandle,
IN        CardApplicationServiceName              cardApplicationServiceName,
IN        CardApplicationServiceLoadPackage       code
                                              );
```

8.7.3 Parameters

connectionHandle connection handle

cardApplicationServiceName name of the card-application-service implemented by the code

code service load package

8.7.4 Prerequisites

A valid `connectionHandle` is expected.

8.7.5 Return codes

```
API_OK

API_INCORRECT_PARAMETER
API_NAMED_ENTITY_NOT_FOUND

API_NOT_INITIALIZED
API_SECURITY_CONDITION_NOT_SATISFIED
API_INSUFFICIENT_RESOURCES
API_COMMUNICATION_FAILURE
```

8.7.6 Impact on current state

None

8.8 CardApplicationServiceDelete

8.8.1 Purpose

This action shall delete the named card-application-service, including the code that implements it, from the current card-application.

8.8.2 Action

```
OUT     ReturnCode                  CardApplicationServiceDelete(
IN      ConnectionHandle                    connectionHandle,
IN      CardApplicationServiceName          cardApplicationServiceName
                                    );
```

8.8.3 Parameters

connectionHandle connection handle

cardApplicationServiceName name of the card-application-service to be deleted

8.8.4 Prerequisites

A valid `connectionHandle` is expected.

8.8.5 Return codes

```
API_OK

API_INCORRECT_PARAMETER
API_NAMED_ENTITY_NOT_FOUND

API_NOT_INITIALIZED
API_SECURITY_CONDITION_NOT_SATISFIED
API_COMMUNICATION_FAILURE
```

8.8.6 Impact on current state

None

8.9 CardApplicationServiceDescribe

8.9.1 Purpose

This action shall return a URL or full description of the named card-application-service. The resulting description allows the client-application to discover functionality beyond the standardized set of card-application-services described in ISO/IEC 24727-3.

8.9.2 Action

```
OUT     ReturnCode                          CardApplicationServiceDescribe(
IN      ConnectionHandle                        connectionHandle,
IN      CardApplicationServiceName              cardApplicationServiceName,
OUT     CardApplicationServiceDescription       serviceDescription
                                            );
```

8.9.3 Parameters

connectionHandle connection handle

cardApplicationServiceName name of the card-application-service to be described

serviceDescription URI, URL, or full description (syntax) of the named card-application-service (may be used as a basis for correct syntax for SAL requests on this named card-application-service via the ExecuteAction action)

8.9.4 Prerequisites

A valid `connectionHandle` is expected.

8.9.5 Return codes

`API_OK`

`API_INCORRECT_PARAMETER`
`API_NAMED_ENTITY_NOT_FOUND`

`API_NOT_INITIALIZED`
`API_SECURITY_CONDITION_NOT_SATISFIED`
`API_COMMUNICATION_FAILURE`

8.9.6 Impact on current state

None

8.10 ExecuteAction

8.10.1 Purpose

This action shall request access to an action in a card-application-service that is not defined in ISO/IEC 24727-3.

The functionality of the actions accessed through this action is out of scope.

8.10.2 Action

```
OUT     ReturnCode                  ExecuteAction(
IN      ConnectionHandle                    connectionHandle,
IN      CardApplicationServiceName          cardApplicationServiceName,
IN      ActionName                          actionName,
IN      ExecuteActionRequest                request,
OUT     ExecuteActionConfirmation           confirmation
                                    );
```

8.10.3 Parameters

connectionHandle	connection handle
cardApplicationServiceName	name of the card-application-service containing the action to be requested
actionName	name of the action to be requested
request	action request data
confirmation	action confirmation data

8.10.4 Prerequisites

A valid `connectionHandle` is expected.

8.10.5 Return codes

```
API_OK

API_INCORRECT_PARAMETER
API_NAMED_ENTITY_NOT_FOUND

API_NOT_INITIALIZED
API_SECURITY_CONDITION_NOT_SATISFIED
API_INSUFFICIENT_RESOURCES
API_COMMUNICATION_FAILURE
```

8.10.6 Impact on current state

None

9 Named data service

9.1 General

This clause defines actions through which data can be managed within a card-application.

9.2 DataSetList

9.2.1 Purpose

This action shall list the names of data-sets defined in the current card-application.

9.2.2 Action

```
OUT     ReturnCode          DataSetList(
IN      ConnectionHandle            connectionHandle,
OUT     DataSetNameList             dataSetNameList
                            );
```

9.2.3 Parameters

connectionHandle connection handle

dataSetNameList list of names of the data-sets defined within the current card-application

9.2.4 Prerequisites

A valid `connectionHandle` is expected.

9.2.5 Return codes

`API_OK`

`API_INCORRECT_PARAMETER`

`API_NOT_INITIALIZED`
`API_SECURITY_CONDITION_NOT_SATISFIED`
`API_COMMUNICATION_FAILURE`

9.2.6 Impact on current state

None

9.3 DataSetCreate

9.3.1 Purpose

This action shall create a new data-set in the current card-application.

9.3.2 Action

```
OUT       ReturnCode            DataSetCreate(
IN        ConnectionHandle            connectionHandle,
IN        DataSetName                 dataSetName,
IN        AccessControlList           dataSetACL
                                );
```

9.3.3 Parameters

connectionHandle connection handle

dataSetName name of the data-set

dataSetACL access control list for the data-set

9.3.4 Prerequisites

A valid `connectionHandle` is expected.

9.3.5 Return codes

```
API_OK

API_INCORRECT_PARAMETER
API_NAME_EXISTS

API_NOT_INITIALIZED
API_SECURITY_CONDITION_NOT_SATISFIED
API_INSUFFICIENT_RESOURCES
API_COMMUNICATION_FAILURE
```

9.3.6 Impact on current state

Upon successful completion, the new data-set becomes the current data-set in the current card-application.

9.4 DataSetSelect

9.4.1 Purpose

This action shall select the named data-set in the current card-application.

9.4.2 Action

```
OUT       ReturnCode           DataSetSelect(
IN        ConnectionHandle           connectionHandle,
IN        DataSetName                dataSetName
                                 );
```

9.4.3 Parameters

connectionHandle connection handle

dataSetName name of the data-set

9.4.4 Prerequisites

A valid `connectionHandle` is expected.

9.4.5 Return codes

```
API_OK

API_INCORRECT_PARAMETER
API_NAMED_ENTITY_NOT_FOUND

API_NOT_INITIALIZED
API_SECURITY_CONDITION_NOT_SATISFIED
API_COMMUNICATION_FAILURE
```

9.4.6 Impact on current state

Upon successful completion, the selected data-set becomes the current data-set in the current card-application.

9.5 DataSetDelete

9.5.1 Purpose

This action shall delete the named data-set within the current card-application including all of the DSIs within that data-set.

9.5.2 Action

```
OUT       ReturnCode          DataSetDelete(
IN        ConnectionHandle            connectionHandle,
IN        DataSetName                 dataSetName
                              );
```

9.5.3 Parameters

connectionHandle connection handle

dataSetName name of the data-set

9.5.4 Prerequisites

A valid `connectionHandle` is expected.

9.5.5 Return codes

```
API_OK

API_INCORRECT_PARAMETER
API_NAMED_ENTITY_NOT_FOUND

API_NOT_INITIALIZED
API_SECURITY_CONDITION_NOT_SATISFIED
API_COMMUNICATION_FAILURE
```

9.5.6 Impact on current state

If the current data-set at the time this action is requested is the data-set to be deleted, the current data-set is undefined following the action confirmation.

9.6 DSIList

9.6.1 Purpose

This action shall list the names of DSIs in the current data-set.

9.6.2 Action

```
OUT       ReturnCode          DSIList(
IN        ConnectionHandle              connectionHandle,
OUT       DSINameList                   dsiNameList
                              );
```

9.6.3 Parameters

connectionHandle connection handle

dsiNameList list of names of the DSIs in the current data-set

9.6.4 Prerequisites

A valid `connectionHandle` is expected. A data-set shall be selected.

9.6.5 Return codes

```
API_OK

API_INCORRECT_PARAMETER

API_NOT_INITIALIZED
API_SECURITY_CONDITION_NOT_SATISFIED
API_PREREQUISITE_NOT_SATISFIED
API_COMMUNICATION_FAILURE
```

9.6.6 Impact on current state

None

9.7 DSICreate

9.7.1 Purpose

This action shall create a new DSI in the current data-set.

9.7.2 Action

```
OUT       ReturnCode          DSICreate(
IN        ConnectionHandle            connectionHandle,
IN        DSIName                     dsiName,
IN        DSIContent                  dsiContent
                              );
```

9.7.3 Parameters

connectionHandle connection handle

dsiName name of the DSI to be created in the current data-set

dsiContent content to be stored in the DSI

9.7.4 Prerequisites

A valid `connectionHandle` is expected. A data-set shall be selected.

9.7.5 Return codes

```
API_OK

API_INCORRECT_PARAMETER
API_NAME_EXISTS

API_NOT_INITIALIZED
API_SECURITY_CONDITION_NOT_SATISFIED
API_PREREQUISITE_NOT_SATISFIED
API_INSUFFICIENT_RESOURCES
API_COMMUNICATION_FAILURE
```

9.7.6 Impact on current state

None

9.8 DSIDelete

9.8.1 Purpose

This action shall delete the named DSI from the current data-set.

9.8.2 Action

```
OUT       ReturnCode          DSIDelete(
IN        ConnectionHandle            connectionHandle,
IN        DSIName                     dsiName
                              );
```

9.8.3 Parameters

connectionHandle connection handle

dsiName name of the DSI to be deleted in the current data-set

9.8.4 Prerequisites

A valid `connectionHandle` is expected. A data-set shall be selected.

9.8.5 Return codes

```
API_OK

API_INCORRECT_PARAMETER
API_NAMED_ENTITY_NOT_FOUND

API_NOT_INITIALIZED
API_SECURITY_CONDITION_NOT_SATISFIED
API_PREREQUISITE_NOT_SATISFIED
API_COMMUNICATION_FAILURE
```

9.8.6 Impact on current state

None

9.9 DSIWrite

9.9.1 Purpose

This action shall replace the contents of the named DSI in the current data-set with the data provided.

9.9.2 Action

```
OUT       ReturnCode           DSIWrite(
IN        ConnectionHandle             connectionHandle,
IN        DSIName                      dsiName,
IN        DSIContent                   dsiContent
                               );
```

9.9.3 Parameters

connectionHandle connection handle

dsiName name of the DSI to be written in the current data-set

dsiContent content to be stored in the DSI

9.9.4 Prerequisites

A valid `connectionHandle` is expected. A data-set shall be selected.

9.9.5 Return codes

```
API_OK

API_INCORRECT_PARAMETER
API_NAMED_ENTITY_NOT_FOUND

API_NOT_INITIALIZED
API_SECURITY_CONDITION_NOT_SATISFIED
API_PREREQUISITE_NOT_SATISFIED
API_INSUFFICIENT_RESOURCES
API_COMMUNICATION_FAILURE
```

9.9.6 Impact on current state

None

9.10 DSIRead

9.10.1 Purpose

This action shall return the contents of the named DSI in the current data-set.

9.10.2 Action

```
OUT       ReturnCode          DSIRead(
IN        ConnectionHandle            connectionHandle,
IN        DSIName                     dsiName,
OUT       DSIContent                  dsiContent
                              );
```

9.10.3 Parameters

connectionHandle connection handle

dsiName name of the DSI to be read in the current data-set

dsiContent content of the DSI

9.10.4 Prerequisites

A valid `connectionHandle` is expected. A data-set shall be selected.

9.10.5 Return codes

```
API_OK

API_INCORRECT_PARAMETER
API_NAMED_ENTITY_NOT_FOUND

API_NOT_INITIALIZED
API_SECURITY_CONDITION_NOT_SATISFIED
API_PREREQUISITE_NOT_SATISFIED
API_COMMUNICATION_FAILURE
```

9.10.6 Impact on current state

None

10 Cryptographic service

10.1 General

This clause defines actions through which cryptographic operations can be performed.

The algorithms used in the performance of these actions depend on the authentication protocols as defined in Annex A.

10.2 Encipher

10.2.1 Purpose

This action shall encipher the provided data according to the cryptographic operation specified in the authentication protocol in the named differential-identity.

10.2.2 Action

```
OUT       ReturnCode           Encipher(
IN        ConnectionHandle             connectionHandle,
IN        DIDScope                     didScope,
IN        DIDName                      didName,
IN        CipherBuffer                 plainText,
OUT       CipherBuffer                 cipherText
                               );
```

10.2.3 Parameters

connectionHandle connection handle

didScope scope of the named differential-identity

didName name of the differential-identity providing the cryptographic algorithm

plainText data to be enciphered

cipherText ciphertext

10.2.4 Prerequisites

A valid `connectionHandle` is expected.

10.2.5 Return codes

```
API_OK

API_INCORRECT_PARAMETER
API_NAMED_ENTITY_NOT_FOUND
API_PROTOCOL_NOT_RECOGNIZED
API_INAPPROPRIATE_PROTOCOL_FOR_ACTION

API_NOT_INITIALIZED
API_SECURITY_CONDITION_NOT_SATISFIED
API_INSUFFICIENT_RESOURCES
API_COMMUNICATION_FAILURE
```

10.2.6 Impact on current state

None

10.3 Decipher

10.3.1 Purpose

This action shall decipher the provided data according to the cryptographic operation of the protocol in the named differential-identity.

10.3.2 Action

```
OUT       ReturnCode          Decipher(
IN        ConnectionHandle            connectionHandle,
IN        DIDScope                    didScope,
IN        DIDName                     didName,
IN        CipherBuffer                cipherText,
OUT       CipherBuffer                plainText
                              );
```

10.3.3 Parameters

connectionHandle connection handle

didScope scope of the named differential-identity

didName name of the differential-identity providing the cryptographic algorithm

cipherText data to be deciphered

plainText plaintext

10.3.4 Prerequisites

A valid `connectionHandle` is expected.

10.3.5 Return codes

```
API_OK

API_INCORRECT_PARAMETER
API_NAMED_ENTITY_NOT_FOUND
API_PROTOCOL_NOT_RECOGNIZED
API_INAPPROPRIATE_PROTOCOL_FOR_ACTION

API_NOT_INITIALIZED
API_SECURITY_CONDITION_NOT_SATISFIED
API_INSUFFICIENT_RESOURCES
API_COMMUNICATION_FAILURE
```

10.3.6 Impact on current state

None

10.4 GetRandom

10.4.1 Purpose

This action shall return a random value generated in accordance with the protocol of the named differential-identity.

10.4.2 Action

```
OUT     ReturnCode          GetRandom(
IN      ConnectionHandle            connectionHandle,
IN      DIDScope                    didScope,
IN      DIDName                     didName,
OUT     RandomDataBuffer            random
                            );
```

10.4.3 Parameters

connectionHandle connection handle

didScope scope of the named differential-identity

didName name of the differential-identity which defines the attributes of the random value

random random value

10.4.4 Prerequisites

A valid `connectionHandle` is expected.

10.4.5 Return codes

```
API_OK

API_INCORRECT_PARAMETER
API_NAMED_ENTITY_NOT_FOUND
API_PROTOCOL_NOT_RECOGNIZED
API_INAPPROPRIATE_PROTOCOL_FOR_ACTION

API_NOT_INITIALIZED
API_SECURITY_CONDITION_NOT_SATISFIED
API_INSUFFICIENT_RESOURCES
API_COMMUNICATION_FAILURE
```

10.4.6 Impact on current state

None

10.5 Hash

10.5.1 Purpose

This action shall hash the provided message according to the authentication protocol and marker of the named differential-identity.

10.5.2 Action

```
OUT       ReturnCode           Hash(
IN        ConnectionHandle             connectionHandle,
IN        DIDScope                     didScope,
IN        DIDName                      didName,
IN        MessageBuffer                message,
OUT       HashBuffer                   hash
                               );
```

10.5.3 Parameters

connectionHandle connection handle

didScope scope of the named differential-identity

didName name of the differential-identity to use for generating the hash

message message to be hashed

hash hash of provided message

10.5.4 Prerequisites

A valid `connectionHandle` is expected.

10.5.5 Return codes

```
API_OK

API_INCORRECT_PARAMETER
API_NAMED_ENTITY_NOT_FOUND
API_PROTOCOL_NOT_RECOGNIZED
API_INAPPROPRIATE_PROTOCOL_FOR_ACTION

API_NOT_INITIALIZED
API_SECURITY_CONDITION_NOT_SATISFIED
API_INSUFFICIENT_RESOURCES
API_COMMUNICATION_FAILURE
```

10.5.6 Impact on current state

None

10.6 Sign

10.6.1 Purpose

This action shall sign the provided message according to the authentication protocol and marker of the named differential-identity.

10.6.2 Action

```
OUT     ReturnCode              Sign(
IN      ConnectionHandle                connectionHandle,
IN      DIDScope                        didScope,
IN      DIDName                         didName,
IN      MessageBuffer                   message,
OUT     SignatureBuffer                 signature
                                );
```

10.6.3 Parameters

connectionHandle connection handle

didScope scope of the named differential-identity

didName name of the differential-identity to use for generating the digital signature

message message to be signed

signature signature of provided message

10.6.4 Prerequisites

A valid `connectionHandle` is expected.

10.6.5 Return codes

```
API_OK

API_INCORRECT_PARAMETER
API_NAMED_ENTITY_NOT_FOUND
API_PROTOCOL_NOT_RECOGNIZED
API_INAPPROPRIATE_PROTOCOL_FOR_ACTION

API_NOT_INITIALIZED
API_SECURITY_CONDITION_NOT_SATISFIED
API_INSUFFICIENT_RESOURCES
API_COMMUNICATION_FAILURE
```

10.6.6 Impact on current state

None

10.7 VerifySignature

10.7.1 Purpose

This action shall perform the verification of a digital signature using the authentication protocol and marker of the named differential-identity.

10.7.2 Action

```
OUT       ReturnCode          VerifySignature(
IN        ConnectionHandle              connectionHandle,
IN        DIDScope                      didScope,
IN        DIDName                       didName,
IN        SignatureBuffer               signature,
IN        MessageBuffer                 message
                              );
```

10.7.3 Parameters

connectionHandle connection handle

didScope scope of the named differential-identity

didName name of the differential-identity to use for verifying the signature

signature signature to be authenticated

message data that was signed

10.7.4 Prerequisites

A valid `connectionHandle` is expected.

10.7.5 Return codes

```
API_OK

API_INCORRECT_PARAMETER
API_NAMED_ENTITY_NOT_FOUND
API_INVALID_SIGNATURE
API_PROTOCOL_NOT_RECOGNIZED
API_INAPPROPRIATE_PROTOCOL_FOR_ACTION

API_NOT_INITIALIZED
API_SECURITY_CONDITION_NOT_SATISFIED
API_INSUFFICIENT_RESOURCES
API_COMMUNICATION_FAILURE
```

10.7.6 Impact on current state

None

10.8 VerifyCertificate

10.8.1 Purpose

This action shall perform the verification of a digital certificate using the authentication protocol and marker of the named differential-identity.

If the certificate supplied is valid but not signed by an on-card differential-identity, successive requests can be made supplying the certificate of the signer of the previous certificate. Repeating this until the supplied certificate is signed by an on-card differential-identity verifies the chain of trust from the on-card differential-identity to the owner of the original certificate supplied.

10.8.2 Action

```
OUT       ReturnCode           VerifyCertificate(
IN        ConnectionHandle           connectionHandle,
IN        DIDScope                   didScope,
IN        DIDName                    rootCert,
IN        CertificateType            certificateType,
IN        Certificate                certificate
                                );
```

10.8.3 Parameters

connectionHandle	connection handle
didScope	scope of the named differential-identity
rootCert	name of the optional differential-identity to use for verifying the signature on the certificate
certificateType	format of certificate, consisting of an index and object identifier
certificate	certificate to be verified

10.8.4 Prerequisites

A valid `connectionHandle` is expected.

10.8.5 Return codes

```
API_OK

API_INCORRECT_PARAMETER
API_NAMED_ENTITY_NOT_FOUND
API_INVALID_KEY
API_INVALID_SIGNATURE
API_PROTOCOL_NOT_RECOGNIZED
API_INAPPROPRIATE_PROTOCOL_FOR_ACTION

API_NOT_INITIALIZED
API_SECURITY_CONDITION_NOT_SATISFIED
API_INSUFFICIENT_RESOURCES
API_COMMUNICATION_FAILURE
```

10.8.6 Impact on current state

None

11 Differential-identity service

11.1 General

This clause defines actions through which differential-identities can be created and managed within a card-application.

11.2 DIDList

11.2.1 Purpose

This action shall list the names of the differential-identities defined within the current card-application.

11.2.2 Action

```
OUT     ReturnCode          DIDList(
IN      ConnectionHandle            connectionHandle,
IN      DIDQualifier                filter,
OUT     DIDNameList                 didNameList
                            );
```

11.2.3 Parameters

connectionHandle connection handle

filter **didNameList** only contains the names of those DIDs with qualifier fields exactly matching the filter; a NULL filter matches all qualifier fields and values.

didNameList list of names of the differential-identities in the current card-application

11.2.4 Prerequisites

A valid connectionHandle is expected.

11.2.5 Return codes

API_OK

API_INCORRECT_PARAMETER

API_NOT_INITIALIZED
API_SECURITY_CONDITION_NOT_SATISFIED
API_COMMUNICATION_FAILURE

11.2.6 Impact on current state

None

11.3 DIDCreate

11.3.1 Purpose

This action shall create a new differential-identity within the current card-application.

The format of the `didUpdateData` parameter is defined in Annex A for each authentication protocol.

11.3.2 Action

```
OUT       ReturnCode            DIDCreate(
IN        ConnectionHandle              connectionHandle,
IN        DIDName                       didName,
IN        ObjectIdentifier              authProtocolOID,
IN        DIDUpdateData                 didUpdateData,
IN        AccessControlList             didACL
                                );
```

11.3.3 Parameters

connectionHandle connection handle

didName name of the differential-identity to be created

authProtocolOID OID of an authentication protocol specified in Annex A or registered elsewhere within ISO/IEC 24727

didUpdateData structure containing parameters for the creation of the differential-identity, specific to the authentication protocol identified by `authProtocolOID`

didACL access control list governing access to the differential-identity

11.3.4 Prerequisites

A valid `connectionHandle` is expected.

11.3.5 Return codes

```
API_OK

API_INCORRECT_PARAMETER
API_NAME_EXISTS
API_PROTOCOL_NOT_RECOGNIZED

API_NOT_INITIALIZED
API_SECURITY_CONDITION_NOT_SATISFIED
API_INSUFFICIENT_RESOURCES
API_COMMUNICATION_FAILURE
```

11.3.6 Impact on current state

None

11.4 DIDGet

11.4.1 Purpose

This action shall return information about a differential-identity recognized in the current card-application.

The returned data shall contain the protocol field, all exportable elements of the maker according to the definition of the protocol in Annex A, and the qualifier field, if any.

11.4.2 Action

```
OUT       ReturnCode          DIDGet(
IN        ConnectionHandle            connectionHandle,
IN        DIDScope                    didScope,
IN        DIDName                     didName,
OUT       DIDStructure                didStructure
                              );
```

11.4.3 Parameters

connectionHandle connection handle

didScope scope of the named differential-identity

didName name of the differential-identity for which discovery information is requested

didStructure structure containing discoverable parameters of the differential-identity

11.4.4 Prerequisites

A valid `connectionHandle` is expected.

11.4.5 Return codes

```
API_OK

API_INCORRECT_PARAMETER
API_NAMED_ENTITY_NOT_FOUND
API_PROTOCOL_NOT_RECOGNIZED

API_NOT_INITIALIZED
API_SECURITY_CONDITION_NOT_SATISFIED
API_COMMUNICATION_FAILURE
```

11.4.6 Impact on current state

None

11.5 DIDUpdate

11.5.1 Purpose

This action shall store or generate new marker for the named differential-identity defined within the current card-application appropriate for its existing protocol.

The `didUpdateData` arugment contains the new marker to be stored or the parameters to be used in the generation of a new marker according to the protocol of the differential-identity. The didUpdateData argument may also contain the differential-identity variant information.

11.5.2 Action

```
OUT       ReturnCode          DIDUpdate(
IN        ConnectionHandle            connectionHandle,
IN        DIDName                     didName,
IN        DIDUpdateData               didUpdateData
                              );
```

11.5.3 Parameters

connectionHandle connection handle

didName name of the differential-identity to be updated

didUpdateData data for the update of a differential-identity

11.5.4 Prerequisites

A valid `connectionHandle` is expected.

11.5.5 Return codes

```
API_OK

API_INCORRECT_PARAMETER
API_NAMED_ENTITY_NOT_FOUND
API_PROTOCOL_NOT_RECOGNIZED

API_NOT_INITIALIZED
API_SECURITY_CONDITION_NOT_SATISFIED
API_INSUFFICIENT_RESOURCES
API_COMMUNICATION_FAILURE
```

11.5.6 Impact on current state

None

11.6 DIDDelete

11.6.1 Purpose

This action shall delete the named differential-identity defined in the current card-application.

11.6.2 Action

```
OUT       ReturnCode          DIDDelete(
IN        ConnectionHandle            connectionHandle,
IN        DIDName                     didName
                              );
```

11.6.3 Parameters

connectionHandle connection handle

didName name of the differential-identity to be deleted from the card-application

11.6.4 Prerequisites

A valid `connectionHandle` is expected.

11.6.5 Return codes

```
API_OK

API_INCORRECT_PARAMETER
API_NAMED_ENTITY_NOT_FOUND

API_NOT_INITIALIZED
API_SECURITY_CONDITION_NOT_SATISFIED
API_COMMUNICATION_FAILURE
```

11.6.6 Impact on current state

Upon successful completion, the authentication state of the differential-identity deleted is removed from the current state.

11.7 DIDAuthenticate

11.7.1 Purpose

This action shall perform the authentication protocol of the named differential-identity recognized in the current card-application.

Consecutive requests of this action for the same connection handle and differential-identity may be necessary to complete the protocol as defined in Annex A or registered elsewhere in ISO/IEC 24727.

11.7.2 Action

```
OUT       ReturnCode                DIDAuthenticate(
IN        ConnectionHandle                connectionHandle,
IN        DIDScope                        didScope,
IN        DIDName                         didName,
IN/OUT    DIDAuthenticationData           authenticationProtocolData,
IN        ConnectionHandle                samConnectionHandle
                                    );
```

11.7.3 Parameters

connectionHandle	connection handle
didScope	scope of the named differential-identity
didName	name of the differential-identity to be authenticated
authenticationProtocolData	data exchanged according to the protocol
samConnectionHandle	optional argument to a card-application that may be used by the ISO/IEC 24727-3 layer

11.7.4 Prerequisites

A valid `connectionHandle` is expected.

11.7.5 Return codes

```
API_OK
API_NEXT_REQUEST

API_INCORRECT_PARAMETER
API_NAMED_ENTITY_NOT_FOUND
API_PROTOCOL_NOT_RECOGNIZED
API_INAPPROPRIATE_PROTOCOL_FOR_ACTION

API_NOT_INITIALIZED
API_SECURITY_CONDITION_NOT_SATISFIED
API_INSUFFICIENT_RESOURCES
API_COMMUNICATION_FAILURE
```

11.7.6 Impact on current state

See Annex A.

12 Authorization service

12.1 General

This clause defines actions used to discover and modify access control lists.

12.2 ACLList

12.2.1 Purpose

This action shall return the access control list for the named target.

If the target type is data-set, the target name shall only apply to data-sets defined in the current card-application. If the target type is differential-identity, the target name shall only apply to differential-identities defined in the current card-application.

12.2.2 Action

```
OUT       ReturnCode            ACLList(
IN        ConnectionHandle                connectionHandle,
IN        TargetType                      targetType,
IN        TargetName                      targetName,
OUT       AccessControlList               targetACL
                                );
```

12.2.3 Parameters

connectionHandle connection handle

targetType type of the target

targetName name of the target

targetACL access control list

12.2.4 Prerequisites

A valid `connectionHandle` is expected.

12.2.5 Return codes

```
API_OK

API_INCORRECT_PARAMETER
API_NAMED_ENTITY_NOT_FOUND

API_NOT_INITIALIZED
API_SECURITY_CONDITION_NOT_SATISFIED
API_COMMUNICATION_FAILURE
```

12.2.6 Impact on current state

None

12.3 ACLModify

12.3.1 Purpose

This action shall modify the access rule for the named action within the access control list of the named target.

If the target type is data-set, the target name shall only apply to data-sets defined in the current card-application. If the target type is differential-identity, the target name shall only apply to differential-identities defined in the current card-application.

12.3.2 Action

```
OUT       ReturnCode                    ACLModify(
IN        ConnectionHandle                        connectionHandle,
IN        TargetType                              targetType,
IN        TargetName                              targetName,
IN        CardApplicationServiceName              cardApplicationServiceName,
IN        ActionName                              actionName,
IN        SecurityCondition                       securityCondition
                                        );
```

12.3.3 Parameters

connectionHandle	connection handle
targetType	type of the target
targetName	name of the target
cardApplicationServiceName	name of the card-application-service containing the named action
actionName	name of the action whose access rule is to be modified
securityCondition	full boolean expression in terms of differential-identity names

12.3.4 Prerequisites

A valid `connectionHandle` is expected.

12.3.5 Return codes

```
API_OK

API_INCORRECT_PARAMETER
API_NAMED_ENTITY_NOT_FOUND

API_NOT_INITIALIZED
API_SECURITY_CONDITION_NOT_SATISFIED
API_INSUFFICIENT_RESOURCES
API_COMMUNICATION_FAILURE
```

12.3.6 Impact on current state

None

Annex A
(normative)

Authentication protocols

A.1 General

The protocol associated with a differential-identity during its creation shall be one of the protocols defined in this annex or registered elsewhere in ISO/IEC 24727. A card-application is not required to implement every protocol defined in this annex.

Authentication of differential-identities is accomplished through the successful execution of the authentication protocols defined in this annex.

In the following figures, general descriptions are provided for the requests placed on the ISO/IEC 24727-2 interface by the ISO/IEC 24727-3 layer rather than an actual encoding of the requests themselves. The actual encoding of the requests is the responsibility of the ISO/IEC 24727-3 implementation. Some ISO/IEC 24727 stack configurations may not require the implementation of an ISO/IEC 24727-2 GCI layer.

A.2 Common Definitions

A.2.1 Symbols and Operators

The following conventions are used in the definition of computations in this Annex.

=?=	denotes a comparison operation
||	denotes a concatenation operation
algorithm (**input**)	denotes the application of **algorithm** to **input**
algorithm [**key**] (**input**)	denotes the application of **algorithm** to **input** using **key**
algorithm^{-1} (**input**)	denotes the application of the inverse of **algorithm** to **input** (For an encryption algorithm, this denotes its application for decipherment.)
RNG (**size**)	denotes the application of a random number generator to generate **size** bytes of data, where the generator used is out of scope

A.2.2 Structures

Actions on the ISO/IEC 24727-3 API supporting creation and update of differential-identities shall supply a `DIDUpdateData` structure as an input argument to those actions. This structure is defined as

```
DIDUpdateData ::= SEQUENCE {
                                marker      OCTET STRING,
                                qualifier   DiDQualifier OPTIONAL
}
```

where `marker` is a specific marker as defined below in each specific sub-clause for each authentication protocol.

A confirmation to the DIDGet action shall include a **`DIDStructure`** parameter as defined below.

```
DIDStructure ::= SEQUENCE {
                              name              DIDName,
                              authProtocol      OBJECT IDENTIFIER,
                              scope             DIDScope,
                              authenticated     BOOLEAN,
                              marker            OCTET STRING,
                              qualifier         DiDQualifierType OPTIONAL
}
```

At a minimum, the DIDGet confirmation shall return valid `name`, `authProtocol`, `scope`, and `authenticated` values. Partial or complete, `marker` and `qualifier` values may be returned in the confirmation, subject to implementation and policy choices.

A.3 Simple Assertion

Authentication of a differential-identity associated with this authentication protocol is achieved by simple assertion.

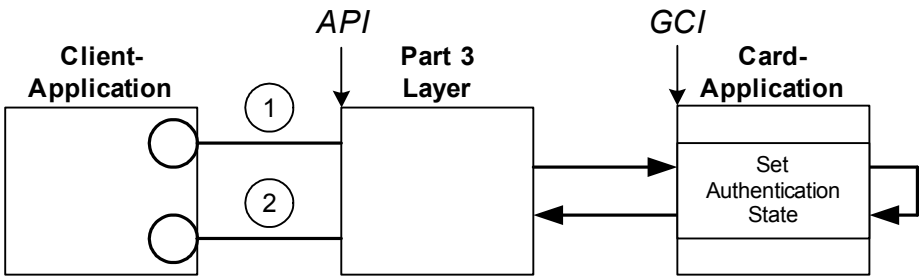

Figure A.1 — Simple Assertion

Protocol Object Identifier

This protocol is identified by the OID `{ iso(1) standard(0) iso24727(24727) part3(3) annex-a(0) simple-assertion(3) }`.

A.3.1 Marker

A differential-identity which uses this protocol has an empty marker (`OCTET STRING` of length zero).

A.3.2 DIDCreate

A request of the DIDCreate action intended to create a differential-identity for use with this authentication protocol shall include a `DIDUpdateData` parameter where `marker` is a zero-length `OCTET STRING` for this authentication protocol.

A.3.3 DIDUpdate

A request of the DIDUpdate action on a differential-identity which uses this protocol shall return the `API_INAPPROPRIATE_PROTOCOL_FOR_ACTION` return code.

A.3.4 DIDGet

The confirmation to the DIDGet action involving a differential-identity which uses this authentication protocol shall include a `DIDStructure`.

A.3.5 Authentication

This authentication protocol is executed through a single request of the DIDAuthenticate action with an empty **`authenticationProtocolData`** parameter as defined below.

A request of the CardApplicationStartSession action on a differential-identity which uses this protocol shall return the `API_INAPPROPRIATE_PROTOCOL_FOR_ACTION` return code.

A.3.5.1 Procedure

(1) **authenticationProtocolData** ::= empty OCTET STRING

(2) **authenticationProtocolData** ::= empty OCTET STRING

A.3.5.2 Impact on current state

Upon successful completion of this protocol, the authentication state of the named differential-identity shall be set to TRUE within the card-application.

A.3.6 Encipher

A request of the Encipher action on a differential-identity which uses this protocol shall return the API_INAPPROPRIATE_PROTOCOL_FOR_ACTION return code.

A.3.7 Decipher

A request of the Decipher action on a differential-identity which uses this protocol shall return the API_INAPPROPRIATE_PROTOCOL_FOR_ACTION return code.

A.3.8 GetRandom

A request of the GetRandom action on a differential-identity which uses this protocol shall return the API_INAPPROPRIATE_PROTOCOL_FOR_ACTION return code.

A.3.9 Hash

A request of the Hash action on a differential-identity which uses this protocol shall return the API_INAPPROPRIATE_PROTOCOL_FOR_ACTION return code.

A.3.10 Sign

A request of the Sign action on a differential-identity which uses this protocol shall return the API_INAPPROPRIATE_PROTOCOL_FOR_ACTION return code.

A.3.11 VerifySignature

A request of the VerifySignature action on a differential-identity which uses this protocol shall return the API_INAPPROPRIATE_PROTOCOL_FOR_ACTION return code.

A.3.12 VerifyCertificate

A request of the VerifyCertificate action on a differential-identity which uses this protocol shall return the API_INAPPROPRIATE_PROTOCOL_FOR_ACTION return code.

A.4 Asymmetric Internal Authenticate

This authentication protocol is a challenge/response protocol using public key cryptography.

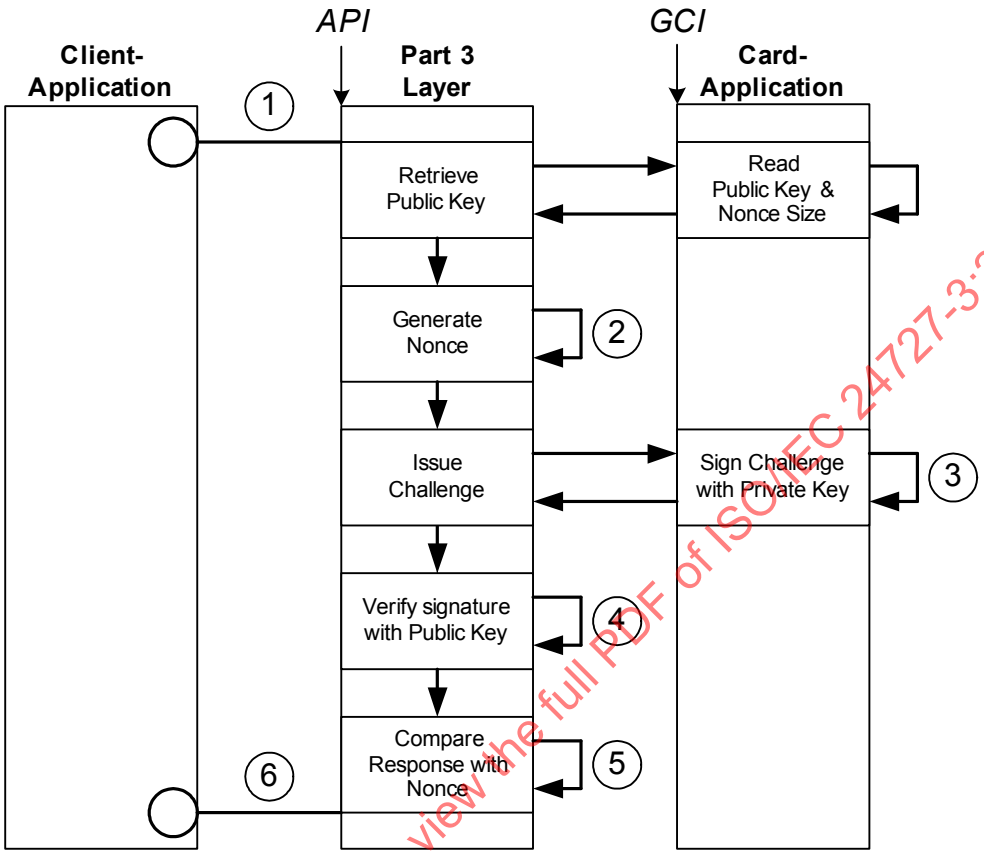

Figure A.2 — Asymmetric Internal Authenticate

A.4.1 Protocol Object Identifier

This protocol is identified by the OID `{ iso(1) standard(0) iso24727(24727) part3(3) annex-a(0) asymmetric-internal-authenticate(4) }`.

A.4.2 Marker

A differential-identity which uses this protocol has the following marker.

```
MarkerAP004 ::= SEQUENCE {
                    signatureAlgorithm    OBJECT IDENTIFIER,
                    hashAlgorithm         OBJECT IDENTIFIER,
                    keySize               INTEGER,
                    CHOICE  {
                              SEQUENCE  {
                                        publicKeyMaterialOCTET STRING,
                                        privateKey        OCTET STRING
                              },
                              generateFlag        NULL
                    }
                    nonceSize   INTEGER
}
```

A.4.3 DIDCreate

A request of the DIDCreate action intended to create a differential-identity for use with this authentication protocol shall include a `DIDUpdateData` parameter where `marker` is the specific marker for this protocol as defined above in A.4.2.

A.4.4 DIDUpdate

A request of the DIDUpdate action involving a differential-identity which uses this authentication protocol shall include a `DIDUpdateData` parameter, where `marker` is the specific marker for this protocol as defined above in A.4.2.

If the original keys were generated within the card-application, as specified by the presence of the `generateFlag` option during the request of the DIDCreate action, a request of the DIDUpdate action which includes keys shall return the `API_INCORRECT_PARAMETER` return code.

If the original keys were supplied by the client-application, as specified by the presence of the `publicKeyMaterial` and `privateKey` options during the request of the DIDCreate action, a request of the DIDUpdate action with the `generateFlag` option shall return the `API_INCORRECT_PARAMETER` return code.

A.4.5 DIDGet

The confirmation to the DIDGet action involving a differential-identity which uses this authentication protocol shall include a `DIDStructure` parameter.

A.4.6 Authentication

This authentication protocol is executed through a single request of the DIDAuthenticate action with an **`authenticationProtocolData`** parameter as defined below.

A request of the CardApplicationStartSession action on a differential-identity which uses this protocol shall return the `API_INAPPROPRIATE_PROTOCOL_FOR_ACTION` return code.

A.4.6.1 Procedure

(1) **`authenticationProtocolData`** `::= empty OCTET STRING`

(2) `nonce` = RNG (`nonceSize`)

(3) `challenge` = `signatureAlgorithm` [`privateKey`] (`nonce`)

(4) `response` = $\texttt{signatureAlgorithm}^{-1}$ [`publicKey`] (`challenge`)

(5) `result` = (`response` =?= `nonce`)

(6) **`authenticationProtocolData`** `::= result BOOLEAN as OCTET STRING`

A.4.6.2 Impact on current state

Upon successful completion of this protocol, the authentication state of the named differential-identity is not affected within the card-application.

A.4.7 Encipher

A request of the Encipher action on a differential-identity which uses this protocol shall return the `API_INAPPROPRIATE_PROTOCOL_FOR_ACTION` return code.

A.4.8 Decipher

`outBuffer` = `encryptionAlgorithm` [`privateKey`] (`inBuffer`)

A.4.9 GetRandom

`random` = RNG (`nonceSize`)

A.4.10 Hash

`hash` = `hashAlgorithm` (`message`)

A.4.11 Sign

`signature` = `encryptionAlgorithm` [`privateKey`] (`message`)

A.4.12 VerifySignature

`message` =?= `encryptionAlgorithm` [`publicKey`] (`signature`)
If TRUE, return `API_OK`, else return `API_INVALID_SIGNATURE`.

A.4.13 VerifyCertificate

Depending on certificateType,
`hash` = `hashAlgorithm` (`certificate` without signature)
`hash` =?= `encryptionAlgorithm` [`publicKey`] (signature from `certificate`)
If TRUE, return `API_OK`, else return `API_INVALID_SIGNATURE`.

A.5 Asymmetric External Authenticate

This authentication protocol is used to establish an authenticated state of a differential-identity in a card-application.

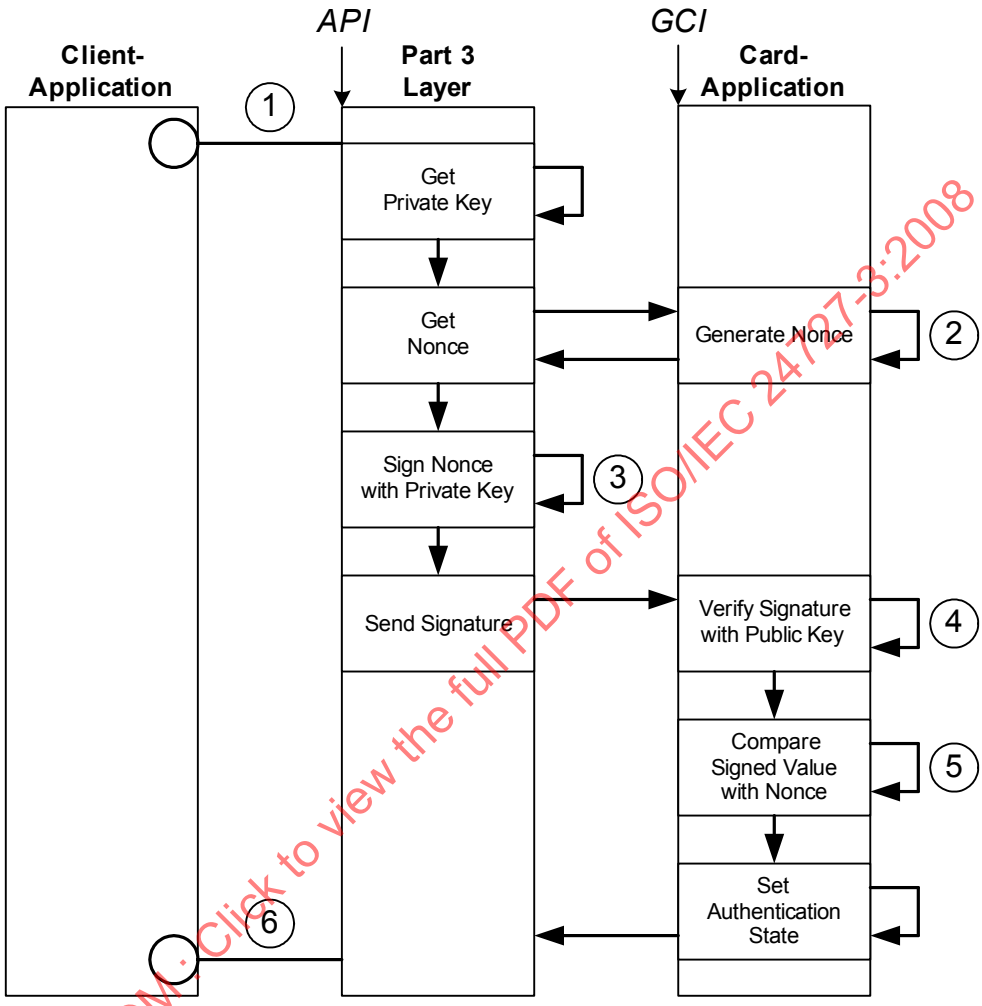

Figure A.3 — Asymmetric External Authenticate

A.5.1 Protocol Object Identifier

This protocol is identified by the OID `{ iso(1) standard(0) iso24727(24727) part3(3) annex-a(0) asymmetric-external-authenticate(5) }`.

A.5.2 Marker

A differential-identity which uses this protocol has the following marker.

```
MarkerAP005 ::= SEQUENCE {
                        encryptionAlgorithm     OBJECT IDENTIFIER,
                        hashAlgorithm           OBJECT IDENTIFIER,
                        keySize                 INTEGER,
                        publicKeyMaterial       OCTET STRING,
                        nonceSize               INTEGER
}
```

A.5.3 DIDCreate

A request of the DIDCreate action intended to create a differential-identity for use with this authentication protocol shall include a `DIDUpdateData` parameter where `marker` is the specific marker for this protocol as defined above in A.5.2.

A.5.4 DIDUpdate

A request of the DIDUpdate action involving a differential-identity which uses this authentication protocol shall include a `DIDUpdateData` parameter, where `marker` is the specific marker for this protocol as defined above in A.5.2.

A.5.5 DIDGet

The confirmation to the DIDGet action involving a differential-identity which uses this authentication protocol shall include a `DIDStructure` parameter.

A.5.6 Authentication

This authentication protocol is executed through a single request of the DIDAuthenticate action with an **`authenticationProtocolData`** parameter as defined below.

A request of the CardApplicationStartSession action on a differential-identity which uses this protocol shall return the `API_INAPPROPRIATE_PROTOCOL_FOR_ACTION` return code.

A.5.6.1 Procedure

(1) **`authenticationProtocolData`** `::= privateKey  OCTET STRING`

or `empty OCTET STRING` if the `samConnectionHandle` option was used in the request to DIDAuthenticate.

(2) `nonce` = `RNG` (`nonceSize`)

(3) `signature` = `encryptionAlgorithm` [`privateKey`] (`nonce`)
If the `samConnectionHandle` option was used in the request to DIDAuthenticate, the ISO/IEC 24727-3 layer shall request this operation be performed by the card-application to which it is connected by this `samConnectionHandle`.

(4) `message` = $\texttt{encryptionAlgorithm}^{-1}$ [`publicKey`] (`signature`)

(5) `result` = (`nonce` =?= `message`)

(6) **`authenticationProtocolData`** `::= empty OCTET STRING`

A.5.6.2 Impact on current state

Upon successful completion of this protocol, if the value of `result` is TRUE, the authentication state of the named differential-identity shall be set to TRUE within the card-application.

A.5.7 Encipher

`outBuffer` = `encryptionAlgorithm` [`publicKey`] (`inBuffer`)

A.5.8 Decipher

A request of the Decipher action on a differential-identity which uses this protocol shall return the `API_INAPPROPRIATE_PROTOCOL_FOR_ACTION` return code.

A.5.9 GetRandom

`random` = `RNG` (`nonceSize`)

A.5.10 Hash

`hash` = `hashAlgorithm` (`message`)

A.5.11 Sign

A request of the Sign action on a differential-identity which uses this protocol shall return the `API_INAPPROPRIATE_PROTOCOL_FOR_ACTION` return code.

A.5.12 VerifySignature

`message` =?= `encryptionAlgorithm` [`publicKey`] (`signature`)
If TRUE, return `API_OK`, else return `API_INVALID_SIGNATURE`.

A.5.13 VerifyCertificate

Depending on certificateType,
`hash` = `hashAlgorithm` (`certificate` without signature)
`hash` =?= `encryptionAlgorithm` [`publicKey`] (signature from `certificate`)
If TRUE, return `API_OK`, else return `API_INVALID_SIGNATURE`.

A.6 Symmetric Internal Authenticate

This authentication protocol is a challenge/response protocol using symmetric cryptography.

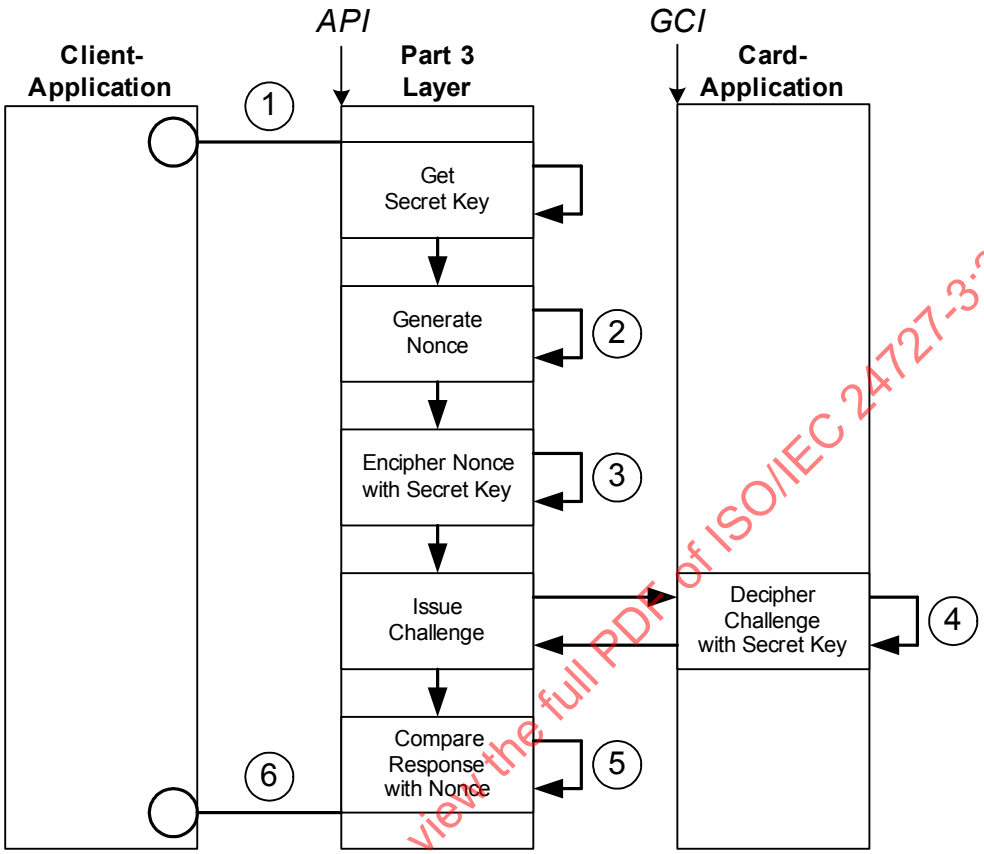

Figure A.4 — Symmetric Internal Authenticate

A.6.1 Protocol Object Identifier

This protocol is identified by the OID `{ iso(1) standard(0) iso24727(24727) part3(3) annex-a(0) symmetric-internal-authenticate(6) }`.

A.6.2 Marker

A differential-identity which uses this protocol has the following marker.

```
MarkerAP006 ::= SEQUENCE {
                        encryptionAlgorithm     OBJECT IDENTIFIER,
                        hashAlgorithm           OBJECT IDENTIFIER,
                        keySize                 INTEGER,
                        secretKey               OCTET STRING,
                        nonceSize               INTEGER
}
```

The `keySize` and `nonceSize` parameters represent the number of bits in the key and nonce, respectively. The `secretKey` parameter is the bitstring of the key itself.

A.6.3 DIDCreate

A request of the DIDCreate action intended to create a differential-identity for use with this authentication protocol shall include a `DIDUpdateData` parameter where `marker` is the specific marker for this protocol as defined above in A.6.2

A.6.4 DIDUpdate

A request of the DIDUpdate action involving a differential-identity which uses this authentication protocol shall include a `DIDUpdateData` parameter, where `marker` is the specific marker for this protocol as defined above in A.6.2.

A.6.5 DIDGet

The confirmation to the DIDGet action involving a differential-identity which uses this authentication protocol shall include a `didStructure` parameter.

A.6.6 Authentication

This authentication protocol is executed through a single request of the DIDAuthenticate action with an **`authenticationProtocolData`** parameter as defined below.

A request of the CardApplicationStartSession action on a differential-identity which uses this protocol shall return the `API_INAPPROPRIATE_PROTOCOL_FOR_ACTION` return code.

A.6.6.1 Procedure

(1) **`authenticationProtocolData`** `::= secretKey OCTET STRING`

or `empty OCTET STRING` if the `samConnectionHandle` option was used in the request to DIDAuthenticate.

(2) `nonce` = RNG (`nonceSize`)

(3) `challenge` = `encryptionAlgorithm` [`secretKey`] (`nonce`)

(4) `response` = $\text{encryptionAlgorithm}^{-1}$ [`secretKey`] (`challenge`)

(5) `result` = (`nonce` =?= `response`)

(6) **`authenticationProtocolData`** `::= result BOOLEAN as OCTET STRING`

A.6.6.2 Impact on current state

Upon successful completion of this protocol, the authentication state of the named differential-identity is not affected within the card-application.

A.6.7 Encipher

`outBuffer` = `encryptionAlgorithm` [`secretKey`] (`inBuffer`)

A.6.8 Decipher

`outBuffer` = $\text{encryptionAlgorithm}^{-1}$ [`secretKey`] (`inBuffer`)

A.6.9 GetRandom

`random` = RNG (`nonceSize`)

A.6.10 Hash

`hash` = `hashAlgorithm` (`message`)

A.6.11 Sign

`digest` = `hashAlgorithm` (`message`)
`signature` = `encryptionAlgorithm` [`secretKey`] (`digest`)

A.6.12 VerifySignature

`digest` = `hashAlgorithm` (`message`)
`digest` =?= $\texttt{encryptionAlgorithm}^{-1}$ [`secretKey`] (`signature`)
If TRUE, return `API_OK`, else return `API_INVALID_SIGNATURE`.

A.6.13 VerifyCertificate

A request of the Encipher action on a differential-identity which uses this protocol shall return the `API_INAPPROPRIATE_PROTOCOL_FOR_ACTION` return code.

A.7 Symmetric External Authenticate

This authentication protocol is used to establish an authenticated state of a differential-identity in a card-application.

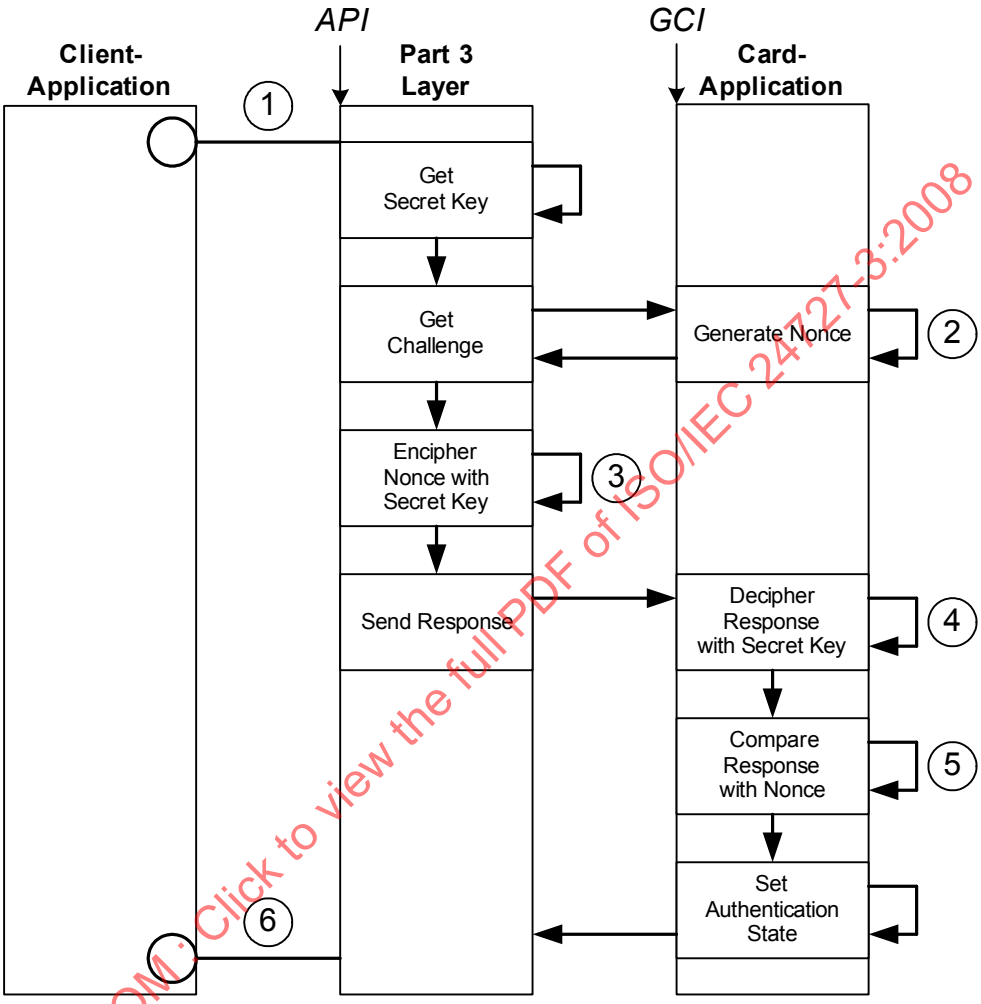

Figure A.5 — Symmetric External Authenticate

A.7.1 Protocol Object Identifier

This protocol is identified by the OID `{ iso(1) standard(0) iso24727(24727) part3(3) annex-a(0) symmetric-external-authenticate(7) }`.

A.7.2 Marker

A differential-identity which uses this protocol has the following marker.

```
MarkerAP007 ::= SEQUENCE {
                        encryptionAlgorithm     OBJECT IDENTIFIER,
                        hashAlgorithm           OBJECT IDENTIFIER,
                        keySize                 INTEGER,
                        secretKey               OCTET STRING,
                        nonceSize               INTEGER
}
```

A.7.3 DIDCreate

A request of the DIDCreate action intended to create a differential-identity for use with this authentication protocol shall include a `DIDUpdateData` parameter where `marker` is the specific marker for this protocol as defined above in A.7.2

A.7.4 DIDUpdate

A request of the DIDUpdate action involving a differential-identity which uses this authentication protocol shall include a `DIDUpdateData` parameter, where `marker` is the specific marker for this protocol as defined above in A.7.2.

A.7.5 DIDGet

The confirmation to the DIDGet action involving a differential-identity which uses this authentication protocol shall include a `DIDStructure` parameter.

A.7.6 Authentication

This authentication protocol is executed through a single request of the DIDAuthenticate action with an **`authenticationProtocolData`** parameter as defined below.

A request of the CardApplicationStartSession action on a differential-identity which uses this protocol shall return the `API_INAPPROPRIATE_PROTOCOL_FOR_ACTION` return code.

A.7.6.1 Procedure

(1) **`authenticationProtocolData`** `::= secretKey:OCTET STRING`

or `empty OCTET STRING` if the `samConnectionHandle` option was used in the request to DIDAuthenticate.

(2) `nonce` = RNG (`nonceSize`)

(3) `response` = `encryptionAlgorithm` [`secretKey`] (`nonce`)

(4) `challenge` = $\texttt{encryptionAlgorithm}^{-1}$ [`secretKey`] (`nonce`)

(5) `result` = (`nonce` =?= `challenge`)

(6) **`authenticationProtocolData`** ::= `empty OCTET STRING`

A.7.6.2 Impact on current state

Upon successful completion of this protocol, the authentication state of the named differential-identity shall be set to the value of `result` within the card-application.

A.7.7 Encipher

`outBuffer` = `encryptionAlgorithm` [`secretKey`] (`inBuffer`)

A.7.8 Decipher

`outBuffer` = `encryptionAlgorithm`$^{-1}$ [`secretKey`] (`inBuffer`)

Note: encryptionAlgorithm is used for decryption and may be a difference form of “encryptionAlgorithm”.

A.7.9 GetRandom

`random` = RNG (`nonceSize`)

A.7.10 Hash

`hash` = `hashAlgorithm` (`message`)

A.7.11 Sign

`digest` = `hashAlgorithm` (`message`)
`signature` = `encryptionAlgorithm` [`secretKey`] (`digest`)

A.7.12 VerifySignature

`digest` = `hashAlgorithm` (`message`)
`digest` =?= `encryptionAlgorithm`$^{-1}$ [`secretKey`] (`signature`)
If TRUE, return `API_OK`, else return `API_INVALID_SIGNATURE`.

A.7.13 VerifyCertificate

A request of the Encipher action on a differential-identity which uses this protocol shall return the `API_INAPPROPRIATE_PROTOCOL_FOR_ACTION` return code.

A.8 Compare

This authentication protocol compares a supplied value with a value stored in the marker of a differential-identity.

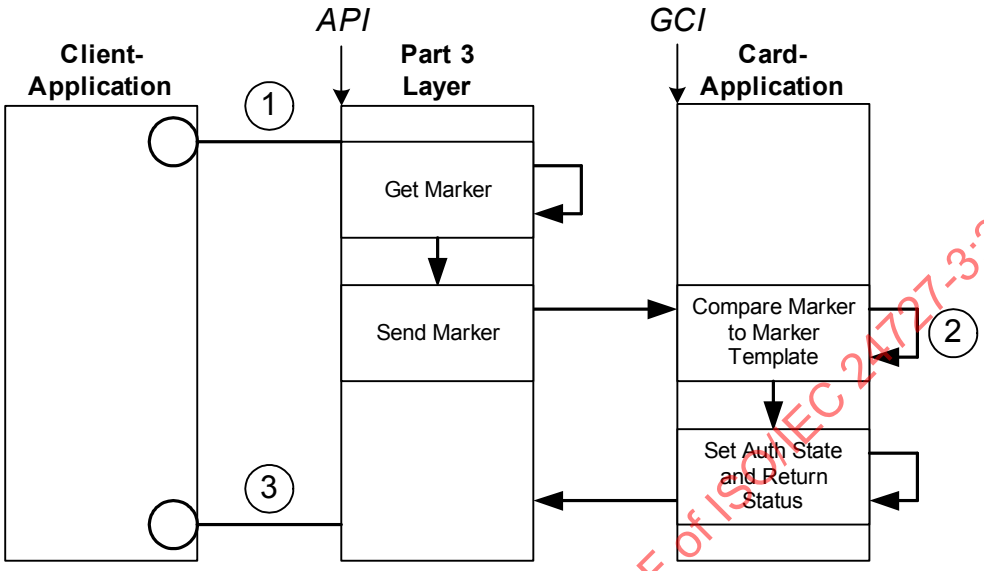

Figure A.6 — Compare

A.8.1 Protocol Object Identifier

This protocol is identified by the OID `{ iso(1) standard(0) iso24727(24727) part3(3) annex-a(0) compare(8) }`.

A.8.2 Marker

A differential-identity which uses this protocol has the following marker.

```
MarkerAP008 ::= SEQUENCE {
                    minDataLength       INTEGER,
                    maxDataLength       INTEGER,
                    paddingCharacter    OCTET STRING,
                    markerTemplate      OCTET STRING
}
```

The `minDataLength` and `maxDataLength` parameters represent the number of bits in the key and nonce, respectively. The `paddingCharacter` parameter is a single byte.

A.8.3 DIDCreate

A request of the DIDCreate action intended to create a differential-identity for use with this authentication protocol shall include a `DIDUpdateData` parameter where `marker` is the specific marker for this protocol as defined above in A.8.2

A.8.4 DIDUpdate

A request of the DIDUpdate action involving a differential-identity which uses this authentication protocol shall include a `DIDUpdateData` parameter, where `marker` is the specific marker for this protocol as defined above in A.8.2

A.8.5 DIDGet

The confirmation to the DIDGet action involving a differential-identity which uses this authentication protocol shall include a `DIDStructure` parameter.

A.8.6 Authentication

This authentication protocol is executed through a single request of the DIDAuthenticate action with an **`authenticationProtocolData`** parameter as defined below.

A request of the CardApplicationStartSession action on a differential-identity which uses this protocol shall return the `API_INAPPROPRIATE_PROTOCOL_FOR_ACTION` return code.

A.8.6.1 Procedure

(1) **`authenticationProtocolData`** `::= markerTemplate OCTET STRING SIZE(minDataLength..maxDataLength)`

The `marker` shall be formatted in big-endian byte order with padding of the least significant byte end.

(2) `result` = (`marker` =?= `markerTemplate`)
The comparison is a bit-by-bit.

(3) **`authenticationProtocolData`** `::= empty OCTET STRING`

A.8.6.2 Impact on current state

Upon successful completion of this protocol, the authentication state of the named differential-identity shall be set to the value of `result` within the card-application.

A.8.7 Encipher

A request of the Encipher action on a differential-identity which uses this protocol shall return the `API_INAPPROPRIATE_PROTOCOL_FOR_ACTION` return code.

A.8.8 Decipher

A request of the Decipher action on a differential-identity which uses this protocol shall return the `API_INAPPROPRIATE_PROTOCOL_FOR_ACTION` return code.

A.8.9 GetRandom

A request of the GetRandom action on a differential-identity which uses this protocol shall return the `API_INAPPROPRIATE_PROTOCOL_FOR_ACTION` return code.

A.8.10 Hash

A request of the Hash action on a differential-identity which uses this protocol shall return the `API_INAPPROPRIATE_PROTOCOL_FOR_ACTION` return code.

A.8.11 Sign

A request of the Sign action on a differential-identity which uses this protocol shall return the `API_INAPPROPRIATE_PROTOCOL_FOR_ACTION` return code.

A.8.12 VerifySignature

A request of the VerifySignature action on a differential-identity which uses this protocol shall return the `API_INAPPROPRIATE_PROTOCOL_FOR_ACTION` return code.

A.8.13 VerifyCertificate

A request of the VerifyCertificate action on a differential-identity which uses this protocol shall return the `API_INAPPROPRIATE_PROTOCOL_FOR_ACTION` return code.

A.9 PIN Compare

This authentication protocol compares a supplied PIN with a value stored in the marker of a differential-identity.

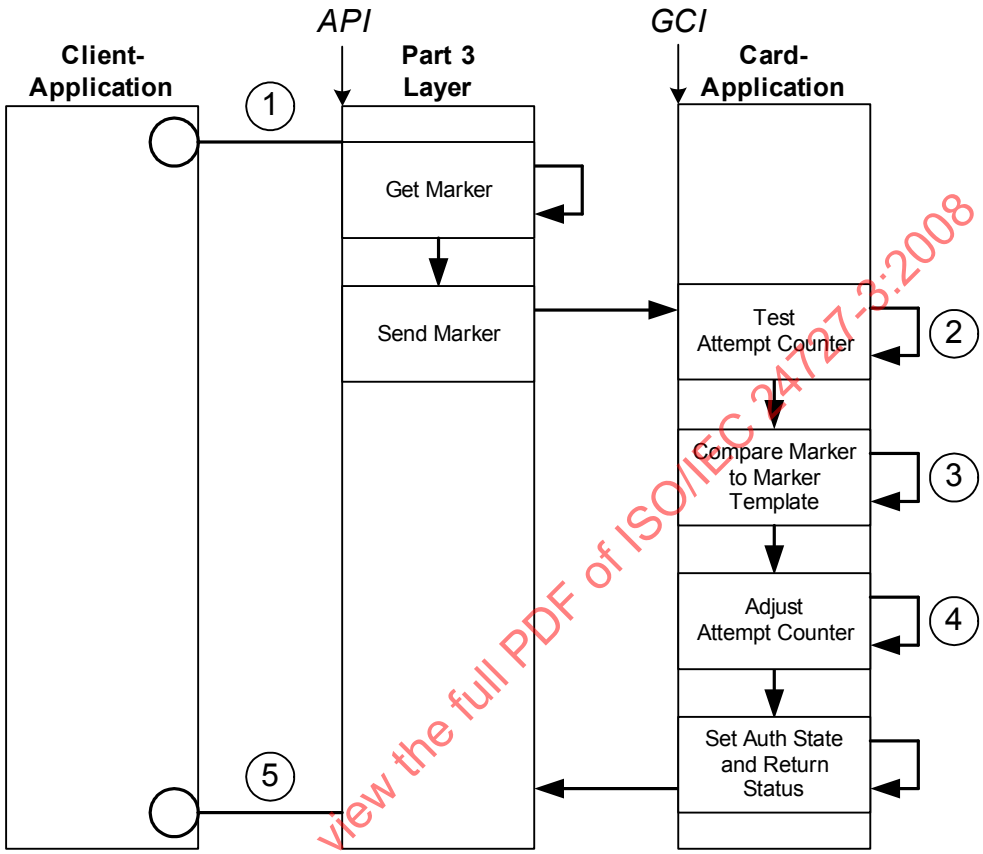

Figure A.7 — PIN Compare

The marker may be a verification code or resetting code according to ISO/IEC 7816-4.

A.9.1 Protocol Object Identifier

This protocol is identified by the OID `{ iso(1) standard(0) iso24727(24727) part3(3) annex-a(0) pin-compare(9) }`.

A.9.2 Marker

A differential-identity which uses this protocol has the following marker.

```
MarkerAP009 ::= SEQUENCE {
                        minDataLength       INTEGER,
                        maxDataLength       INTEGER,
                        storedLength        INTEGER,
                        paddingCharacter    OCTET STRING,
                        markerTemplate      OCTET STRING,
                        maxAttempts         INTEGER,
                        attemptsCounter     INTEGER,
                        pinRef              OCTET STRING,
                        pinValue            VisibleString
}
```

The minDataLength and maxDataLength parameters represent the number of bits in the key and nonce, respectively. The paddingCharacter parameter is a single byte.

The markerTemplate shall be formatted in big-endian byte order with padding of the least significant byte end.

A.9.3 DIDCreate

A request of the DIDCreate action intended to create a differential-identity for use with this authentication protocol shall include a DIDUpdateData parameter where marker is the specific marker for this protocol as defined above in A.9.2

A.9.4 DIDUpdate

A request of the DIDUpdate action involving a differential-identity which uses this authentication protocol shall include a DIDUpdateData parameter, where marker is the specific marker for this protocol as defined above in A.9.2

The performance of this action shall reset attemptsCounter to zero.

A.9.5 DIDGet

The confirmation to the DIDGet action involving a differential-identity which uses this authentication protocol shall include a didStructure parameter.

A.9.6 Authentication

This authentication protocol is executed through a single request of the DIDAuthenticate action with an **authenticationProtocolData** parameter as defined below.

A request of the CardApplicationStartSession action on a differential-identity which uses this protocol shall return the API_INAPPROPRIATE_PROTOCOL_FOR_ACTION return code.

A.9.6.1 Procedure

(1) **authenticationProtocolData** ::= markerTemplate OCTET STRING
SIZE(minDataLength..maxDataLength)

The marker shall be formatted in big-endian byte order with padding of the least significant byte end.

(2) If attemptsCounter is equal to or greater than maxAttempts, performance of this request shall proceed to to #5 below.

(3) result = (marker =?= markerTemplate)
The comparison is bit-by-bit.

(4) If result is TRUE, attemptsCounter is reset to zero. Otherwise, attemptsCounter is incremented.
retries = maxAttempts – attemptsCounter

(5) **authenticationProtocolData** ::= retries INTEGER OPTIONAL as OCTET STRING

A.9.6.2 Impact on current state

Upon successful completion of this protocol, the authentication state of the named differential-identity shall be set to the value of result within the card-application.

A.9.7 Encipher

A request of the Encipher action on a differential-identity which uses this protocol shall return the `API_INAPPROPRIATE_PROTOCOL_FOR_ACTION` return code.

A.9.8 Decipher

A request of the Decipher action on a differential-identity which uses this protocol shall return the `API_INAPPROPRIATE_PROTOCOL_FOR_ACTION` return code.

A.9.9 GetRandom

A request of the GetRandom action on a differential-identity which uses this protocol shall return the `API_INAPPROPRIATE_PROTOCOL_FOR_ACTION` return code.

A.9.10 Hash

A request of the Hash action on a differential-identity which uses this protocol shall return the `API_INAPPROPRIATE_PROTOCOL_FOR_ACTION` return code.

A.9.11 Sign

A request of the Sign action on a differential-identity which uses this protocol shall return the `API_INAPPROPRIATE_PROTOCOL_FOR_ACTION` return code.

A.9.12 VerifySignature

A request of the VerifySignature action on a differential-identity which uses this protocol shall return the `API_INAPPROPRIATE_PROTOCOL_FOR_ACTION` return code.

A.9.13 VerifyCertificate

A request of the VerifyCertificate action on a differential-identity which uses this protocol shall return the `API_INAPPROPRIATE_PROTOCOL_FOR_ACTION` return code.

A.10 Biometric Compare

This authentication protocol compares a supplied value with a value stored in the marker of a differential-identity.

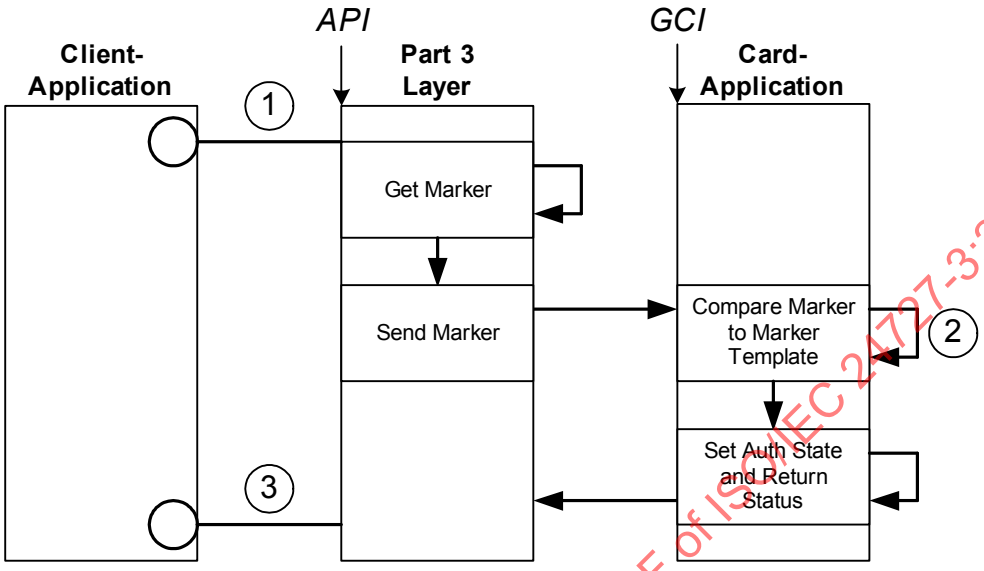

Figure A.8 — Biometric Compare

A.10.1 Protocol Object Identifier

This protocol is identified by the OID `{ iso(1) standard(0) iso24727(24727) part3(3) annex-a(0) biometric-compare(10) }`.

A.10.2 Marker

A differential-identity which uses this protocol has the following marker.

```
MarkerAP010 ::= SEQUENCE {
                    bit                 OCTET STRING,
                    markerTemplate      OCTET STRING
}
```

The `bit` parameter shall be formatted as an ISO/IEC 7816-11 biometric information template. The `markerTemplate` parameter shall be formatted as an ISO/IEC 7816-11 biometric template.

A.10.3 DIDCreate

A request of the DIDCreate action intended to create a differential-identity for use with this authentication protocol shall include a `DIDUpdateData` parameter where `marker` is the specific marker for this protocol as defined above in A.10.2

A.10.4 DIDUpdate

A request of the DIDUpdate action involving a differential-identity which uses this authentication protocol shall include a `DIDUpdateData` parameter, where `marker` is the specific marker for this protocol as defined above in A.10.2

A.10.5 DIDGet

The confirmation to the DIDGet action involving a differential-identity which uses this authentication protocol shall include a `DIDStructure` parameter

A.10.6 Authentication

This authentication protocol is executed through a single request of the DIDAuthenticate action with an **`authenticationProtocolData`** parameter as defined below.

A request of the CardApplicationStartSession action on a differential-identity which uses this protocol shall return the `API_INAPPROPRIATE_PROTOCOL_FOR_ACTION` return code.

A.10.6.1 Procedure

(1) **`authenticationProtocolData`** `::= MarkerAP010 as OCTET STRING`

(2) result = (marker =?= markerTemplate)
The comparison is as deinfed in the 7816-11 biometric information template.

(3) **`authenticationProtocolData`** `::= empty OCTET STRING`

A.10.6.2 Impact on current state

Upon successful completion of this protocol, the authentication state of the named differential-identity shall be set to the value of `result` within the card-application.

A.10.7 Encipher

A request of the Encipher action on a differential-identity which uses this protocol shall return the `API_INAPPROPRIATE_PROTOCOL_FOR_ACTION` return code.

A.10.8 Decipher

A request of the Decipher action on a differential-identity which uses this protocol shall return the `API_INAPPROPRIATE_PROTOCOL_FOR_ACTION` return code.

A.10.9 GetRandom

A request of the GetRandom action on a differential-identity which uses this protocol shall return the `API_INAPPROPRIATE_PROTOCOL_FOR_ACTION` return code.

A.10.10 Hash

A request of the Hash action on a differential-identity which uses this protocol shall return the `API_INAPPROPRIATE_PROTOCOL_FOR_ACTION` return code.

A.10.11 Sign

A request of the Sign action on a differential-identity which uses this protocol shall return the `API_INAPPROPRIATE_PROTOCOL_FOR_ACTION` return code.

A.10.12 VerifySignature

A request of the VerifySignature action on a differential-identity which uses this protocol shall return the `API_INAPPROPRIATE_PROTOCOL_FOR_ACTION` return code.

A.10.13 VerifyCertificate

A request of the VerifyCertificate action on a differential-identity which uses this protocol shall return the `API_INAPPROPRIATE_PROTOCOL_FOR_ACTION` return code.

A.11 Mutual Authentication with Key Establishment

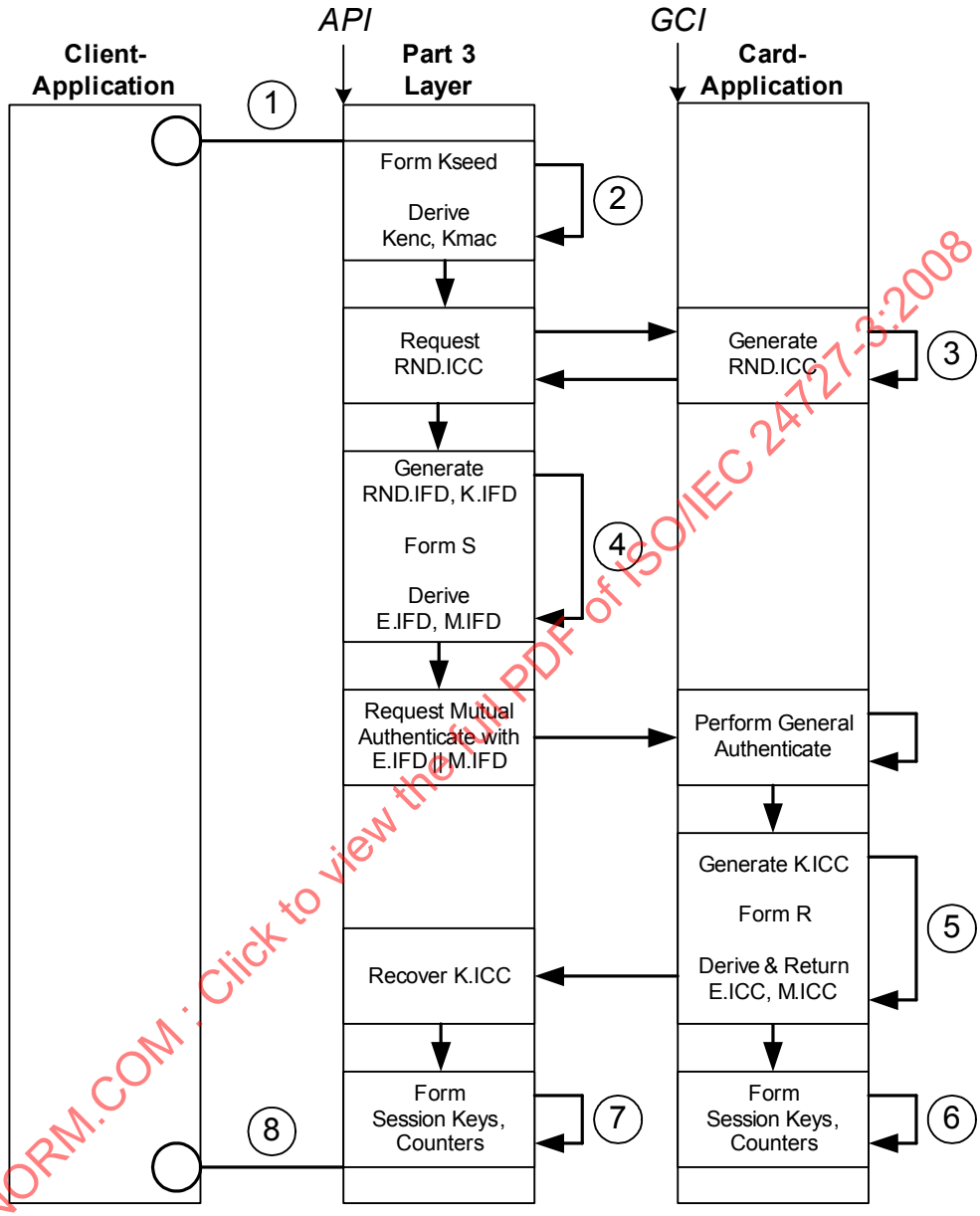

Figure A.9 — Mutual Authentication with Key Establishment

A.11.1 Protocol Object Identifier

This protocol is identified by the OID `{ iso(1) standard(0) iso24727(24727) part3(3) annex-a(0) mutual-authentication-with-key-establishment(11) }`.

A.11.2 Marker

A differential-identity which uses this protocol has the following marker.

```
MarkerAP011 ::= SEQUENCE {
          encryptionAlgorithm         OBJECT IDENTIFIER,
          macAlgorithm                OBJECT IDENTIFIER,
```

```
            derivationAlgorithmK_enc    OBJECT IDENTIFIER,
            derivationAlgorithmK_mac    OBJECT IDENTIFIER,
            derivationAlgorithmK_IFD    OBJECT IDENTIFIER,
            derivationAlgorithmSessionKeysAndCounters   OBJECT IDENTIFIER
}
```

A.11.3 DIDCreate

A request of the DIDCreate action intended to create a differential-identity for use with this authentication protocol shall include a `DIDUpdateData` parameter where `marker` is the specific marker for this protocol as defined above in A.11.2.

A.11.4 DIDUpdate

A request of the DIDUpdate action on a differential-identity which uses this protocol shall return the `API_INAPPROPRIATE_PROTOCOL_FOR_ACTION` return code.

A.11.5 DIDGet

The confirmation to the DIDGet action involving a differential-identity which uses this authentication protocol shall include a `DIDStructure` parameter.

A.11.6 Authentication

This authentication protocol is executed through a single request of the CardApplicationStartSession actions with an **`authenticationProtocolData`** parameter as defined below.

A request of the DIDAuthenticate action on a differential-identity which uses this protocol shall return the `API_INAPPROPRIATE_PROTOCOL_FOR_ACTION` return code.

A.11.6.1 Procedure

(1) **`authenticationProtocolData`** `::= K_seed OCTET STRING`

(2) `K_enc` = `derivationAlgorithmK_enc` (`Kseed`)
`K_mac` = `derivationAlgorithmK_mac` (`Kseed`)

(3) `RND.ICC` = RNG (8 bytes)

(4) `RND.IFD` = RNG (8 bytes)
`K.IFD` = RNG (16 bytes)
`S` = `RND.IFD` || `RND.ICC` || `K.IFD`
`E.IFD` = `encryptionAlgorithm` [`K_enc`] (`S`)
`M.IFD` = `macAlgorithm [K_mac]` (`E.IFD`)

(5) `M.IFD =?= macAlgorithm [K_mac] (E.IFD)`
`S'` = $\text{encryptionAlgorithm}^{-1}$ [`K_enc`] (`E.IFD`)
Recover `RND.ICC` from `S'` and compare with original.
If not equal, return some error.
Recover `K.IFD from S'`.
`K.ICC` = RNG (16 bytes)
`R` = `RND.ICC` || `RND.IFD` || `K.ICC`
`E.ICC` = `encryptionAlgorithm` [`K_enc`] (`R`)
`M.ICC` = `macAlgorithm [K_mac]` (`E.ICC`)

(6) Derive session keys and send sequence counter (SSC)
K_enc || K_mac || SSC = derivationAlgorithmSessionKeysAndCounters(K.ICC)

(7) `result = ( M.ICC =?= macAlgorithm [K_mac] (E.ICC) )`
R' = $\texttt{encryptionAlgorithm}^{-1}$ [K_enc] (E.ICC)
Recover RND.IFD from R' and compare with original.
If not equal, return some error.
Recover K.ICC from R.
K_enc || K_mac | SSC = derivationAlgorithmSessionKeysAndCounters(K.ICC)

(8) **authenticationProtocolData** ::= result BOOLEAN as OCTET STRING

A.11.6.2 Impact on current state

Upon successful completion of this protocol, the authentication state of the named differential-identity shall be set to TRUE within the card-application, and a session will be started.

A.11.7 Encipher

A request of the Encipher action on a differential-identity which uses this protocol shall return the API_INAPPROPRIATE_PROTOCOL_FOR_ACTION return code.

A.11.8 Decipher

A request of the Decipher action on a differential-identity which uses this protocol shall return the API_INAPPROPRIATE_PROTOCOL_FOR_ACTION return code.

A.11.9 GetRandom

A request of the GetRandom action on a differential-identity which uses this protocol shall return the API_INAPPROPRIATE_PROTOCOL_FOR_ACTION return code.

A.11.10 Hash

A request of the Hash action on a differential-identity which uses this protocol shall return the API_INAPPROPRIATE_PROTOCOL_FOR_ACTION return code.

A.11.11 Sign

A request of the Sign action on a differential-identity which uses this protocol shall return the API_INAPPROPRIATE_PROTOCOL_FOR_ACTION return code.

A.11.12 VerifySignature

A request of the VerifySignature action on a differential-identity which uses this protocol shall return the API_INAPPROPRIATE_PROTOCOL_FOR_ACTION return code.

A.11.13 VerifyCertificate

A request of the VerifyCertificate action on a differential-identity which uses this protocol shall return the API_INAPPROPRIATE_PROTOCOL_FOR_ACTION return code.

A.11.14 CardApplicationEndSession

A request to this action after a differential-identity utilizing this protocol has been authenticated, shall set this differential-identity authentication state to FALSE, and return the API_OK return code.

A.12 Client-Application Mutual Authentication with Key Establishment

This authentication protocol is patterned after the Mutual Authentication with Key Establishment protocol defined by CEN/EN 14890-1 and CEN/EN 14890-2.

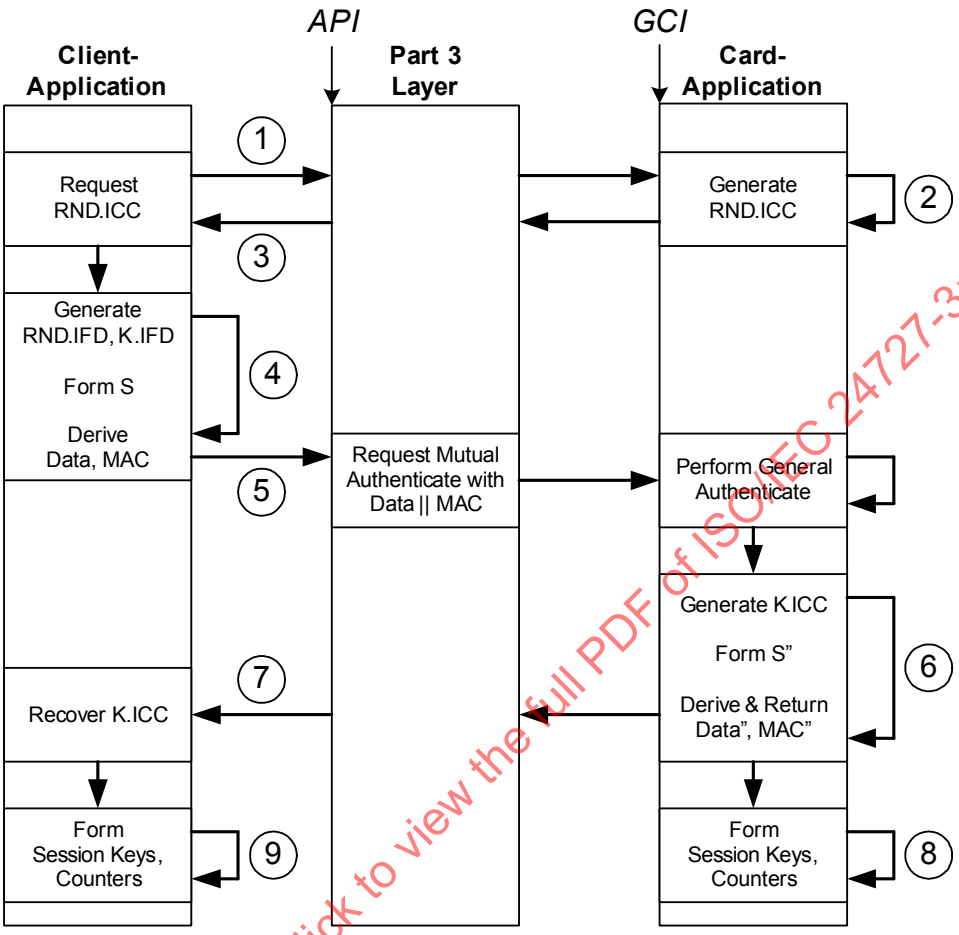

Figure A.10 — Client-Application Mutual Authentication with Key Establishment

A.12.1 Protocol Object Identifier

This protocol is identified by the OID `{ iso(1) standard(0) iso24727(24727) part3(3) annex-a(0) client-application-mutual-authentication-wke(12) }`.

A.12.2 Marker

A differential-identity which uses this protocol has the following marker.

```
MarkerAP012 ::= SEQUENCE {
                encryptionAlgorithm                          OBJECT IDENTIFIER,
                macAlgorithm                                 OBJECT IDENTIFIER,
                encryptionAlgorithmForSessionKey             OBJECT IDENTIFIER,
                macAlgorithmForSessionKey                    OBJECT IDENTIFIER,
                derivationAlgorithmSessionKeysAndCounter     OBJECT IDENTIFIER,
                K_enc                                        OCTET STRING,
                K_mac                                        OCTET STRING,
                DIV_IFD                                      OCTET STRING
}
```

The `DIV_IFD` value is 8 bytes long.

A.12.3 DIDCreate

A request of the DIDCreate action intended to create a differential-identity for use with this authentication protocol shall include a `DIDUpdateData` parameter where `marker` is the specific marker for this protocol as defined above in A.12.2.

A.12.4 DIDUpdate

A request of the DIDUpdate action on a differential-identity which uses this protocol shall include `DIDUpdateData` parameter, where `marker` is the specific marker for this protocol as defined above in A.12.2.

A.12.5 DIDGet

The confirmation to the DIDGet action involving a differential-identity which uses this authentication protocol shall include a `DIDStructure` parameter.

A.12.6 Authentication

This authentication protocol is executed through consecutive requests of the DIDAuthenticate or CardApplicationStartSession action with the same connection handle and differential-identity and with **`authenticationProtocolData`** parameters as defined below.

A.12.6.1 Prerequisite

The client-application shall obtain the 8-byte `DIV.ICC` prior to Step 1.

A.12.6.2 Procedure

(1) **`authenticationProtocolData`** `::= empty OCTET STRING`

(2) `RND.ICC` = RNG (8 bytes)

(3) **`authenticationProtocolData`** `::= RND.ICC  OCTET STRING`

(4) RND.IFD = RNG (8 bytes)
K.IFD = RNG (32 bytes)
`S = RND.IFD || SN.IFD || RND.ICC || SN.ICC || K.IFD`
Data = `encryptionAlgorithm` [`K_enc`] (`S`)
MAC = `macAlgorithm` [`K_mac`] (`Data`)

(5)
```
authenticationProtocolData ::= SEQUENCE {
                                   Data   OCTET STRING,
                                   MAC    OCTET STRING
                                   }
```

(6) `MAC =?= macAlgorithm [K_mac] (Data)`
`S'` = $\text{encryptionAlgorithm}^{-1}$ [`K_enc`] (`Data`)
Recover `RND.ICC` and `DIV.ICC` from `S'` and compare with original.
Recover `K.IFD from S'`.
`K.ICC` = RNG (32 bytes)
S" = `RND.ICC || DIV.ICC || RND.IFD || DIV.IFD || K.ICC`
`Data"` = `encryptionAlgorithm` [`K_enc`] (`S"`)
`MAC"` = `macAlgorithm [K_mac] (Data")`

(7) **authenticationProtocolData** ::= SEQUENCE {
Data" OCTET STRING,
MAC" OCTET STRING
}

(8) Compute session keys and send sequence counter (SSC) from key seed $K_{IFD/ICC}$.
K_enc || K_mac || SSC = derivationAlgorithmSessionKeysAndCounters(K.ICC)

(9) result = (MAC" =?= macAlgorithm [K_mac] (Data"))
S"' = encryptionAlgorithm^{-1} [K_enc] (Data")
Recover RND.IFD and DIV.IFD from S"' and compare with original.
Recover K.ICC from S"'.
Compute session keys and send sequence counter(SCC) from key seed $K_{IFD/ICC}$.
K_enc || K_mac || SSC = derivationAlgorithmSessionKeysAndCounters(K.ICC)

A.12.6.3 Impact on current state

Upon successful completion of this protocol, the authentication state of the named differential-identity shall be set to TRUE within the card-application, and a session will be started.

A.12.7 Encipher

A request of the Encipher action on a differential-identity which uses this protocol shall return the API_INAPPROPRIATE_PROTOCOL_FOR_ACTION return code.

A.12.8 Decipher

A request of the Decipher action on a differential-identity which uses this protocol shall return the API_INAPPROPRIATE_PROTOCOL_FOR_ACTION return code.

A.12.9 GetRandom

A request of the GetRandom action on a differential-identity which uses this protocol shall return the API_INAPPROPRIATE_PROTOCOL_FOR_ACTION return code.

A.12.10 Hash

A request of the Hash action on a differential-identity which uses this protocol shall return the API_INAPPROPRIATE_PROTOCOL_FOR_ACTION return code.

A.12.11 Sign

A request of the Sign action on a differential-identity which uses this protocol shall return the API_INAPPROPRIATE_PROTOCOL_FOR_ACTION return code.

A.12.12 VerifySignature

A request of the VerifySignature action on a differential-identity which uses this protocol shall return the API_INAPPROPRIATE_PROTOCOL_FOR_ACTION return code.

A.12.13 VerifyCertificate

A request of the VerifyCertificate action on a differential-identity which uses this protocol shall return the API_INAPPROPRIATE_PROTOCOL_FOR_ACTION return code.

A.12.14 CardApplicationEndSession

A request to this action after a differential-identity utilizing this protocol has been authenticated, shall set this differential-identity authentication state to FALSE, and return the API_OK return code.

A.13 Client-Application Asymmetric External Authenticate

This authentication protocol is used to establish an authenticated state of a differential-identity in a card-application.

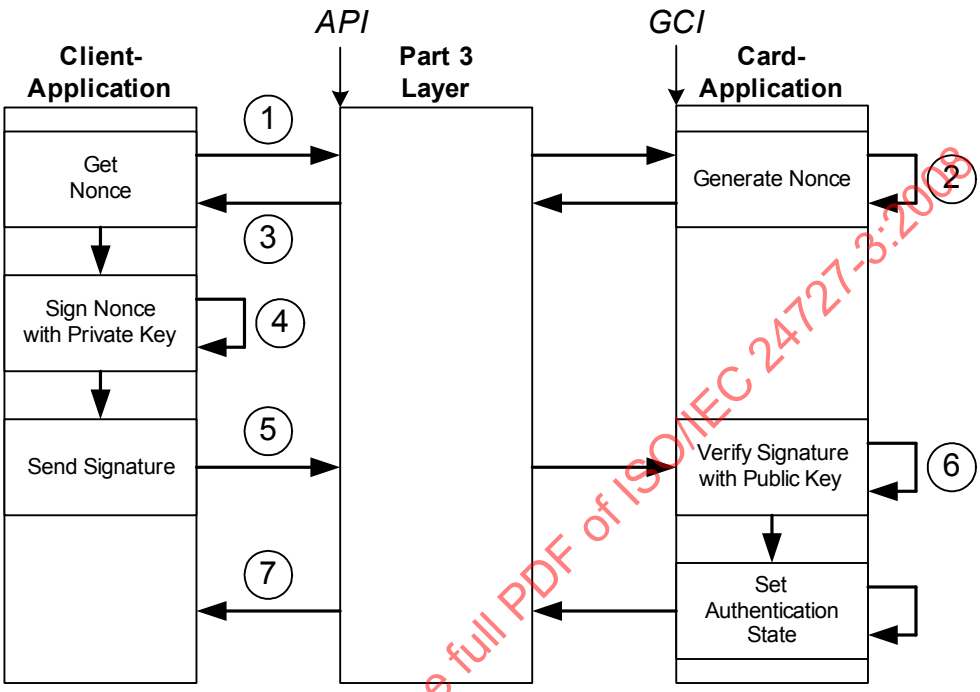

Figure A.11 — Client-Application Asymmetric External Authenticate

A.13.1 Protocol Object Identifier

This protocol is identified by the OID `{ iso(1) standard(0) iso24727(24727) part3(3) annex-a(0) client-app-asymmetric-external-authenticate(13) }`.

A.13.2 Marker

A differential-identity which uses this protocol has the following marker.

```
MarkerAP013 ::= SEQUENCE {
                        encryptionAlgorithm     OBJECT IDENTIFIER,
                        hashAlgorithm           OBJECT IDENTIFIER,
                        keySize                 INTEGER,
                        publicKeyMaterial       OCTET STRING,
                        nonceSize               INTEGER
}
```

A.13.3 DIDCreate

A request of the DIDCreate action intended to create a differential-identity for use with this authentication protocol shall include a `DIDUpdateData` parameter where `marker` is the specific marker for this protocol as defined above in A.13.2.

A.13.4 DIDUpdate

A request of the DIDUpdate action involving a differential-identity which uses this authentication protocol shall include a `DIDUpdateData` parameter, where `marker` is the specific marker for this protocol as defined above in A.13.2.

A.13.5 DIDGet

The confirmation to the DIDGet action involving a differential-identity which uses this authentication protocol shall include a `DIDStructure` parameter.

A.13.6 Authentication

This authentication protocol is executed through a single request of the DIDAuthenticate action with an **`authenticationProtocolData`** parameter as defined below.

A request of the CardApplicationStartSession action on a differential-identity which uses this protocol shall return the `API_INAPPROPRIATE_PROTOCOL_FOR_ACTION` return code.

A.13.6.1 Procedure

(1) **`authenticationProtocolData`** `::= empty OCTET STRING`

(2) `nonce` = `RNG` (`nonceSize`)

(3) **`authenticationProtocolData`** `::= nonce OCTET STRING`

(4) `signature` = `encryptionAlgorithm` [`privateKey`] (`nonce`)

(5) **`authenticationProtocolData`** `::= signature OCTET STRING`

(6) `message` = $\texttt{encryptionAlgorithm}^{-1}$ [`publicKey`] (`signature`)
`result` = (`message` =?= `nonce`)

(7) **`authenticationProtocolData`** `::= empty OCTET STRING`

A.13.6.2 Impact on current state

Upon successful completion of this protocol, if the value of `result` is TRUE, the authentication state of the named differential-identity shall be set to TRUE within the card-application.

A.13.7 Encipher

`outBuffer` = `encryptionAlgorithm` [`publicKey`] (`inBuffer`)

A.13.8 Decipher

A request of the Decipher action on a differential-identity which uses this protocol shall return the `API_INAPPROPRIATE_PROTOCOL_FOR_ACTION` return code.

A.13.9 GetRandom

`random` = `RNG` (`nonceSize`)

A.13.10 Hash

`hash` = `hashAlgorithm` (`message`)

A.13.11 Sign

A request of the Sign action on a differential-identity which uses this protocol shall return the `API_INAPPROPRIATE_PROTOCOL_FOR_ACTION` return code.

A.13.12 VerifySignature

`message` =?= `encryptionAlgorithm` [`publicKey`] (`signature`)
If TRUE, return `API_OK`, else return `API_INVALID_SIGNATURE`.

A.13.13 VerifyCertificate

Depending on certificateType,
`hash` = `hashAlgorithm` (`certificate` without signature)
`hash` =?= `encryptionAlgorithm` [`publicKey`] (signature from `certificate`)
If TRUE, return `API_OK`, else return `API_INVALID_SIGNATURE`.

A.14 Modular Extended Access Control Protocol (M-EAC)

The client application can initiate the execution of the protocol by calling Differential-IdentityAuthenticate with a DIDName, which has the Mutual Authenticate with non traceability specified as authentication protocol.

This protocol is patterned after the “Privacy constrained device authentication with non traceability based on ELC” described in prEN 14890-1 chapter 8.6.

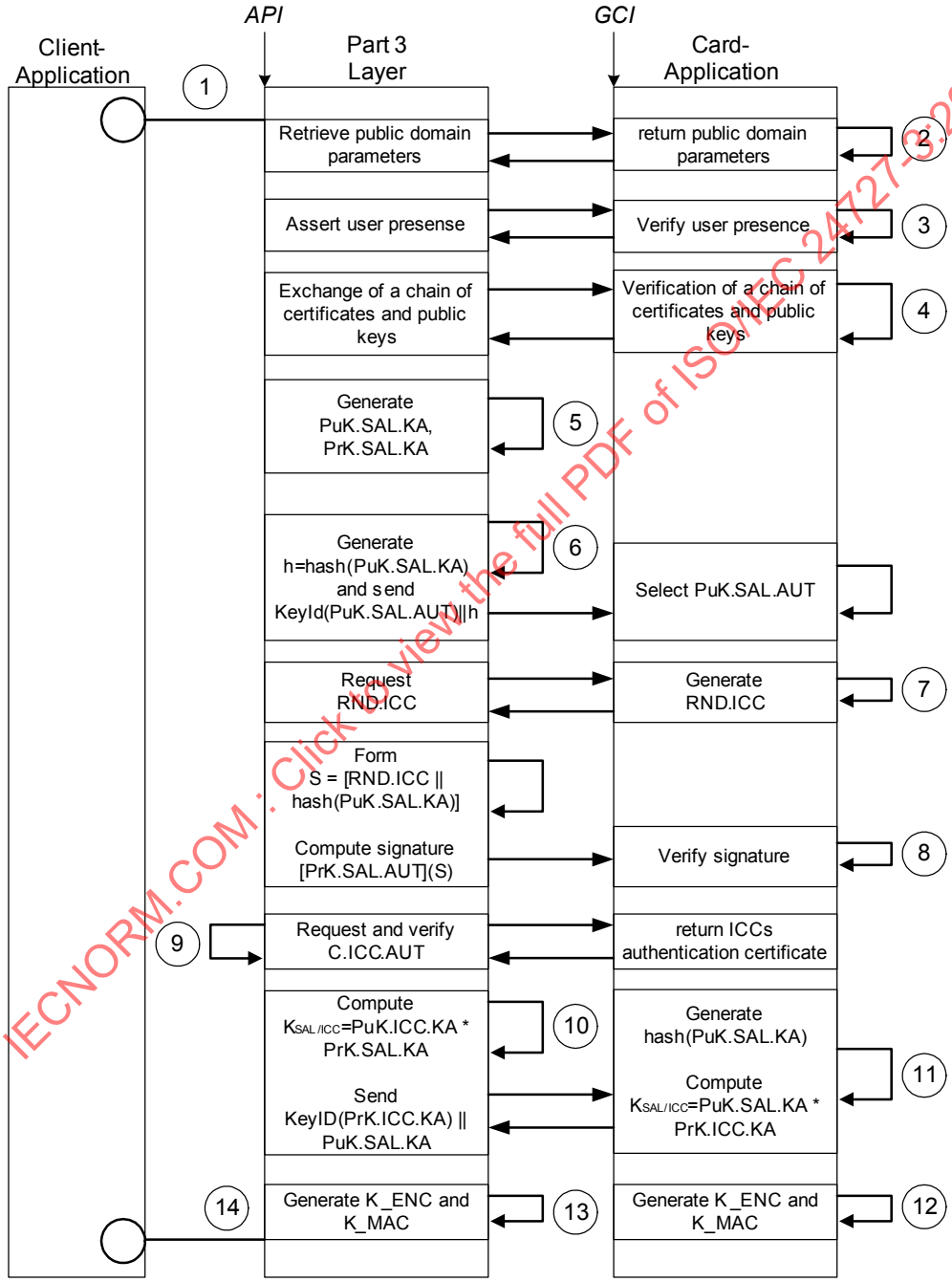

Figure A.12 — Modular Extended Access Control Protocol (M-EAC)

A.14.1 Protocol Object Identifier

This authentication protocol is identified by the OIDs:

1. { iso(1) standard(0) iso24727(24727) part3(3) annex-a(0)
   authentication-protocol(0) Modular-extended-access-control-protocol(14)
   without-local-authentication (0) }
2. { iso(1) standard(0) iso24727(24727) part3(3) annex-a(0)
   authentication- protocol(0) Modular-extended-access-control-protocol(14)
   with-non-traceability(1) }
3. { iso(1) standard(0) iso24727(24727) part3(3) annex-a(0)
   authentication-protocol(0) Modular-extended-access-control-protocol(14)
   with-local-authentication(2)}

A.14.2 Marker

A differential-identity which uses this protocol has the following marker.

```
MarkerAP014 ::= SEQUENCE {
                        encryptionAlgorithm                 OBJECT IDENTIFIER,
                        hashAlgorithm                       OBJECT IDENTIFIER,
                        encryptionAlgorithmForSessionKey    OBJECT IDENTIFIER,
                        macAlgorithmForSessionKey           OBJECT IDENTIFIER,
                        K_enc                               OCTET STRING,
                        K_mac                               OCTET STRING,
                        derivationAlgorithmK_enc            OBJECT IDENTIFIER,
                        derivationAlgorithmK_mac            OBJECT IDENTIFIER,
                        keySize                             INTEGER,
                        CHOICE {
                                SEQUENCE {
                                        publicKeyMaterial       OCTET STRING,
                                        privateKey              OCTET STRING
                                },
                                generateFlag                NULL
                        },
                        nonceSize                           INTEGER
}
```

A.14.3 DIDCreate

A request of the DIDCreate action intended to create a differential-identity for use with this authentication protocol shall include a `DIDUpdateData` parameter where `marker` is the specific marker for this protocol as defined above in A.14.2.

A.14.4 DIDUpdate

A request of the DIDUpdate action on a differential-identity which uses this protocol shall return the `API_INAPPROPRIATE_PROTOCOL_FOR_ACTION` return code.

A.14.5 DIDGet

The confirmation to the DIDGet action involving a differential-identity which uses this authentication protocol shall include a `DIDStructure` parameter.

A.14.6 Authentication

This authentication protocol is executed through a single request of the DIDAuthenticate action with an **`authenticationProtocolData`** parameter as defined below.

A request of the CardApplicationStartSession action on a differential-identity which uses this protocol shall return the `API_INAPPROPRIATE_PROTOCOL_FOR_ACTION` return code.

A.14.6.1 Procedure

(1) **`authenticationProtocolData`** `::= publicKeyMaterial  OCTET STRING`

(2) Retrieval of public domain parameters

(3) Conditional proof of user presence

--- Selection of verification key and certificate ---
(4) Verify Root CA public key and SAL's CA certificate
`verifySiganture (PuK.RCA.AUT)`
`verifyCertificate [PuK.RCA.AUT](C_CV.CA`$_{SAL}$`.CS_AUT)`

Verify signature of SAL CA authentication certificate's public key and SAL's authentication certificate
`verifySiganture (PuK.CA`$_{SAL}$`.CS_AUT)`
`verifyCertificate [PuK.CA`$_{SAL}$`.CS_AUT](C_CV.SAL.AUT)`

ICC stores `PuK.SAL.AUT` in temporary location.

--- Authenticate SAL (external authentication) ---
(5) SAL generates key agreement key pair by random
`PuK.SAL.KA = RNG (keySize)`
`PrK.SAL.KA = RNG (keySize)`

(6) `hash = hashAlgorithm (PuK.SAL.KA)`
`message = (keyId (PuK.SAL.AUT) || hash)`

(7) `RND.ICC = RNG (nonceSize)`

(8) `S = [RND.ICC || hashAlgorithm (PuK.SAL.KA)]`
`signature = encryptionAlgorithm [PrK.SAL.AUT](S)`

Verify signature
`digest = encryptionAlgorithm [PuK.SAL.AUT](signature)`
recover `RND.ICC` and compare with original

--- Authenticate ICC (internal authentication) ---
(9) Verify ICC's authentication certificate
`verifyCertificate [PuK.ICC.KA](C.ICC.AUT)`

(10) SAL computes key agreement key
$K_{SAL/ICC}$ `= [PuK.ICC.KA * PrK.SAL.KA]`
`message = (keyId(PrK.ICC.KA) || PuK.SAL.KA)`

(11) ICC computes key agreement key
`hash =?= hashAlgorithm(PuK.SAL.KA)`
$K_{ICC/SAL}$ `= [PuK.SAL.KA * PrK.ICC.KA]`

(12) ICC generates session keys for encryption and MAC calculation
`K_enc = derivationAlgorithmK_enc ()`
`K_mac = derivationAlgorithmK_mac ()`

(13) SAL generates session keys for encryption and MAC calculation
K_enc = derivationAlgorithmK_enc ()
K_mac = derivationAlgorithmK_mac ()

--- Establish secure session ---
(14) **authenticationProtocolData** ::= result BOOLEAN as OCTET STRING
}

A.14.6.2 Impact on current state

Upon successful completion of this protocol, the authentication state of the named differential-identity is not affected within the card-application.

A.14.7 Encipher

outBuffer = encryptionAlgorithm [privateKey] (inBuffer)
outBuffer = encryptionAlgorithmForSessionKey [K_enc] (inBuffer)

A.14.8 Decipher

outBuffer = encryptionAlgorithm [publicKey] (inBuffer)
outBuffer = $encryptionAlgorithmForSessionKey^{-1}$ [K_enc] (inBuffer)

A.14.9 GetRandom

random = RNG (nonceSize)

A.14.10 Hash

hash = hashAlgorithm (message)

A.14.11 Sign

signature = encryptionAlgorithm [privateKey] (message)

A.14.12 VerifySignature

message =?= encryptionAlgorithm [publicKey] (signature)
If TRUE, return API_OK, else return API_INVALID_SIGNATURE.

A.14.13 VerifyCertificate

Depending on certificate Type,
hash = hashAlgorithm (certificate without signature)
hash =?= encryptionAlgorithm [publicKey] (signature from certificate)
If TRUE, return API_OK, else return API_INVALID_SIGNATURE.

A.15 Key Transport with mutual authentication based on RSA

The client application can initiate the execution of the protocol by calling Differential-IdentityAuthenticate with a DIDName, which has the Key Transport with Mutual Authentication based on RSA specified as authentication protocol.

This protocol is patterned after the “Key transport protocol based on RSA” described in prEN 14890-1 chapter 8.4.

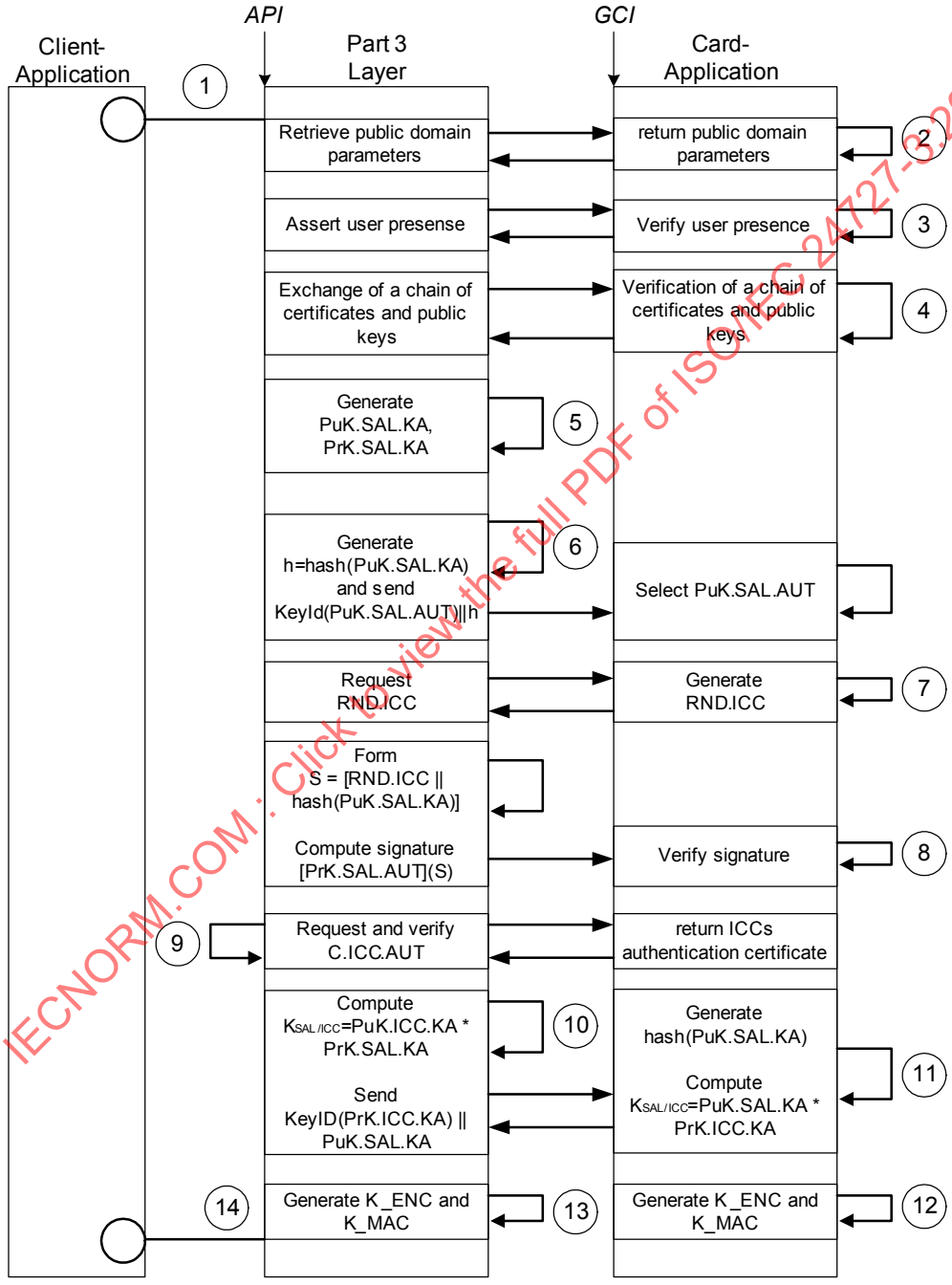

Figure A.13 — Key Transport with mutual authentication based on RSA

A.15.1 Protocol Object Identifier

This authentication protocol is identified by the OID `{ iso(1) standard(0) iso24727(24727) part3(3) annex-a(0) key-transport-with-mutual-authentication(15) }`.

A.15.2 Marker

A differential-identity which uses this protocol has the following marker.

```
MarkerAP015 ::= SEQUENCE {
                        encryptionAlgorithm       OBJECT IDENTIFIER,
                        hashAlgorithm             OBJECT IDENTIFIER,
                        keySize                   INTEGER,
                        CHOICE {
                                SEQUENCE {
                                          publicKey   OCTET STRING,
                                          privateKey  OCTET STRING
                                },
                                generateFlag        NULL
                        },
                        nonceSize                 INTEGER
}
```

A.15.3 DIDCreate

A request of the DIDCreate action intended to create a differential-identity for use with this authentication protocol shall include a `DIDUpdateData` parameter where `marker` is the specific marker for this protocol as defined above in A.15.2.

A.15.4 DIDUpdate

A request of the DIDUpdate action on a differential-identity which uses this protocol shall return the `API_INAPPROPRIATE_PROTOCOL_FOR_ACTION` return code.

A.15.5 DIDGet

The confirmation to the DIDGet action involving a differential-identity which uses this authentication protocol shall include a `DIDStructure` parameter.

A.15.6 Authentication

This authentication protocol is executed through a single request of the DIDAuthenticate action with an **`authenticationProtocolData`** parameter as defined below.

A request of the CardApplicationStartSession action on a differential-identity which uses this protocol shall return the `API_INAPPROPRIATE_PROTOCOL_FOR_ACTION` return code.

A.15.6.1 Procedure

(1) **`authenticationProtocolData`** `::= empty OCTET STRING`

(2) Multi-stage verification of public keys and certificates of SAL
(2.1) Verify CA's certificate with root CA's public key
`verifyCertificate [PuK.RCA.AUT] (C_CV.CA.CS_AUT)`

On verification success, the ICC stores the $PuK.CA_{SAL}.CS\_AUT$ contained in the certificate.

(2.2) Verify SAL's authentication certificate
verifyCertificate [$PuK.CA_{SAL}.CS\_AUT$] (C_CV.SAL.AUT)

On verification success, the ICC stores the PuK.SAL.AUT contained in the certificate in a temporary location.

(3) Multi-stage verification of public keys and certificates of ICC
SAL retrieves from ICC:
ICC's CA certificate C.CA.AUT containing CA's public key $PuK.CA_{ICC}.CS\_AUT$.
ICC's certificate C.ICC.AUT containing ICC's public key PuK.ICC.AUT used for authentication.

verifyCertificate [$PuK.CA_{ICC}.CS\_AUT$] (C.ICC.AUT)

On verification success, the SAL stores the PuK.ICC.AUT contained in the certificate temporarily.

(4) ICC to select private authentication key PrK.ICC.AUT
(5) ICC to SAL's public key PuK.SAL.AUT for encryption

--- Authenticate ICC ---
(6) SAL generates and sends
SN.SAL = serial number of SAL (8 LSB)
RND.SAL = RNG (nonceSize)
challenge = (RND.SAL || SN.SAL)

(7) ICC generates a signature and responds
K_{ICC} = ICC's key token information
signature = encryptionAlgorithm [PrK.ICC.AUT](challenge || K_{ICC})
response = encryptionAlgorithm [PuK.ICC.AUT](signature)

(8) SAL verifies signature contained in ICC's response
signature = encryptionAlgorithm [PrK.ICC.AUT](response)
result =?= encryptionAlgorithm [PuK.ICC.AUT](signature)

--- Authenticate SAL ---
(9) SAL generates signature to be verified by ICC
K_{SAL} = SAL's key token information
SN.ICC = serial number of ICC (8 LSB)
RND.ICC = RNG (nonceSize)

(10) S = (RND.ICC || SN.ICC)
signature = encryptionAlgorithm [PrK.SAL.AUT] (K_{SAL} || S)
challenge = encryptionAlgorithm [PuK.ICC.AUT] (signature)

(11) ICC verifies that K_{SAL} is different from K_{ICC} and verifies challenge sent by SAL
K_{SAL} =?= K_{ICC}
signature = encryptionAlgorithm [PrK.ICC.AUT](challenge)
result = encryptionAlgorithm [PuK.SAL.AUT](signature)

(12) **authenticationProtocolData** ::= result BOOLEAN as OCTET STRING
}

A.15.6.2 Impact on current state

Upon successful completion of this protocol, the authentication state of the named differential-identity is not affected within the card-application.

A.15.7 Encipher

`outBuffer` = `encryptionAlgorithm` [`privateKey`] (`inBuffer`)

A.15.8 Decipher

`outBuffer` = `encryptionAlgorithm` [`publicKey`] (`inBuffer`)

A.15.9 GetRandom

`random` = RNG (`nonceSize`)

A.15.10 Hash

`hash` = `hashAlgorithm` (`message`)

A.15.11 Sign

`signature` = `encryptionAlgorithm` [`privateKey`] (`message`)

A.15.12 VerifySignature

`message` =?= `encryptionAlgorithm` [`publicKey`] (`signature`)
If TRUE, return `API_OK`, else return `API_INVALID_SIGNATURE`.

A.15.13 VerifyCertificate

Depending on `certificateType`,
`hash` = `hashAlgorithm` (`certificate` without signature)
`hash` =?= `encryptionAlgorithm` [`publicKey`] (signature from `certificate`)
If TRUE, return `API_OK`, else return `API_INVALID_SIGNATURE`.

A.16 Age Attainment

The purpose of this protocol is to allow a cardholder to prove that he/she has attained a specific age, without releasing the cardholder's date of birth.

Some smart card implementations may store the cardholder date of birth. Often the date of birth is considered confidential yet there are situations where the cardholder has to prove that they have attained a specific age e.g. in order to access services that have age restrictions placed upon them. This authentication protocol is designed to indicate whether a card holder has attained a certain age without the actual date of birth being released.

This authentication protocol compares an input date to the date of birth stored in a DSI and returns success if the input date is equal to or more recent than the date of birth.

The value of the date of birth shall never be released by this authentication protocol.

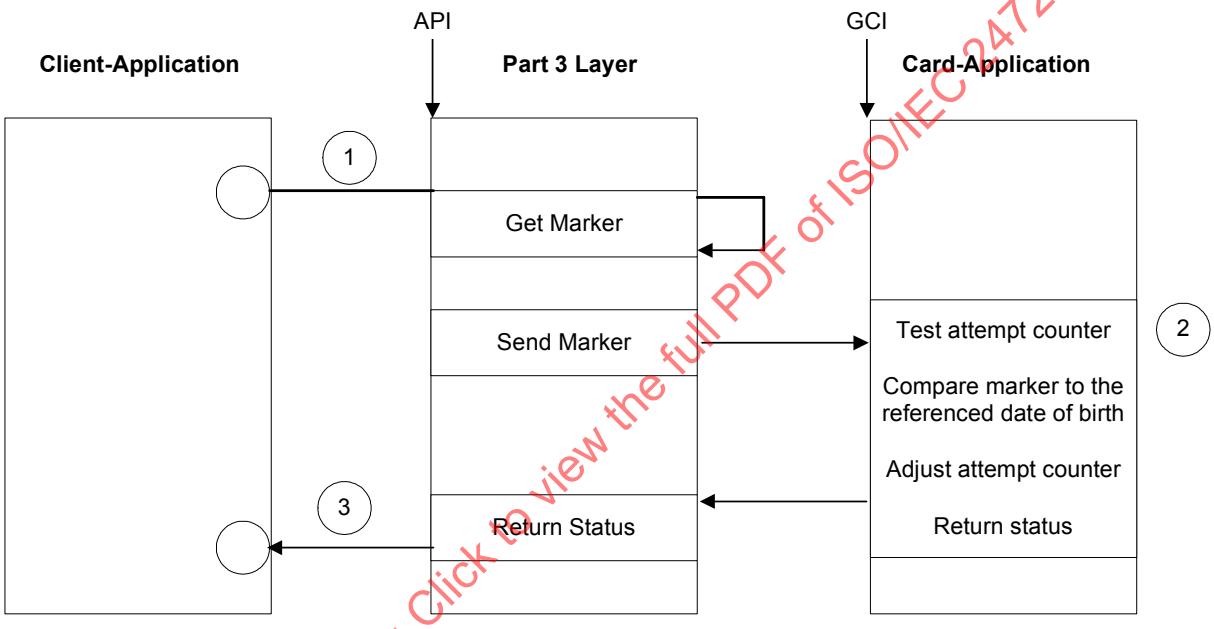

Figure A.14 — Age Attainment

A.16.1 Protocol Object Identifier

This authentication protocol is identified by the OID `{ iso(1) standard(0) iso24727(24727) part3(3) annex-a(0) age-attainment(16) }`.

A.16.2 Marker

A differential-identity which uses this protocol shall have the following marker.

```
MarkerAP016 ::= SEQUENCE {
                        dateOfBirthReference    DSIReference,
                        attainedDate            OCTET STRING
}
```

Where dateofBirthReference shall reference a date stored within the card-application. The structure of the reference is defined as follows:

```
DSIReference ::= SEQUENCE {
                              aID                    APPLICATION IDENTIFIER,
                              dataSetName            DataSetName,
                              dSIName                DSIName
}
```

The attainedDate and the date of birth stored in the DSI referenced by dateOfBirthReference shall be of the yyyymmdd format.

A.16.3 DIDCreate

A request of the DIDCreate action involving a differential-identity which uses this authentication protocol shall include a DIDUpdateData parameter where marker is the specific marker for this protocol as defined above in A.16.2.

A.16.4 DIDUpdate

A request of the DIDUpdate action on a differential-identity which uses this authentication protocol shall include a DIDUpdateData parameter, where marker is the specific marker for this protocol as defined above in A.16.2.

A.16.5 DIDGet

The confirmation to the DIDGet action involving a differential-identity which uses this authentication protocol shall include a DIDStructure parameter.

A.16.6 Authentication

This authentication protocol shall be executed through a single request of the DIDAuthenticate action with an **authenticationProtocolData** parameter as defined below.

A request of the CardApplicationStartSession action on a differential-identity which uses this protocol shall return the API_INAPPROPRIATE_PROTOCOL_FOR_ACTION return code.

A.16.6.1 Procedure

(1) **authenticationProtocolData** ::= attainedDate OCTET STRING

The attainedDate shall be formatted in the yyyymmdd format as shall be the date of birth referenced in the dateOfBirthReference.

(2) Ensure that the authentication state has not been set, as only one authentication attempt shall be permitted for each session.

Compare the attainedDate to the date stored in the DSI reference in the dateOfBirthReference field.

If the attainedDate is equal or more recent than the date of birth, return success. Otherwise, return failure.

(3) Return status.

A.16.6.2 Impact on current state

Upon completion of this authentication protocol, either successful or unsuccessful, the authentication state of the differential-identity shall be set to “true”. This shall indicate that age attainment authentication has occurred and can not be repeated for this session.

A.16.7 Encipher

A request on a differential-identity which uses this protocol shall return the `API_INAPPROPRIATE_FOR_ACTION` return code.

A.16.8 Decipher

A request on a differential-identity which uses this protocol shall return the `API_INAPPROPRIATE_FOR_ACTION` return code.

A.16.9 GetRandom

A request on a differential-identity which uses this protocol shall return the `API_INAPPROPRIATE_FOR_ACTION` return code.

A.16.10 Hash

A request on a differential-identity which uses this protocol shall return the `API_INAPPROPRIATE_FOR_ACTION` return code.

A.16.11 Sign

A request on a differential-identity which uses this protocol shall return the `API_INAPPROPRIATE_FOR_ACTION` return code.

A.16.12 VerifySignature

A request on a differential-identity which uses this protocol shall return the `API_INAPPROPRIATE_FOR_ACTION` return code.

A.16.13 VerifyCertificate

A request on a differential-identity which uses this protocol shall return the `API_INAPPROPRIATE_FOR_ACTION` return code.

A.17 Asymmetric Session Key Establishment

This protocol establishes symmetric session keys by facilitation asymmetric cryptography to transport the off-card generated session keys to the card.

An optional mechanism to either store a static asymmetric key pair (including a digital certificate) or an ephemeral asymmetric key pair provides flexibility for the issuance authority to leverage session key establishmet with or without asymmetric card authentications.

In the case of a static key pair and digital certificate, the client-application may verify the card authenticity.

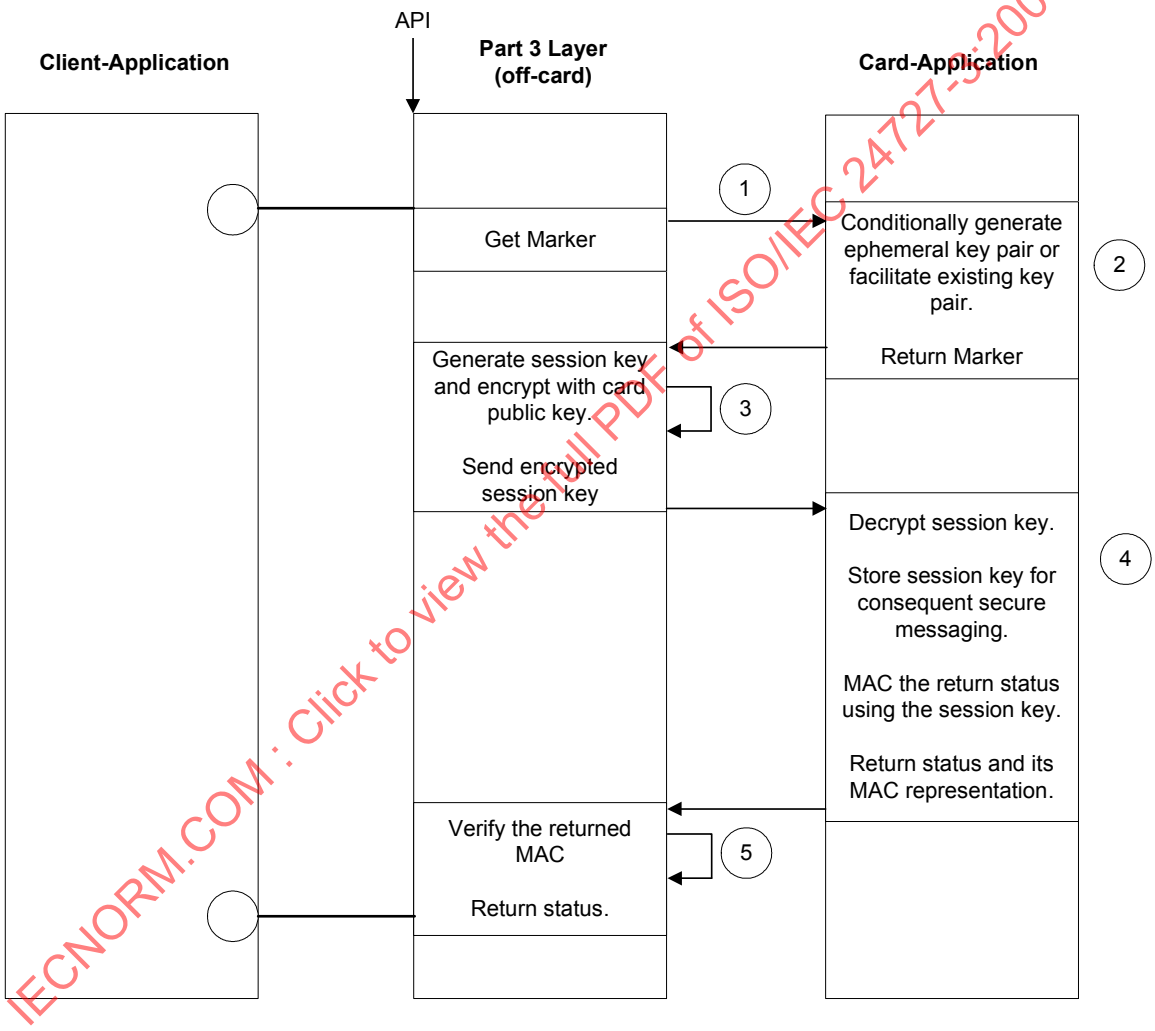

Figure A.15 — Asymmetric Session Key Establishment

A.17.1 Protocol Object Identifier

This protocol is identified by the OID `{ iso(1) standard(0) iso2747(24727) part3(3) annex-a(0) asymmetric-session-key-establishment(17) }`

A.17.2 Marker

A differential-identity which uses this protocol has the following marker.

```
MarkerAP017 ::= SEQUENCE {
                    encryptionAlgorithm         OBJECT IDENTIFIER,
                    keySize                     INTEGER,
                    keyTransportProtectionType  INTEGER,
                    privateTransKeyReference    DIDReference
                    CHOICE {
                              SEQUENCE {
                                        privateKey          OCTET STRING,
                                        publicKeyMaterial   OCTET STRING
                              },
                              SEQUENCE {
                                        privateKeyReference     DIDReference,
                                        publicKeyReference      DIDReference
                              },
                              generateFlag          BOOLEAN
                    },
                    sessionMACAlgorithm         OBJECT IDENTIFIER,
                    sessionENCAlgorithm         OBJECT IDENTIFIER
}
```

The `keyTransportProtectionType` shall indicate whether the private key is transmitted in clear text `(0)` or encrypted `(1)`. The `publicKeyMaterial` can be either a public key or a digital certificate containing a public key. Each of the parameters `privateKeyReference`, `publicKeyMaterialReference` and `privateTransKeyReference` reference a cryptographic key. The structure of the reference is as defined below:

```
DIDReference ::= SEQUENCE {
                              scope     DIDScope,
                              dIDName   DIDName
}
```

A.17.3 DIDCreate

A request of the DIDCreate action intended to create a differential-identity for use with this protocol shall include a `DIDUpdateData` parameter where `marker` is the specific marker for this protocol as defined above in A.17.2.

In the case where a private and public key pair is generated outside of the client-application and the private key is to be transported confidentially between the client-application and the card-application during the DIDCreate action (i.e. the `keyTransportProtectionType` is set to `(1)`, “encrypted”), the following apply:

— the private key shall be packaged using the secure transport syntax (ref A.17.15) format

— an existing private key shall be referenced in the `privateTransKeyReference` parameter

— the corresponding public key to the `privateTransKeyReference` shall be used during the creation of the secure transport syntax (ref A.17.15) package

— the card-application implementation of this authentication protocol shall process the secure transport syntax (ref A.17.15) package to extract the private key for storage in the DID.

If the `generateFlag` is set to `TRUE`, then the public and private key shall be generated on each DIDAuthenticate action.

If the `keyTransportProtectionType` is set to encrypted `(1)`, then the private key must be a reference to an existing key (differential-identity) available to the client-application.

A.17.4 DIDUpdate

A request of the DIDUpdate action involving a differential-identity which uses this authentication protocol shall include a `DIDUpdateData` parameter, where `marker` is the specific marker for this protocol as defined above in A.17.2.

The fields of the `marker` structure are interpreted exactly as described for the DIDCreate action, above.

A.17.5 DIDGet

A confirmation to the DIDGet action involving a differential-identity which uses this protocol shall include a `didStructure` parameter.

For this specific protocol, if the `generateFlag` is set to TRUE then no public key material or reference to a public key material shall be returned. If the `generateFlag` is not set, then the asymmetric key pair is static or referenced, and the `publicKeyMaterial` shall be returned.

A.17.6 Authentication

This protocol is executed through a single request of the CardApplicationStartSession action with an empty `authenticationProtocolData` parameter as defined below.

A request of the DIDAuthentication action on a differential-identity which uses this protocol shall return API_INAPPROPRIATE_PROTOCOL_FOR_ACTION return code.

A.17.6.1 Procedure

```
(0) authenticationProtocoldata ::= SEQUENCE {
                          certVerificationProcess   INTEGER,
                          crlAddress                OCTET STRING OPTIONAL,
                          ocspAddress               OCTET STRING OPTIONAL
    }
```

where the certVerifcationProcess parameter is an indicator consisting of a bit-mask derived from the following values:

— (0) No verification

— (1) Verify signature

— (2) Verify expiry date

— (4) Check CRL

— (8) Check OCSP responder

i.e. a verification that requires signature verification (1), expiry date verification (2), and the consultation of a OCSP responder (8) would constitute a bit-mask of 11 (1 + 2 + 8).

(1) The SAL shall request the Asymmetric Session Key Establishment marker.

(2) If the DID marker `generateFlag` is set to `TRUE`, an asymmetric key pair shall be generated and the public key shall be returned in the marker.
If the DID marker `generateFlag` is not set, the static `publicKeyMaterial`, which may be either an asymmetric public key or a digital certificate, shall be returned in the marker.

(3) The SAL may optionally verify the digital certificate if supplied, in accordance with the `certVerificationProcess` indicator.
The SAL shall generate a symmetric session key which shall be encrypted with the public key contained in the DID marker (i.e. either the public key directly if supplied, or the public key contained in the digital certificate, if supplied).

```
K_Mac = sessionMacAlgorithm()
K_Enc = sessionEncAlgorithm()

sessionKeys ::= SEQUENCE {
                              K_Mac OCTET STRING,
                              K_Enc OCTET STRING
}
```

If the marker returned contains the public key then

```
encryptedData = encryptionAlgorithm(publicKey, sessionKeys)
```

else if the marker returned contains the digital certificate

```
encryptedData = encryptionAlgorithm(digitalCertificate(publicKey),
sessionKeys)
```

The SAL shall return the encrypted session keys.

(4) The card-application decrypts the encrypted session keys using the private key and stores the session keys for subsequent ISO/IEC 24727 secure messaging.

```
sessionKeys = encryptionAlgorithm-1 (privateKey, encryptedData)
```

MAC the `status code`

```
macData = macAlgorithm(K_Mac, statusCode)
```

The card-application shall return a `status code` as well as the `macData`.

(5) The SAL verifies the returned `macData` to establish the authenticity of the referenced private key.

A.17.6.2 Impact on current state

Upon successful completion of this authentication protocol, the authentication state of the named differential-identity shall be set to TRUE within the card-application.

A.17.7 Encipher

A request action on a differential-identity which uses this protocol shall return the API_INAPPROPRIATE_PROTOCOL_FOR_ACTION return code.

A.17.8 Decipher

A request action on a differential-identity which uses this protocol shall return the API_INAPPROPRIATE_PROTOCOL_FOR_ACTION return code.

A.17.9 GetRandom

A request action on a differential-identity which uses this protocol shall return the API_INAPPROPRIATE_PROTOCOL_FOR_ACTION return code.

A.17.10 Hash

A request action on a differential-identity which uses this protocol shall return the API_INAPPROPRIATE_PROTOCOL_FOR_ACTION return code.

A.17.11 Sign

A request action on a differential-identity which uses this protocol shall return the API_INAPPROPRIATE_PROTOCOL_FOR_ACTION return code.

A.17.12 VerifySignature

A request action on a differential-identity which uses this protocol shall return the API_INAPPROPRIATE_PROTOCOL_FOR_ACTION return code.

A.17.13 VerifyCertificate

A request action on a differential-identity which uses this protocol shall return the API_INAPPROPRIATE_PROTOCOL_FOR_ACTION return code.

A.17.14 CardApplicationEndSession

A request to this action after a differential-identity utilizing this protocol has been authenticated, shall set this differential-identity authentication state to `FALSE`, and return the `API_OK` return code.

A.17.15 Secure Transport Syntax

Note: This syntax is an adaptation of PKCS#7 envelopedData content type – RFC 2315.

A.17.15.1 Transport package definition

```
ContentInfo ::= SEQUENCE {
                        contentType     ContentType,
                        content         EnvelopedData
}

ContentType ::= OBJECT IDENTIFIER

EnvelopeDataVersion ::= INTEGER (0)

CertificateSerialNumber  ::=  INTEGER

KeyEncryptionAlgorithmIdentifier ::= AlgorithmIdentifier

ContentEncryptionAlgorithmIdentifier ::= AlgorithmIdentifier

AlgorithmIdentifier ::=  SEQUENCE  {
               algorithm       OBJECT IDENTIFIER,
               parameters      ANY DEFINED BY algorithm OPTIONAL
}

EnvelopedData ::= SEQUENCE {
                        version               EnvelopeDataVersion,
                        recipientInfos        RecipientInfos,
                        encryptedContentInfo  EncryptedContentInfo
}
```

```
RecipientInfos ::= SET OF RecipientInfo

RecipientInfo ::= SEQUENCE {
        version                 EnvelopeDataVersion,
        issuerAndSerialNumber   IssuerAndSerialNumber,
        keyEncryptionAlgorithm  KeyEncryptionAlgorithmIdentifier,
        encryptedKey            EncryptedKey
}

IssuerAndSerialNumber ::= SEQUENCE {
issuer                  Name,
serialNumber            CertificateSerialNumber
}

EncryptedKey ::= OCTET STRING

EncryptedContentInfo ::= SEQUENCE {
   contentType                  ContentType,
   contentEncryptionAlgorithm   ContentEncryptionAlgorithmIdentifier,
   encryptedContent             [0] IMPLICIT EncryptedContent OPTIONAL
}

EncryptedContent ::= OCTET STRING
```

A.17.15.2 Descriptions

ContentInfo – contentType: indicates the type of content. This document defines one content type of envelopedData that shall have an object identifier of OID { iso(1) standard(0) iso2747(24727) part3(3) annex-a(0) asymmetric-session-key-establishment(17) envelopedData (1)}.

ContentInfo – content: the content to be transmitted. In this case it shall be the envelopedData.

EnvelopedData – EnvelopeDataVersion: is the syntax version number. It shall be 0 for this version of the enveloped data format.

EnvelopedData – recipientInfos: is a collection of per-recipient information. There must be at least one element in the collection.

EnvelopedData – encryptedContentInfo: is the encrypted content information.

RecipientInfo – version: is the syntax version number. It shall be 0 for this version of the document.

RecipientInfo – issuerAndSerialNumber specifies the recipient's certificate (and thereby the recipient's distinguished name and public key) by issuer distinguished name and issuer-specific serial number.

RecipientInfo – keyEncryptionAlgorithm: identifies the key-encryption algorithm (and any associated parameters) under which the content-encryption key is encrypted with the recipient's public key.

RecipientInfo – encryptedKey is the result of encrypting the content-encryption key with the recipient's public key.

EncryptedContentInfo – contentType: indicates the type of content. This document defines one content type of envelopedData that shall have an object identifier of OID { iso(1) standard(0) iso2747(24727) part3(3) annex-a(0) asymmetric-session-key-establishment(17) envelopedData (1) }.

EncryptedContentInfo – contentEncryptionAlgorithm: identifies the content-encryption algorithm (and any associated parameters) under which the content is encrypted.

EncryptedContentInfo – encryptedContent: is the result of encrypting the content.

A.17.15.3 Package Construction Process

A content-encryption key for a particular content-encryption algorithm is generated at random.

Encrypt the content encryption key with the recipient (i.e. client-application) public key.

Collate the encrypted content encryption key and other receipient specific information into the RecipientInfo value.

Encrypt the content with the encryption key (apply padding if required).

Collate the RecipientInfo values with the encrypted content into an EnvelopedData value.

A.17.15.4 Package Interpretation Process

Decrypt the encrypted content encryption key with the recipients (i.e. smartcard) private key.

Decrypt the encrypted content with the recovered content encryption key.

A.18 Secure PIN Compare

This authentication protocol utilizes secure transport syntax (ref A.17.14) to securely and anonymously transmit the PIN. The card-application shall extract the PIN from the secure transport syntax (ref A.17.14) package and compares it with a value stored in the marker of a differential-identity.

The attemptsCounter value in this authentication protocol shall be updatable in order to facilitate the locking of the authentication protocol (i.e. set attemptsCounter to maxAttempts).

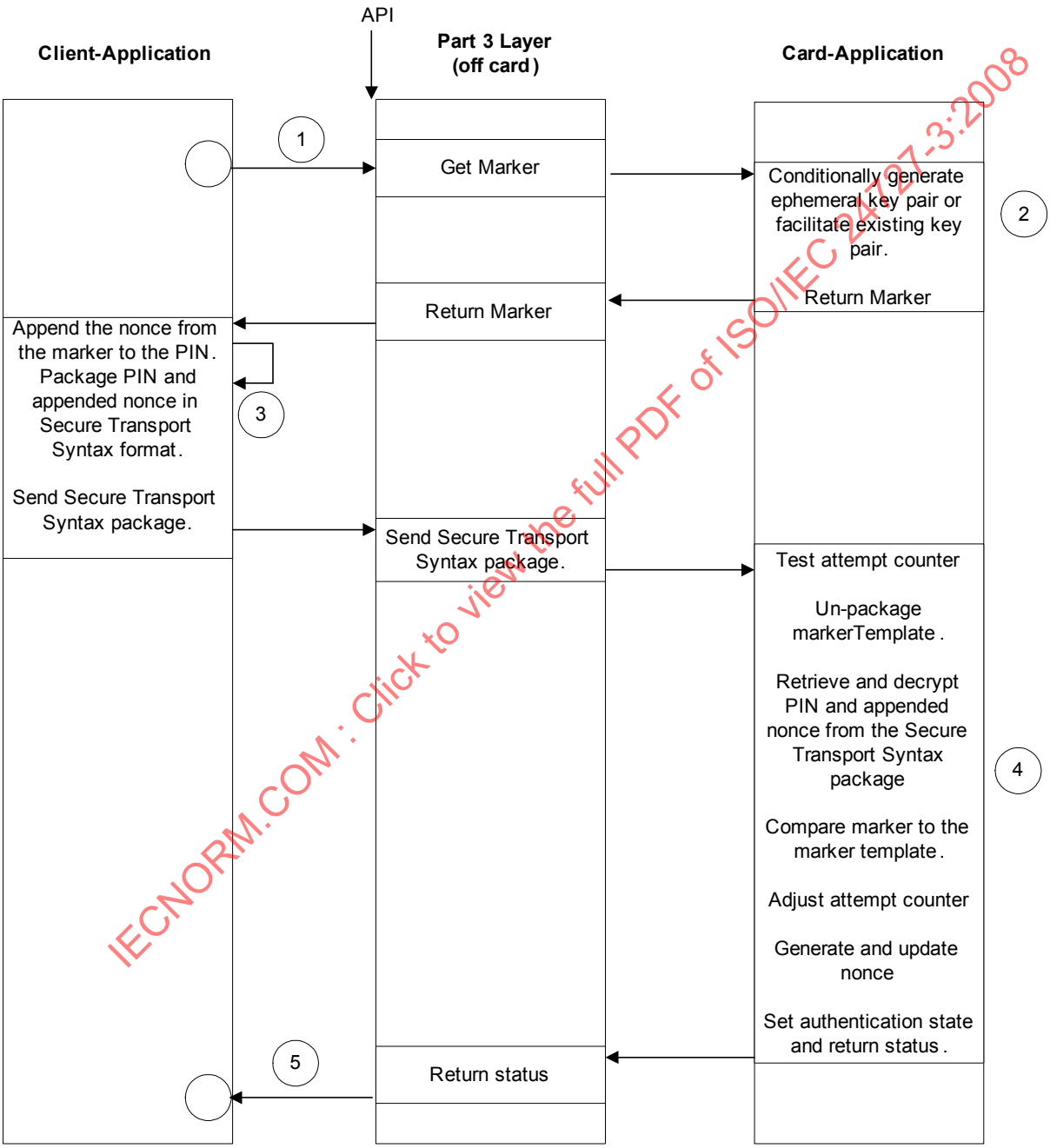

Figure A.16 — Secure PIN Compare

A.18.1 Protocol Object Identifier

This protocol is identified by the OID `{ iso(1) standard(0) iso24727(24727) part3(3) annex-a(0) secure-pin-compare(18) }`

A.18.2 Marker

A differential-identity which uses this protocol has the following marker:

```
markerAP018 ::= SEQUENCE {
                    minDataLength                       INTEGER,
                    maxDataLength                       INTEGER,
                    paddingCharacter                    OCTET STRING,
                    markerTemplate                      OCTET STRING,
                    maxAttempts                         INTEGER,
                    attemptsCounter                     INTEGER,
                    encryptionAlgorithm                 OBJECT IDENTIFIER,
                    keySize                             INTEGER,
                    keyTransportProtectionType          INTEGER,
                    nonceSize                           INTEGER,
                    nonce                               INTEGER,
                    CHOICE {
                            SEQUENCE {
                                    privateKey          OCTET STRING,
                                    publicKeyMaterial   OCTET STRING
                            },
                            SEQUENCE {
                                    privateKeyReference         DIDReference,
                                    publicKeyMaterialReference DIDReference
                            },
                            generateFlag NULL
                    }
}
```

The `keyTransportProtectionType` shall indicate whether the `markerTemplate` is transmitted in clear text(0) or encrypted(1) during the DIDCreate or DIDUpdate action.

The `privateKeyReference` and `publicKeyMaterialReference` parameters shall reference a cryptographic key known through a differential-identity. The structure of the reference is defined below:

```
DIDReference ::= SEQUENCE {
                            scope    DIDScope,
                            dIDName  DIDName
}
```

A.18.3 DIDCreate

A request of the DIDUpdate action involving a differential-identity which uses this authentication protocol shall include a `DIDUpdateData` parameter where `marker` is the specific marker for this protocol as defined above in A.18.2.

In the case where the `markerTemplate` is to be transported confidentially between the client-application and card-application during the DIDCreate action (i.e. the `keyTransportProtectionType` is set to “encrypted”), the following apply:

— The `markerTemplate` shall be packaged using the secure transport syntax (ref A.17.14) format;

— A static private and public key shall be referenced in the `privateKeyReference` parameter and the `publicKeyMaterialReference` respectively;

— The public key referenced by `publicKeyMaterialReference` shall be used during the creation of the secure transport syntax (ref A.17.14) package.

— The card-application implementation of this authentication protocol shall process the secure transport syntax (ref A.17.14) package to extract the `markerTemplate` for storage in the DID.

In the case where the `generateFlag` is set to "true" the `markerTemplate` shall not be transported confidentially and no secure packaging is required during the DIDCreate action.

A.18.4 DIDUpdate

A request of the DIDUpdate action involving a differential-identity which uses this authentication protocol shall include a `DIDUpdateData` parameter, where `marker` is the specific marker for this protocol as defined above in A.18.2. The fields of the `marker` structure are interpreted exactly as described for the DIDCreate action, above.

The `attemptsCounter` counter may be set to `maxAttempts` to disable a differential-identity which utilizes this authentication protocol.

A.18.5 DIDGet

A request of the DIDGet action on a differential-identity which uses this protocol shall include a `DIDStructure` parameter.

A.18.6 Authentication

This authentication protocol shall be executed through two requests. The first request is the DIDGet action which returns the marker containing the required public key. The second request is the DIDAuthenticate action with an `authenticationProtocolData` parameter as defined below.

A request of the CardApplicationStartSession() action on a differential-identity which uses this protocol shall return the API_INAPPROPRIATE_PROTOCOOL_FOR_ACTION return code.

A.18.6.1 Procedure

(1) Request to retrieve the marker that contains the public key.

(2) If the DID marker `generateFlag` is set to TRUE, an asymmetric key pair shall be generated by the card-application for the current session.
if the DID marker generateFlag is set to FALSE, the referenced key pair shall be utilized by the card-application for the current session.
The marker including the public key material and current nonce shall be returned to the client-application.

(3) Verify that the `attemptsCounter` is less than the `maxAttempts`. The client-application shall generate a symmetric key and package the PIN appended with the nonce using the generated symmetric key and the card-application supplied public key as specified by the secure transport syntax defined in A.17.14.

The `markerTemplate`, with the contained secure transport syntax (ref. A.17.14) package, is returned to the card-application.

```
authenticationProtocolData ::= ContentInfo(secure transport syntax A.17.14, 7.
General syntax)
```

(4) If the value of `attemptsCounter` is greater or equal to the value of `maxAttempts`, this request shall proceed with step #5.
The card-application unwraps the secure transport syntax (ref. A.17.14) template using the `privateKey` to decrypt the `encryptedKey` contained in the `ContentInfo` of the secure transport syntax template.

The `encryptedKey` in conjunction with other `ContentInfo` is used to decrypt the PIN and the appended `nonce`.

```
result = (decryptedPIN =?= markerTemplate)
```

The PIN and appended nonce comparison is bit-by-bit.

If result is TRUE, `attemptsCounter` is reset to zero. Otherwise, `attemptsCounter` is incremented.

Generate and store the new `nonce`.

```
nonce = RNG(nonceSize)
```

(5)
```
authenticationProtocolData ::= SEQUENCE {
                                    maxAttempts       INTEGER OPTIONAL,
                                    attemptsCounter   INTEGER OPTIONAL
                                    }
```

A.18.6.2 Impact on current state

Upon successful completion of this protocol, the authentication state of the named differential-identity shall be set to the value of result within the card-application.

A.18.7 Encipher

A request action on a differential-identity which uses this protocol shall return the API_INAPPROPRIATE_PROTOCOL_FOR_ACTION return code.

A.18.8 Decipher

A request action on a differential-identity which uses this protocol shall return the API_INAPPROPRIATE_PROTOCOL_FOR_ACTION return code.

A.18.9 GetRandom

A request action on a differential-identity which uses this protocol shall return the API_INAPPROPRIATE_PROTOCOL_FOR_ACTION return code.

A.18.10 Hash

A request action on a differential-identity which uses this protocol shall return the API_INAPPROPRIATE_PROTOCOL_FOR_ACTION return code.

A.18.11 Sign

A request action on a differential-identity which uses this protocol shall return the API_INAPPROPRIATE_PROTOCOL_FOR_ACTION return code.

A.18.12 VerifySignature

A request action on a differential-identity which uses this protocol shall return the API_INAPPROPRIATE_PROTOCOL_FOR_ACTION return code.

A.18.13 VerifyCertificate

A request action on a differential-identity which uses this protocol shall return the API_INAPPROPRIATE_PROTOCOL_FOR_ACTION return code.

A.19 EC Key Agreement with Card-Application Authentication

This authentication protocol is an card-application internal authentication protocol and key agreement using EC cryptography. The initator generates an empemeral key pair but has not static key pair; the responder has only a static key pair. This protocol is used to establish authentication (SKmac) and encryption (SKenc) session keys for further secure messaging between the ISO/IEC 24727-3 layer and the card-application.

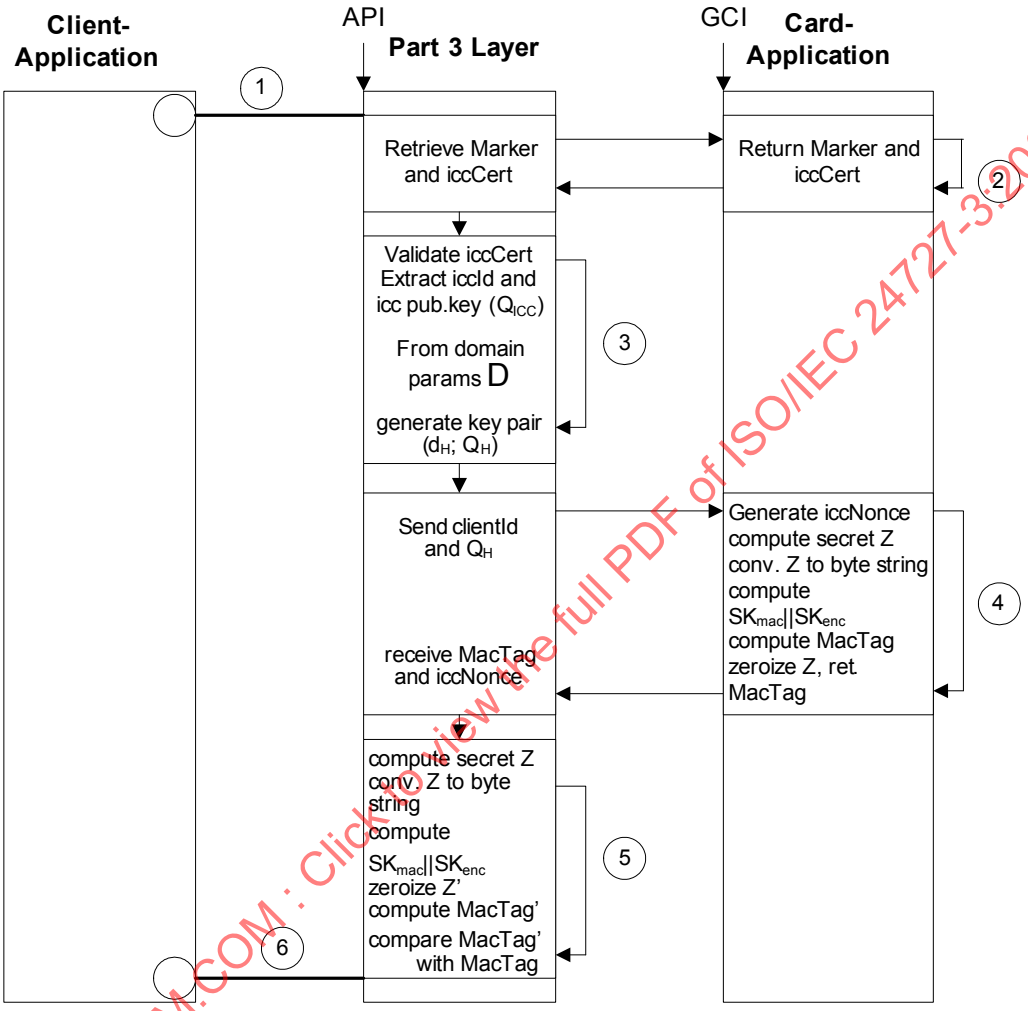

Figure A.17 — EC Key Agreement with Card-Application Authentication

A.19.1 Protocol Object Identifier

This protocol is identified by the OID `{ iso(1) standard(0) iso24727(24727) part3(3) annex-a(0) ec-key-agreement-with-card-application-authentication(19) }`

A.19.2 Marker

A differential-identity which uses this protocol has the following marker:

```
MarkerAP019 ::= SEQUENCE {
                        domainParameters            OBJECT IDENTIFIER,
                        keyEstablishmentAlgorithm   OBJECT IDENTIFIER,
                        kDFHashAlgorithm            OBJECT IDENTIFIER,
                        sessionMacAlgorithm         OBJECT IDENTIFIER,
                        sessionEncAlgorithm         OBJECT IDENTIFIER,
                        nonceSize                   INTEGER,
                        CHOICE {
                                SEQUENCE {
                                          iccPublicKey      OCTET STRING,
                                          iccPrivateKey     OCTET STRING
                                },
                                genKeyPairFlag NULL
                        }
                        iccIdentifier               OCTET STRING,
                        iccCert                     OCTET STRING
}
```

A.19.3 DIDCreate

A request of the DIDCreate action intended to create a differential-identity for use with this protocol shall include a `DIDUpdateData` parameter where `marker` is the specific marker for this protocol as defined above in A.19.2.

A.19.4 DIDUpdate

A request of the DIDUpdate action involving a differential-identity which uses this protocol shall include a `DIDUpdateData` parameter where `marker` is the specific marker for this protocol as defined above in A.19.2.

A.19.5 DIDGet

A confirmation to the DIDGet action involving a differential-identity which uses this protocol shall include a `DIDStructure` parameter.

A.19.6 Authentication

This protocol is executed through a simple request of the CardApplicationStartSession action with an `authenticationProtocolData` parameter as defined below.

A.19.6.1 Procedure

(1) `authenticationProtocolData ::= clientIdentifier OCTET STRING`

(2) Validate `iccCert`. Generate ephemeral key pair (d_H;Q_H) from domain parameters D. Verify that Q_H is valid for Domain Parameters D.

(3) `iccNonce = RNG(nonceSize)`
`Z = keyEstablishmentAlgorithm(`d_{ICC}`;`Q_H`)`

Convert `Z` to a byte string using field-element-to-byte-string.

Compute session keys using the derivation function:

SK_{mac} || SK_{enc} = KDF(Z, keyDataLen, otherInfo)

Where:

```
otherInfo ::= SEQUENCE {
      sessionMacAlgorithm       OBJECT IDENTIFIER,
      sessionEncAlgorithm       OBJECT IDENTIFIER,
      clientIdentifier          OCTET STRING,
      iccIdentifier             OCTET STRING,
      iccNonce                  OCTET STRING
}

keyDataLen = algoKeyLengthInBits(seesionMacAlgorithm) +
               algoKeyLengthInBits(sessionEncAlgorithm)
N = keyDataLen / (hashLengthInBits(kDFHashAlgorithm))
DerivedKeyingMaterial = kDFHashAlgorithm(0x00000001 || Z || otherInfo)  ||
                        kDFHashAlgorithm(0x00000002 || Z || otherInfo)  ||
                        kDFHashAlgorithm((0x00000000 + N) || Z || otherInfo)
```

KDF is the (keyDataLen) left-most bits from DerivedKeyingMaterial.

The coordinates of Q_H are converted from field elements to byte strings Q_H'

```
MacData = "KC_1_V" || iccIdentifier || clientIdentifier || NULL || QH'
MacTag = sessionMacAlgorithm(SKmac, MacData)
```

(4) Verify that Q_{ICC} is valid for Domain Parameters D

Z = keyEstablishmentAlgorithm(d_H;Q_{ICC})

Convert Z to a byte string using field-element-to-byte-string.

Use KDF defined at step (3)

SK_{mac} || SK_{enc} = KDF(Z; KeyDataLen; otherInfo)

The coordinates of QH are converted from field elements to byte strings QH'

```
MacData' = "KC_1_V" || iccIdentifier || clientIdentifier || NULL || QH'
MacTag' = sessionMacAlgorithm(SKmac, MadData)
```

(5) authenticationProtocol ::= empty OCTET STRING

return_code = API_OK

A.19.6.2 Impact on current state

Upon successful completion of this authentication protocol, the authentication state of the named differential-identity is not affected within the card-application, however, a session with secure messaging capability is started.

A.19.7 Encipher

A request action on a differential-identity which uses this protocol shall return the API_INAPPROPRIATE_PROTOCOL_FOR_ACTION return code.

A.19.8 Decipher

A request action on a differential-identity which uses this protocol shall return the API_INAPPROPRIATE_PROTOCOL_FOR_ACTION return code.

A.19.9 GetRandom

A request action on a differential-identity which uses this protocol shall return the API_INAPPROPRIATE_PROTOCOL_FOR_ACTION return code.

A.19.10 Hash

A request action on a differential-identity which uses this protocol shall return the API_INAPPROPRIATE_PROTOCOL_FOR_ACTION return code.

A.19.11 Sign

A request action on a differential-identity which uses this protocol shall return the API_INAPPROPRIATE_PROTOCOL_FOR_ACTION return code.

A.19.12 VerifySignature

A request action on a differential-identity which uses this protocol shall return the API_INAPPROPRIATE_PROTOCOL_FOR_ACTION return code.

A.19.13 VerifyCertificate

A request action on a differential-identity which uses this protocol shall return the API_INAPPROPRIATE_PROTOCOL_FOR_ACTION return code.

A.20 EC Key Agreement with Mutual Authentication

This authentication protocol is a mutual authentication protocol with key agreement using Eliptic Curve cryptography.

For forward secrecy, both client and ICC use an ephermeral EC key pair and a static EC key pair to establish the session keys.

This protocol is used to establish authentication (SKmac) and encryption (SKenc) session keys for further secure messaging between the ISO/IEC 24727-3 layer and the card-application.

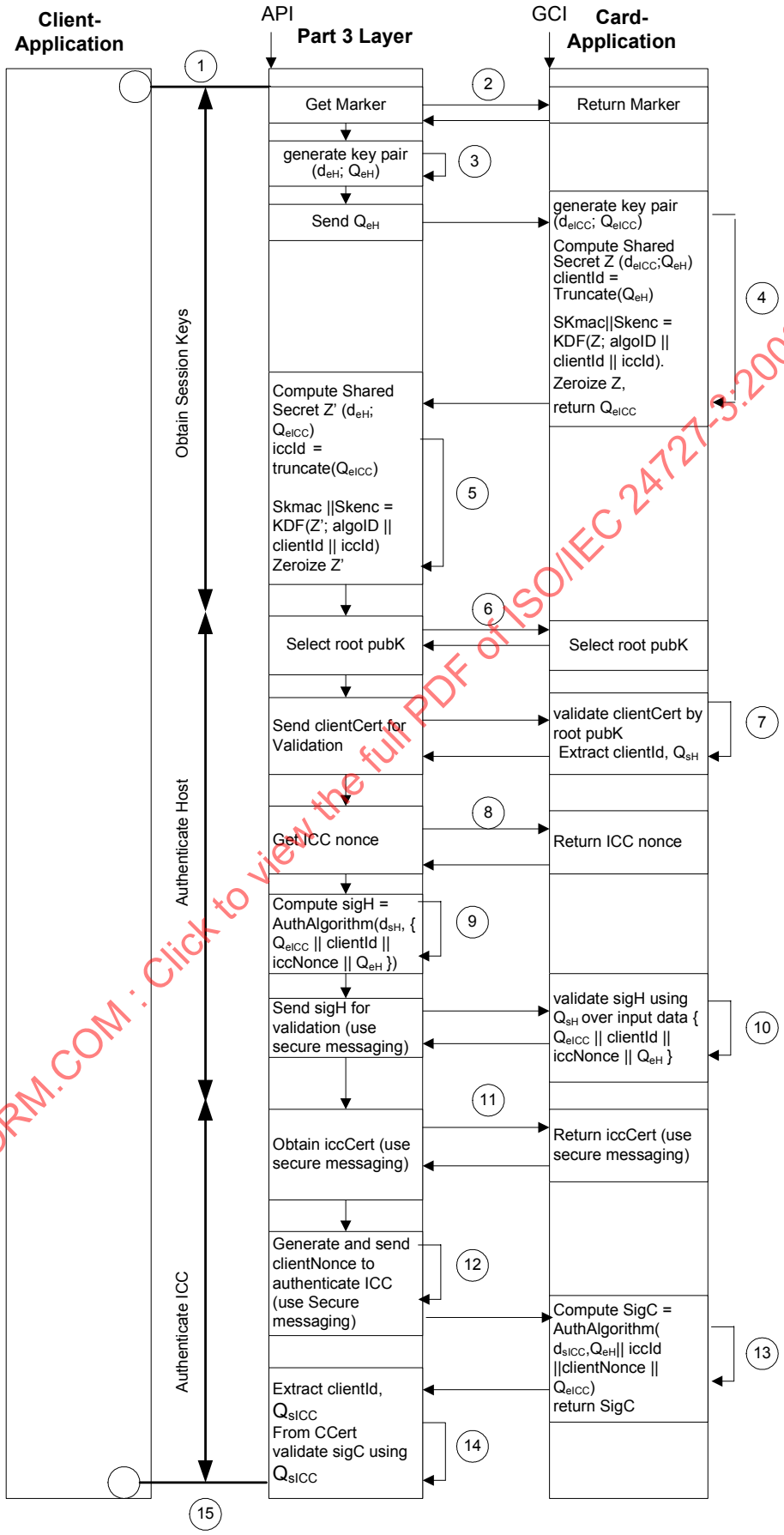

Figure A.18 — EC Key Agreement with Mutual Authentication

A.20.1 Protocol Object Identifier

This protocol is identified by the OID `{ iso(1) standard(0) iso24727(24727) part3(3) annex-a(0) ec-key-agreement-with-mutual-authentication(20) }`

A.20.2 Marker

A differential-identity which uses this protocol has the following marker:

```
MarkerAP020 ::= SEQUENCE {
            DomainParameters                OBJECT IDENTIFIER,
            KeyEstablishmentAlgorithm       OBJECT IDENTIFIER,
            KDFHashAlgorithm                OBJECT IDENTIFIER,
            SessionMacAlgorithm             OBJECT IDENTIFIER,
            SessionEncAlgorithm             OBJECT IDENTIFIER,
            AuthAlgorithm                   OBJECT IDENTIFIER,
            nonceSize                       INTEGER,
            CHOICE {
                    SEQUENCE {
                                iccPublicKey    OCTET STRING,
                                iccPrivateKey   OCTET STRING
                    },
                    genKeyPairFlag          NULL
            },
            iccIdentifier           OCTET STRING,
            iccCert                 OCTET STRING,
            rootIdentifier          OCTET STRING,
            rootPublicKey           OCTET STRING,
            iccCertKnown            BOOLEAN,
            verifyClientCert        BOOLEAN,
            EnforcePrivacy          BOOLEAN
}
```

A.20.3 DIDCreate

A request of the DIDCreate action intended to create a differential-identity for use with this protocol shall include a `DIDUpdateData` parameter where `marker` is the specific marker for this protocol as defined above in A.20.2

A.20.4 DIDUpdate

A request of the DIDUpdate action involving a differential-identity which uses this protocol shall include a `DIDUpdateData` parameter, where `marker` is the specific marker for this protocol as defined above in A.20.2.

A.20.5 DIDGet

A confirmation to the DIDGet action involving a differential-identity which uses this protocol shall include a `DIDStructure` parameter.

A.20.6 Authentication

This protocol is executed through a simple request of the CardApplicationStartSession() action with the `authenticationProtocolData` parameter defined below.

A.20.6.1 Procedure

(1) authenticationProtocolData ::= SEQUENCE {
clientIdentifier OCTET STRING
}

(2) Obtain Domain Pa Parameters D from Marker template. The returned information must not include any data specific to the ICC. All keys, certificates and identifiers must be set to NULL.

(3) Generate ephemeral key pair(d_{eH}, Q_{eH}) from domain parameters D Send Q_{eH} to ICC.

(4) Verify that Q_{eH} is valid for Domain Parameters D Generate ephemeral key pair (d_{eICC},Q_{eICC}) from domain parameters D.

$Z = KeyEstablishmentAlgorithm(d_{eICC}, Q_{eH})$

Convert Z to a byte string using Field-element-to-Byte-String

Convert the coordinates of Q_{eICC} and Q_{eH} from field elements to byte strings Q_{ICC}' and Q_H'.

clientIdentifier = Truncate(QICC')
iccIdentifier = Truncate(QH')

Where Truncate() returns the left-most 4 bytes of its parameter.

Compute session keys using the derivation function KDF:

SKmac || SKenc = KDF(Z, keyDataLen, otherInfo)

Where

otherInfo ::= SEQUENCE {

sessionMacAlgorithm	OBJECT IDENTIFIER,
sessionMacAlgorithm	OBJECT IDENTIFIER,
clientIdentifier	OBJECT IDENTIFIER,
iccIdentifier	OBJECT IDENTIFIER

}

keyDataLen = algoKeyLengthInBits(sessionMacAlgorithm)
+ algokeyLengthInBits(sessionEncAlgorithm)

N = keyDataLen / (hashLengthInBits(KDFHashAlgorithm))

DerivedKeyingMaterial = KDFHashAlgorithm(0x00000001 || Z || otherInfo) ||
KDFHashAlgorithm(0x00000002 || Z || otherInfo) ||
…
KDFHashAlgorithm((0x0000000 + N) || Z || otherInfo)

KDF is the keyDataLen left-most bits from DerivedKeyingMaterial.

Zeroize Z and return Q_{eICC}.

(5) Verify that Q_{eICC} is valid for domain parameters D.

$Z = KeyEstablishmentAlgorithm(D_{eH};Q_{eICC})$

Convert Z to a byte string using Field-element-to-Byte-String

Convert the coordinates of Q_{eICC} and Q_{eH} from field elements to byte strings Q_{ICC}' and Q_H'.

```
clientIdentifier = Truncate(Q_ICC')
iccIdentifier = Truncate(Q_H')
```

Where `Truncate()` returns the left-most 4 bytes of its parameter.

Use `KDF` defined at step (4)

```
SKmac || SKenc = KDF(Z; keyDataLen; otherInfo)
```
Zeroize `Z`.

A channel for secure messaging (`SKmac`, `SKenc`) is opened.

(6) Select `rootPublicKey` for further certificate verification.

(7) If `verifyClientCert` is set to TRUE, verify `clientCert` based on `rootPublicKey`

Extract `clientIdentifier` from `clientCert` and the host authentication public key Q_{SH}.

Verify that Q_{SH} is valid for domain Parameters D.

if `verifyClientCert` is set to `FALSE`, then assume

```
clientIdentifier = rootIdentifier
```

And

```
Q_SH = rootPublicKey
```

(8) Use secure messaging to transport `iccNonce`.

```
iccNonce = RNG(nonceSize)
```

(9) Convert the coordinates of Q_{eICC} and Q_{eH} from field elements to byte strings Q_{ICC}' and Q_H'

```
sigH = AuthAlgorithm(d_sH, { Q_ICC' || clientIdentifier || iccNonce || Q_H'})
```

The `AuthAlgorithm` must comply with the domain parameters and do not allow message recovery.

(10) `sigH` must be transported using secure messaging.

Validate `sigH` using Q_{SH} over input data `{ Q_ICC' || clientIdentifier || iccNonce || Q_H'}`

The client-application has been authenticated.

(11) If `iccCertKnown` is set to TRUE, then return `iccIdentifier` instead of `iccCert`. Encrypt `iccCert` using secure messaging if `enforcePrivacy` value is set to `TRUE`.

If `enforcePrivacy` value is set to `FALSE`, then the card-application certificate does not need to be encrypted or MACed with secure messaging.

(12) if `iccCertKnown` is set to `FALSE`, extract Q_{sICC} from `iccCert`. Verify that Q_{sICC} is valid for domain parameters D.

`clientNonce` shall be transuported using secure messaging:

```
clientNonce = RNG(nonceSize)
```

(13) Convert the coordinates of Q_{eICC} and Q_{eH} from field elements to byte strings Q_{ICC}' and Q_{H}'

Locate d_{sICC}, authentication private key corresponding to Q_{sICC} , iccIdentifier, and iccCert.

sigC = AuthAlgorithm(d_{sICC}, { Q_{ICC}' || iccIdentifier || clientNonce || Q_{H}' })
authAlgorithm must comply with the domain parameters and does not allow message recovery.

(14) sigC shall be transported using secure messaging

validate sigC using Q_{sICC} over input data { Q_{ICC}' || iccIdentifier || clientNonce || Q_{H}' }

The client-application has been authenticated.

(15) authenticationProtocolData ::= empty OCTET STRING

return_code = API_OK

A.20.6.2 Impact on current state

Upon successful completion of this authentication protocol, the authentication state of the named differential-identity shall be set to the value of result within the card-application, and a session with secure messaging capability is started.

A.20.7 Encipher

A request action on a differential-identity which uses this protocol shall return the API_INAPPROPRIATE_PROTOCOL_FOR_ACTION return code.

A.20.8 Decipher

A request action on a differential-identity which uses this protocol shall return the API_INAPPROPRIATE_PROTOCOL_FOR_ACTION return code.

A.20.9 GetRandom

A request action on a differential-identity which uses this protocol shall return the API_INAPPROPRIATE_PROTOCOL_FOR_ACTION return code.

A.20.10 Hash

A request action on a differential-identity which uses this protocol shall return the API_INAPPROPRIATE_PROTOCOL_FOR_ACTION return code.

A.20.11 Sign

A request action on a differential-identity which uses this protocol shall return the API_INAPPROPRIATE_PROTOCOL_FOR_ACTION return code.

A.20.12 VerifySignature

A request action on a differential-identity which uses this protocol shall return the API_INAPPROPRIATE_PROTOCOL_FOR_ACTION return code.

A.20.13 VerifyCertificate

A request action on a differential-identity which uses this protocol shall return the API_INAPPROPRIATE_PROTOCOL_FOR_ACTION return code.

A.21 Simple EC-DH Key Agreement

This authentication protocol is a simple protocol with key agreement using EC cryptography. Both the client-application and card-application generate and use an ephemeral key pair but no static key pair for the key agreement.

This protocol is used to establish authentication (SKmac) and encryption (SKenc) session keys for further secure messaging between the ISO/IEC 24727-3 layer and the card-application.

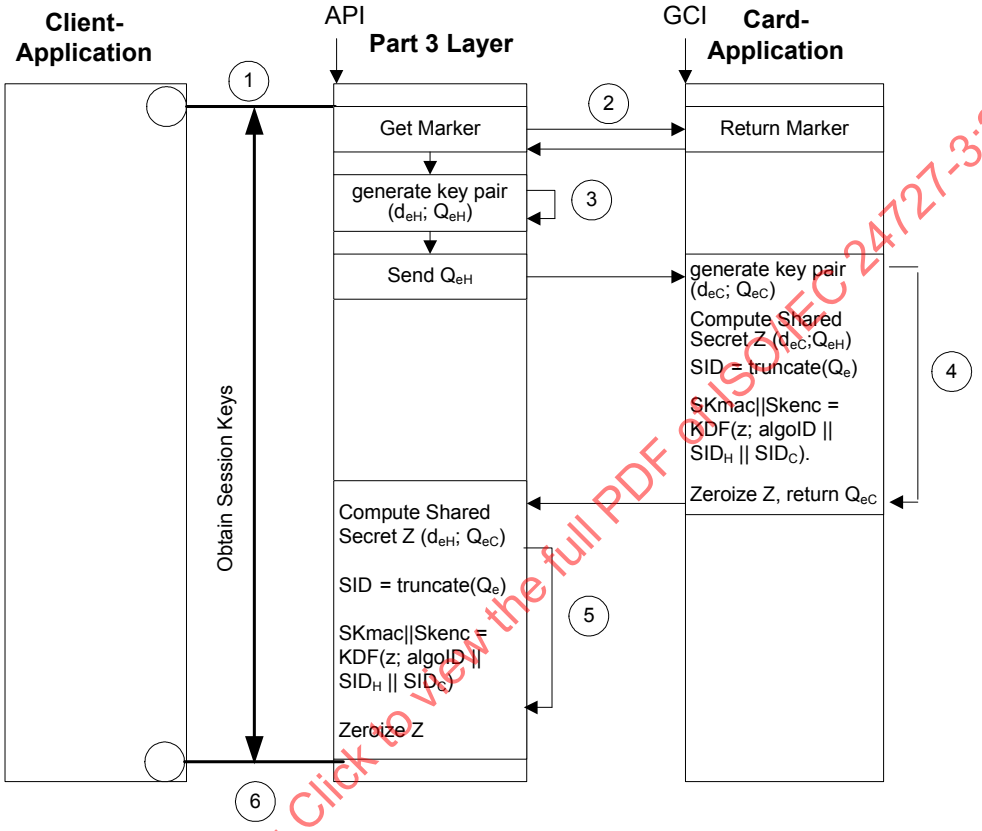

Figure A.19 — Simple EC-DH Key Agreement

A.21.1 Protocol Object Identifier

This protocol is identified by the OID `{ iso(1) standard(0) iso24727(24727) part3(3) annex-a(0) simple-ec-dh-key-agreement(21) }`

A.21.2 Marker

A differential-identity which uses this protocol has the following marker:

```
MarkerAP021 ::= SEQUENCE {
                        DomainParameters           OBJECT IDENTIFIER,
                        KeyEstablishmentAlgorithm  OBJECT IDENTIFIER,
                        KDFHashAlgorithm           OBJECT IDENTIFIER,
                        SessionMacAlgorithm        OBJECT IDENTIFIER,
                        SessionEncAlgorithm        OBJECT IDENTIFIER
}
```

A.21.3 DIDCreate

A request of the DIDCreate action intended to create a differential-identity for use with this protocol shall include a `DIDUpdateData` parameter where `marker` is the specific marker for this protocol as defined above in A.21.2.

A.21.4 DIDUpdate

A request action on a differential-identity which uses this protocol shall return the API_INAPPROPRIATE_PROTOCOL_FOR_ACTION return code.

A.21.5 DIDGet

A confirmation to the DIDGet action involving a differential-identity which uses this protocol shall include a `DIDStructure` parameter.

A.21.6 Authentication

This protocol is executed through a simple request of the CardApplicationStartSession() action with an `authenticationProtocolData` parameter as defined below.

(1) `authenticationProtocolData ::= empty OCTET STRING`

(2) Obtain domain parameters D from Marker template. The returned information must not include any data specific to the ICC. All keys, certificates, and identifiers must be set to NULL.

(3) Generate ephermeral key pair $(d_{eH}; Q_{eH})$ from domain parameters D. Send Q_{eH} to ICC.

(4) Verify that Q_{eH} is valid for Domain Parameters D

Generate ephemeral key pair $(d_{eICC}; Q_{eICC})$ from domain parameters D.

`Z = KeyEstablishmentAlgorithm(deICC;QeH)`

Convert Z to a byte string using field-element-to-byte-string

Convert the coordinates of Q_{eICC} and Q_{eH} from field elements to byte strings Q_{ICC}' and Q_{H}'

`clientIdentifier = Truncate(`Q_{ICC}`')`
`iccIdentifier = Truncate(`Q_{H}`')`

Where `Truncate()` returns the left-most bytes of its parameter

Compute session keys using the derivation function `KDF`:

$SK_{mac} \,||\, SK_{enc}$ = KDF (Z, keydataLen, otherInfo)

```
otherInfo ::= SEQUENCE {
        SessionMacAlgorithm      OBJECT IDENTIFIER,
        SessionEncAlgorithm      OBJECT IDENTIFIER,
        clientIdentifier         OCTET STRING,
        iccIdentifier            OCTET STRING,
        iccNonce                 OCTET STRING
    }
```

```
keyDataLen = algoKeyLengthInBits(SessionMacAlgorithm) +
AlgoKeyLengthInBits(SessionEncAlgorithm)

N = keyDataLen / (hashLengthInBits (KDFHashAlgorithm))

DerivedKeyingMaterial = KDFHashAlgorithm(0x00000001 || Z || otherInfo) ||
                        KDFHashAlgorithm(0x00000002 || Z || otherInfo) ||
                        …
                        KDFHashAlgorithm((0x00000000 + N) || Z || otherInfo)
```

`KDF` is the (`keyDataLen`) left-most bits from `DerivedKeyingMaterial`.

Zeroize `Z` and return Q_{eICC}.

(5) Verify that QeICC is valid for domain parameters D

Z = KeyEstablishmentAlgorithm(deH;QeICC)

Convert Z to a byte string using field-element-to-byte-string

Convert the coordinates of QeICC and QeH from field elements to byte strings QICC' and QH'

clientIdentifier = Truncate(QICC')
iccIdentifier = Truncate(QH')

Where Truncate() returns the left-most 4 bytes of its parameter.

Use KDF defined at step (4)

SKmac || SKenc = KDF(Z; keyDataLen; otherInfo)

Zeroize Z

A channel for secure messaging (SKmac,SKenc) is opened.

(6) `authenticationProtocolData ::= empty OCTET STRING`

```
return_code = API_OK
```

A.21.6.1 Impace on current state

Upon successful completion of this authentication protocol, the authentication state of the named differential-identity is not affected within the card-application; however, a session with secure messaging capability is started.

A.21.7 Encipher

A request action on a differential-identity which uses this protocol shall return the API_INAPPROPRIATE_PROTOCOL_FOR_ACTION return code.

A.21.8 Decipher

A request action on a differential-identity which uses this protocol shall return the API_INAPPROPRIATE_PROTOCOL_FOR_ACTION return code.

A.21.9 GetRandom

A request action on a differential-identity which uses this protocol shall return the API_INAPPROPRIATE_PROTOCOL_FOR_ACTION return code.

A.21.10 Hash

A request action on a differential-identity which uses this protocol shall return the API_INAPPROPRIATE_PROTOCOL_FOR_ACTION return code.

A.21.11 Sign

A request action on a differential-identity which uses this protocol shall return the API_INAPPROPRIATE_PROTOCOL_FOR_ACTION return code.

A.21.12 VerifySignature

A request action on a differential-identity which uses this protocol shall return the API_INAPPROPRIATE_PROTOCOL_FOR_ACTION return code.

A.21.13 VerifyCertificate

A request action on a differential-identity which uses this protocol shall return the API_INAPPROPRIATE_PROTOCOL_FOR_ACTION return code.

A.22 GP Asymmetric Authentication

This authentication protocol is a challenge-response protocol using public key cryptography[1]. This protocol is used both to establish an authenticated state of a differential-identity in a card-application and to transport two session keys for message authentication (SKmac) and for encryption (SKenc) between the ISO/IEC 24727-3 layer and the card-application for further secure messaging[2].

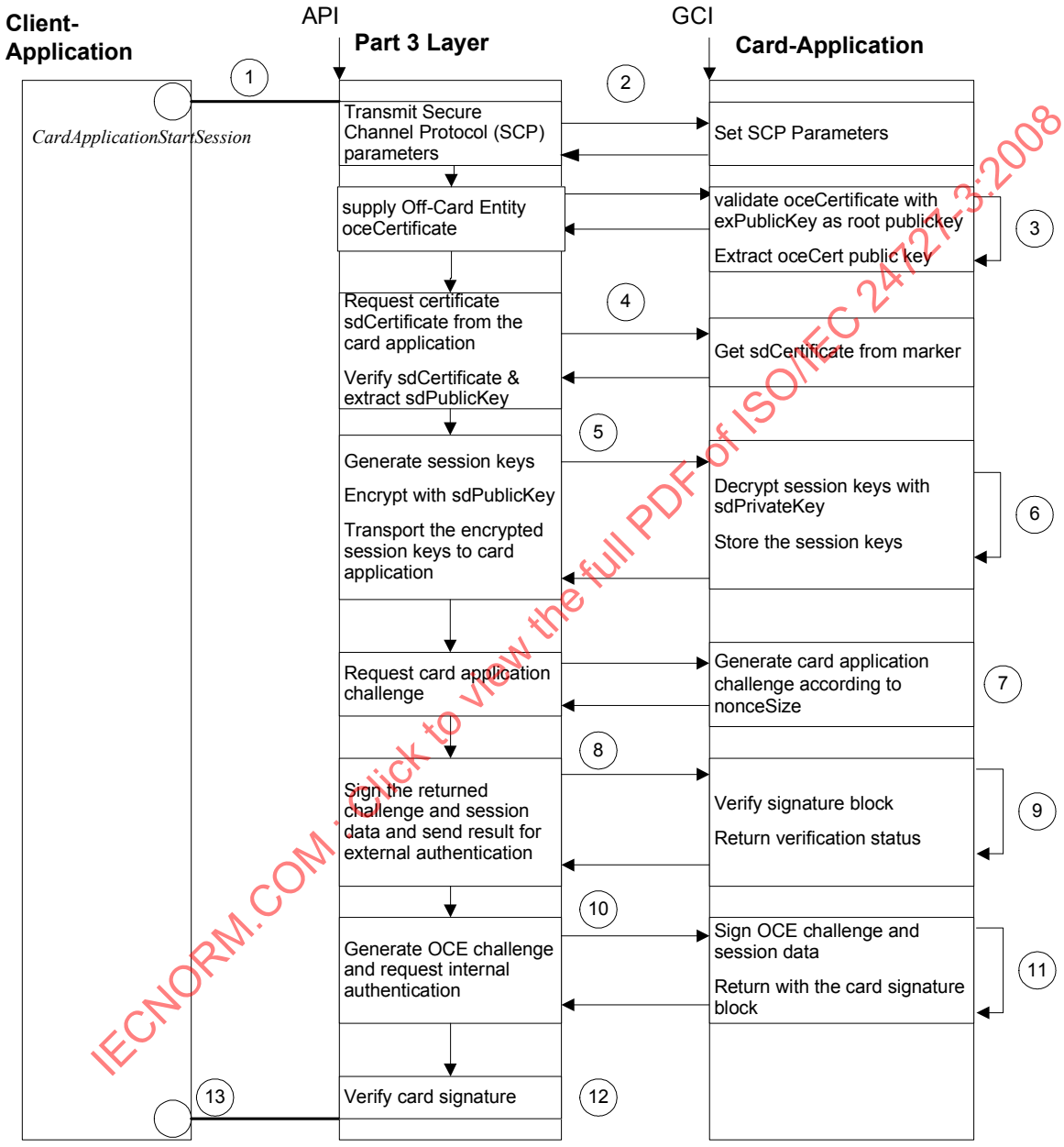

Figure A.20 — GP Asymmetric Authentication

1) Defined as “SCP10” with key transport option (*i* = 0x02) in the GlobalPlatform 2.2 Card Specification.

2) With this authentication protocol, the private key isss hosted within thhhe SAL. This implies that the client-authentication and the SAL shall be running in the same trusted environment. The appropriate configurations for this scheme (ref. ISO/IEC 24727-4) are Loyal-stack, Opaque-ICC-stack, and Remote-ICC-Stack.

A.22.1 Protocol Object Identifier

This protocol is identified by the following OIDS:

```
{ iso(1) iso24727(24727) part3(3) annex-a(0) gp-asymmetric-
authentication(22) }
```

if only external authentication is required,

```
{ iso(1) iso24727(24727) part3(3) annex-a(0) gp-asymmetric-authentication(22)
mutual-auth(0) }
```

if internal authentication (steps (10) thru (12)) is required.

A.22.2 Marker

A differential-identity which uses this protocol has a marker defined with the following template:

```
MarkerAP022 ::= SEQUENCE {
    sdPublicKey                 OCTET STRING,
    sdPrivateKey                OCTET STRING,
    sdCertificate               OCTET STRING,
    exPublicKey                 OCTET STRING, // PK.TP_EX.AUT as defined in GP 2.2
    (F.1.2.1 Overview)
    ocePubliKey                 OCTET STRING, // PK.OCE.AUT as defined in GP 2.2
    (F.1.2.1 Overview)
    kaExPublicKey               OCTET STRING, // PK.KA_EX.AUT as defined in GP 2.2
    (F.1.2.1)
    kaInCertificate             OCTET STRING, // CERT.KA_IN.AUT as defined in GP
    2.2 (F.1.2.1 Overview)
    sessionMacAlgo              OBJECT IDENTIFIER,
    sessionEncAlgo              OBJECT IDENTIFIER,
    keyEncryptionAlgo           OBJECT IDENTIFIER,
    hashAlgorithm               OBJECT IDENTIFIER,
    nonceSize                   INTEGER
    }
```

A.22.3 DIDCreate

A request of DIDCreate action intended to create a differential-identity for use with this protocol shall include a `DIDUpdateData` parameter where `marker` is the specific marker for this protocol as defined above in A.22.2.

A.22.4 DIDUpdate

A request of the DIDUpdate action involving a differential-identity which uses this authentication protocol shall include a `DIDUpdateData` parameter, where `marker` is the specific marker for this protocol as defined above in A.22.2.

A.22.5 DIDGet

A confirmation to the DIDGet action involving a differential-identity which uses this authentication protocol shall include a `DIDStructure` parameter.

A.22.6 Authentication

This protocol is executed through a simple request of the CardApplicationStartSession action with an authententicationProtocolData parameter as defined below.

A request of the DIDAuthentication action on a differential-identity which uses this protocol shall return the API_INAPPROPRIATE_PROTOCOL_FOR_ACTION return code.

A.22.6.1 Procedure

(1)
```
authenticationProtocolData ::= SEQUENCE {
oceCertificate      OCTET STRING,
securityLevel       OCTET STRING,    // As specified in GlobalPlatform
                                        Specifications 2.2 Table 10-1
SCPParametersP1P2   OCTET STRING,    // As specified in GlobalPlatform
                                        Specifications 2.2 F4.5.2 and F4.5.4
}
```

(2) The SCP parameters are communicated to the card-application

(3) The security domain(SD) of the card-application verifies `oceCertificate` and establishes the validity of `ocePublicKey`, the corresponding public key ,using `exPublicKey` as the authority public key stored with the marker.

(4) The ISO/IEC 24727-3 layer verifies `sdCertificate` and establishes the validity of `sdPublicKey` based on the authority certificate and the policies of the issuer of `sdCertificate`.

(5) The ISO/IEC 24727-3 layer generates session keys for MAC and ENC, encrypts them with `sdPublicKey` and transports the encrypted data block `encData` to the card-application.

```
skMac = sessionMacAlgo()

skEnc = sessionEncAlgo()

crtMac ::= SEQUENCE {
            crtTag          OCTET STRING, // 'B4' (CCT) or 'B8' (CT)
            crtLength       OCTET STRING,
            usageQualifier  OCTET STRING, // '95.01.XX'
            skMac           OCTET STRING, // 'D1.18.XX…XX'
            cryptoAlgo      OCTET STRING, // GP 2.2 Table F-11 and 11.1.8, with a
                                             value corresponding to sessionMacAlgo
                                             OID
            sequenceCounter OCTET STRING // '91.08.XX…XX'
 }

crtEnc ::= SEQUENCE {
            crtTag          OCTET STRING, // 'B4' (CCT) or 'B8' (CT)
            crtLength       OCTET STRING,
            usageQualifier  OCTET STRING, // '95.01.XX'
            skEnc           OCTET STRING, // 'D1.18.XX…XX'
            cryptoAlgo      OCTET STRING, // GP 2.2 Table F-11 and 11.1.8, with a
                                             value corresponding to sessionMacAlgo
                                             OID
            sequenceCounter OCTET STRING // '91.08.XX…XX'
 }
```

```
keyData ::= SEQUENCE {
            paddingPattern     OCTET STRING, // '00.02.FF…FF.00'
            securityLevelTag   OCTET STRING, // 'D3'
            securityLevelLen   OCTET STRING, // 'D1'
            securityLevel      OCTET STRING,
            crtMac             OCTET STRING,
            crtEnc             OCTET STRING
}

encData = keyEncryptionAlgo[sdPublicKey](keyData)
```

(6) The card-application decrypts the received data block with sdPrivateKey to derive the session keys for MAC and ENC, and stores them for later secure message usage.

```
keyData = keyEncryptionAlgo-1[sdPrivateKey](encData)
```

(7) The card-application generates a challenge for external authentication

```
sdNonce = RNG(nonceSize)
```

(8) The ISO/IEC 24727-3 layer sends its signature to the card-application for external authentication. The ISO/IEC 24727-3 layer signature is the result of generating a digest (hash) over a set of data, creating a signature block, and signing the signature block with `ocePrivateKey`, the private key corresponding to `oceCertificate`.

```
dataToHash ::= SEQUENCE {
                securityLevelTag     OCTET STRING, // 'D3'
                securityLevelLen     OCTET STRING, // '01'
                securityLevel        OCTET STRING,
                crtMac               OCTET STRING,
                crtEnc               OCTET STRING,
                sdNonce              OCTET STRING
}

hashData = hashAlgorithm(dataToHash)

sigBlock ::= SEQUENCE {
                paddingPattern       OCTET STRING, // '00.01.FF…FF.00'
                derHashAlgorithm     OCTET STRING, // DER encoded OID
                                                      (of GP's hash algo)
                derHashData          OCTET STRING // DER encoded hashData
}

oceSignature = authAltorithm[ocePrivateKey](sigBlock)
```

(9) The card-application verifies oceSignature using the previously derived ocePublicKey in step (2). The secure channel is now available if the verification succeeds. If internal authentication is not required (first OID supported by this authentication protocol), proceed to step (13).

(10) If internal authentication is required (for the second OID supported by this authentication protocol), then execute steps (10), (11), and (12)

OCE generates a nonce to challenge the card-application:

oceNonce = RNG(nonceSize)

(11) The card-application sends its signature back to the OCE for internal authentication. The card-application signature is the result of generating a digest (hash) over a set of data, creating a signature block, and signing the block with the `sdPrivateKey`:

```
sdSigBlock ::= SEQUENCE {
                  paddingPattern       OCTET STRING,    // '00.01.FF…FF.00'
                  derHashAlgorithm     OCTET STRING,    // DER encoded OID
                  derHashData          OCTET STRING     // DER encoded hashData

hashData = hashAlgorithm(dataToHash)

sdDataToHash ::= SEQUENCE {
                  crtMac               OCTET STRING,
                  crtEnc               OCTET STRING,
                  oceNonce             OCTET STRING
}

sdSignature = authAlgorithm[sdPrivateKey](sdSigBlock)
```

(12) The ISO/IEC 24727-3 layer verifies sdSignature using the previously derived sdPublicKey in step (3). This confirms that the card-application has correctly derived the session keys sent by the ISO/IEC 24727-3 layer.

(13) `authenticationProtocolData ::= empty OCTET STRING`
`return_code = API_OK` if authentication is completed successfully.

A.22.6.2 Impact on current state

Upon successful completion of this authentication protocol, the authentication stat of the named differential-identity shall be set to TRUE within the card-application, and a session with secure messaging capability is started.

A.22.7 Encipher

A request of this action on a differential-identity which uses this protocol shall return the `API_INAPPROPRIATE_PROTOCOL_FOR_ACTION` return code.

A.22.8 Decipher

A request of this action on a differential-identity which uses this protocol shall return the `API_INAPPROPRIATE_PROTOCOL_FOR_ACTION` return code.

A.22.9 GetRandom

A request of this action on a differential-identity which uses this protocol shall return the `API_INAPPROPRIATE_PROTOCOL_FOR_ACTION` return code.

A.22.10 Hash

A request of this action on a differential-identity which uses this protocol shall return the `API_INAPPROPRIATE_PROTOCOL_FOR_ACTION` return code.

A.22.11 Sign

A request of this action on a differential-identity which uses this protocol shall return the `API_INAPPROPRIATE_PROTOCOL_FOR_ACTION` return code.

A.22.12 VerifySignature

A request of this action on a differential-identity which uses this protocol shall return the `API_INAPPROPRIATE_PROTOCOL_FOR_ACTION` return code.

A.22.13 VerifyCertificate

A request of this action on a differential-identity which uses this protocol shall return the `API_INAPPROPRIATE_PROTOCOL_FOR_ACTION` return code.

A.22.14 CardApplicationEndSession

A request to this action after a differential-identity utilizing this protocol has been authenticated, shall set this differential-identity authentication state to `FALSE`, and return the `API_OK` return code.

A.23 GP Symmetric Authentication (Explicit Mode)

This authentication protocol is a challenge-response protocol using symmetric key cryptography[3]. This protocol is used both to establish an authenticated state of a differential-dientity in a card-application and to agree on two session keys for message authentication (skMac) and for encryption (skEnc) between the ISO/IEC 24727-3 layer and the card-application for further secure messaging.

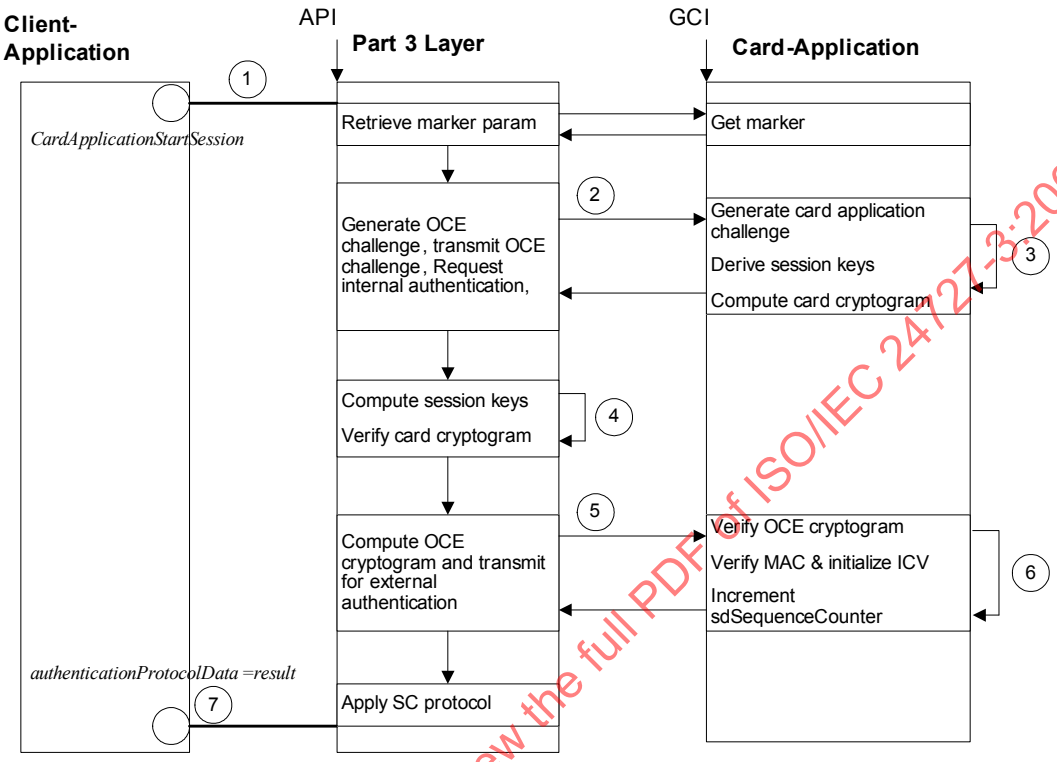

Figure A.21 — GP Symmetric Authentication (Explicit Mode)

A.23.1 Protocol Object Identifier

This protocol is identified by the OID `{ iso(1) standard(0) iso24727(24727) part3(3) annex-a(0) gp-symmetric-authentication-explicit-mode(23) }`

A.23.2 Marker

A differential-identity which uses this protocol has a marker defined as the following:

```
MarkerAP023 ::= SEQUENCE {
      sdStaticEncKey          OCTET STRING,
      sdStaticMacKey          OCTET STRING,
      sdStaticDekKey          OCTET STRING,
      sdSequenceCounter       OCTET STRING,
      sessionMacAlgo          OBJECT IDENTIFIER,
      sessionEncAlgo          OBJECT IDENTIFIER,
      keyDerivationAlgo       OBJECT IDENTIFIER,
      nonceSize               INTEGER
   }
```

3) Defined as SCP02 in explicit mode (*i* = 0x25) in GlobalPlatform 2.2 Card Specification.

A.23.3 DIDCreate

A request of the DIDCreate action intended to create a differential-identity for use with this authentication protocol shall include a `DIDUpdateData` parameter where `marker` is the specific marker for this protocol as defined above in A.23.2.

A.23.4 DIDUpdate

A request of the DIDUpdate action involving a differential-identity which uses this authentication protocol shall include a `DIDUpdateData` parameter, where `marker` is the specific marker for this protocol as defined above in A.23.2.

A.23.5 DIDGet

A confirmation of a request to the DIDGet action involving a differential-identity which uses this protocol shall include a `DIDStructure` parameter.

A.23.6 Authentication

This protocol is executed through a simple request of the CardApplicationStartSession action with an authenticationProtocolData parameter as defined below.

A request of the DIDAuthentication action on a differentia-identity which uses this protocol shall return the API_INAPPROPRIATE_PROTOCOL_FOR_ACTION return code.

A.23.6.1 Procedure

(1) **authenticationProtocolData** ::=

```
securityLevel    OCTET STRING // As specified in GlobalPlatform
                                 Card Specification 2.2 Table E-11
```

(2) The ISO/IEC 24727-3 layer generates a challenge to the card-application for internal authentication:

```
oceNonce = RNG(nonceSize)
```

(3) The card-application also generates a nonce, then derives the session keys and computes the card-application cryptogram:

```
sdNonce = RNG(nonceSize)

skMac = keyDerivationAlgo[sdStaticEncKey](sdSequenceCounter)
skEnc = keyDerivationAlgo[sdStaticMacKey](sdSequenceCounter)

sdInput ::= SEQUENCE {
                    oceNonce                OCTET STRING,
                    sdSequenceCounter       OCTET STRING,
                    sdNonce                 OCTET STRING,
                    padding                 OCTET STRING
}

sdCryptogram = authAlgorithm[skEnc](sdInput)
```

(4) The ISO/IEC 24727-3 layer also computes the card-application as in step (3), and performs a comparison. It authenticates the card-application when the comparison succeeds.

(5) The ISO/IEC 24727-3 layer computes and off-card entity cryptogram and sends it together with a CMAC pattern to the card-applicatioon for external authentication:

```
oceInput ::= SEQUENCE {
                    sdSequenceCounter       OCTET STRING,
                    sdNonce                 OCTET STRING,
                    oceNonce                OCTET STRING,
                    padding                 OCTET STRING
}

oceCryptogram = authAlgorithm[skEnc](oceInput)

macInput ::= SEQUENCE {
                    apduHeader              OCTET STRING,
                    oceCryptogram           OCTET STRING
}

CMAC = sessionMacAlgo[skMac](macInput)
```

(6) The card-application computes the same cryptogram and `CMAC` and performs a comparison with the data received in step (5). It authenticates the ISO/IEC 24727-3 layer if the comparison is successful, and increments the `sdSequenceCounter` and sets the initial `ICV` with the `CMAC` value.

(7) **`authenticationProtocolData`** `::= empty OCTET STRING`

`return_code = API_OK` if authentication is completed successfully.

A.23.6.2 Impact on current state

Upon successful completion of this authentication protocol, the authentication state of the named differential-identity shall be set to the value of 'result' within the card-application, and a session with secure messaging is started.

A.23.7 Encipher

A request to this action referencing a differential-identity which uses this protocol shall return the `API_INAPPROPRIATE_PROTOCOL_FOR_ACTION` return code.

A.23.8 Decipher

A request to this action referencing a differential-identity which uses this protocol shall return the `API_INAPPROPRIATE_PROTOCOL_FOR_ACTION` return code.

A.23.9 GetRandom

A request to this action referencing a differential-identity which uses this protocol shall return the `API_INAPPROPRIATE_PROTOCOL_FOR_ACTION` return code.

A.23.10 Hash

A request to this action referencing a differential-identity which uses this protocol shall return the `API_INAPPROPRIATE_PROTOCOL_FOR_ACTION` return code.

A.23.11 Sign

A request to this action referencing a differential-identity which uses this protocol shall return the `API_INAPPROPRIATE_PROTOCOL_FOR_ACTION` return code.

A.23.12 VerifySignature

A request to this action referencing a differential-identity which uses this protocol shall return the `API_INAPPROPRIATE_PROTOCOL_FOR_ACTION` return code.

A.23.13 VerifyCertificate

A request to this action referencing a differential-identity which uses this protocol shall return the `API_INAPPROPRIATE_PROTOCOL_FOR_ACTION` return code.

A.23.14 CardApplicationEndSession

A request to this action after a differential-identity has been successfully authenticated shall set this differential-identity authentication state to `FALSE`, and return the `API_OK` return code.

A.24 GP Symmetric Authentication (Implicit Mode)

This authentication protocol is a simple assertion indicating that the subsequent communication with the card-application will be in implicit-mode secure messaging[4)]. This protocol is used to both establish an authenticated state of a differential-identity in a card-application and to agree on two session keys for message authentication (skMac) and for encryption (skEnc) between the ISO/IEC 24727-3 layer and the card-application for further secure messaging.

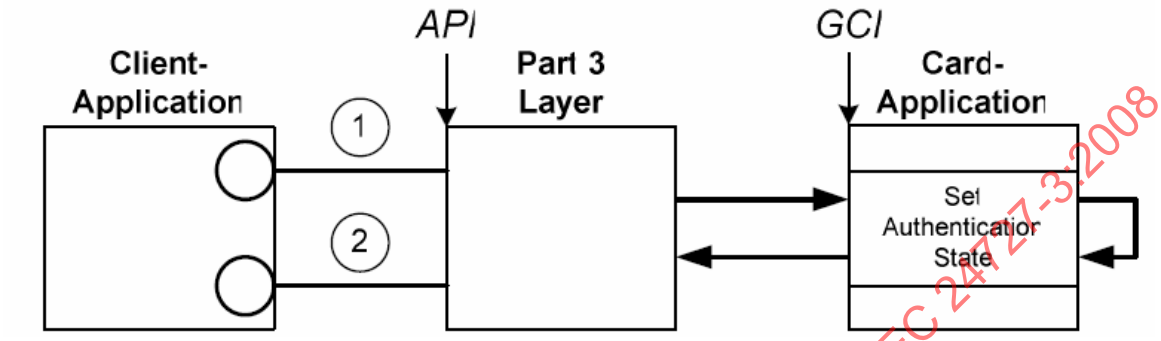

Figure A.22 — GP Symmetric Authentication (Implicit Mode)

A.24.1 Protocol Object Identifier

This protocol is identified by the following OID: `{ iso(1) standard(0) iso24727(24727) part3(3) annex-a(0) gp-asymmetric-authentication-implicit-mode(24) }`

A.24.2 Marker

A differential-identity which uses this protocol shall have a marker defined as the following:

```
SpecificMarkerStruct ::= SEQUENCE {
      sdStaticBaseKey          OCTET STRING,
      sdSequenceCounter        OCTET STRING,
      sessionMacAlgo           OBJECT IDENTIFIER,
      sessionEncAlgo           OBJECT IDENTIFIER,
      keyDerivationAlgo        OBJECT IDENTIFIER
   }
```

A.24.3 DIDCreate

A request of the DIDCreate action intended to create a differential-identity for use with this protocol shall include a `didStructure` parameter as defined below:

```
didStructure ::= SEQUENCE {
                              authProtocol   OBJECT IDENTIFIER,
                              marker         OCTET STRING,
                              scope          BOOLEAN OPTIONAL,
                              qualifier      DIDQualifier OPTIONAL

}
```

4) Defined as SCP02 in implicit mode (*i* = 0x29) in GlobalPlatform Card Specification 2.2.

A.24.4 DIDUpdate

A request of the DIDUpdate action involving a differential-identity which uses this authentication protocol shall include a `SpecificMarkerStruct` as defined in A.24.2.

A.24.5 DIDGet

A confirmation of a request to the DIDGet action involving a differential-identity which uses this protocol shall include a `SpecificMarkerStruct` defined in A.24.2.

A.24.6 Authentication

This protocol is executed through a simple request of the CardApplicationStartSession action with a `NULL authenticationProtocolData` parameter, as it is implicity known by each party.

A request of the DIDAuthenticate action on a differential-identity which uses this protocol shall return the `API_INAPPROPRIATE_PROTOCOL FOR ACTION` return code.

A.24.6.1 Procedure

(1) `authenticationProtocolData ::= empty OCTET STRING`

(2) `authenticationProtocolData ::= empty OCTET STRING`

`return_code = API_OK` if authentication is completed successfully.

A.24.6.2 Impact on current state

Upon successful completion of this authentication protocol, the authentication state of the named differential-identity shall be set to TRUE within the card-application.

Upon subsequently receipt of the first APDU in secure messaging, the session keys are computed as follows:

```
skMac = keyDerivationAlgo[sdStaticBaseKey](sdSequenceCounter)

skEnc = keyDerivationAlgo[sdStaticBaseKey](sdSequenceCounter)
```

The ICV is set to “MAC over AID as specified in the GlobalPlatform card Specification 2.2 Table E-1 for computing the first CMAC attached to the first APDU protected with seure messaging. Upon successful verification of the CMAC value, the card-application increments `sdSequenceCounter`.

A.24.7 Encipher

A request to this action referencing a differential-identity which uses this protocol shall return the `API_INAPPROPRIATE_PROTOCOL_FOR_ACTION` return code.

A.24.8 Decipher

A request to this action referencing a differential-identity which uses this protocol shall return the `API_INAPPROPRIATE_PROTOCOL_FOR_ACTION` return code.

A.24.9 GetRandom

A request to this action referencing a differential-identity which uses this protocol shall return the `API_INAPPROPRIATE_PROTOCOL_FOR_ACTION` return code.

A.24.10 Hash

A request to this action referencing a differential-identity which uses this protocol shall return the `API_INAPPROPRIATE_PROTOCOL_FOR_ACTION` return code.

A.24.11 Sign

A request to this action referencing a differential-identity which uses this protocol shall return the `API_INAPPROPRIATE_PROTOCOL_FOR_ACTION` return code.

A.24.12 VerifySignature

A request to this action referencing a differential-identity which uses this protocol shall return the `API_INAPPROPRIATE_PROTOCOL_FOR_ACTION` return code.

A.24.13 VerifyCertificate

A request to this action referencing a differential-identity which uses this protocol shall return the `API_INAPPROPRIATE_PROTOCOL_FOR_ACTION` return code.

A.24.14 CardApplicationEndSession

A request to this action after a differential-identity has been successfully authenticated shall set this differential-identity authentication state to `FALSE`, and return the `API_OK` return code.

Annex B
(normative)

Cryptographic algorithms

B.1 Interoperability requirements

This annex provides the defining source for the cryptographic algorithms used in the authentication protocols specified in Annex A.

B.2 Symmetric Algorithms

B.2.1 References

18033-2 refers to hash functions in 10118-2 and 10118-3.

18033-2 describes key derivation functions KDF1 and KDF2 with dependences on hash functions.

18033-2 refers to MAC algorithms in 9797-1 and 9797-2.

18033-2 refers to block ciphers in 18033-3.

18033-2 describes symmetric ciphers SC1 and SC2 with dependences on block ciphers and key derivation functions.

18033-2 describes key encapsulation mechanisms (KEM).

18033-2 describes data encapsulation mechanisms (DEM).

18033-2 describes hybrid ciphers.

18033-2 describes ElGamal-based KEM.

18033-2 describes RSA-based asymmetric cipher and KEM.

18033-3 defines TDEA (which is 3DES) in terms of DES.

18033-3 defines DES (in Annex A).

18033-3 defines MISTY1.

18033-3 defines CAST-128.

18033-3 defines AES.

18033-3 defines Camellia.

18033-3 defines SEED.

18033-4 describes the construction of stream ciphers from block ciphers.

18033-4 defines MUGI keystream generator.

18033-4 defines SNOW 2.0 keystream generator.

B.2.2 From 18033-3

The following OIDs and the algorithms to which they refer are defined in ISO/IEC 18033-3.

```
OID ::= OBJECT IDENTIFIER

is18033-3         OID ::= { iso(1) standard(0) is18033(18033) part3(3) }
```

```
id-bc64           OID ::= { is18033-3 cipher-64-bit(1) }

id-bc128          OID ::= { is18033-3 cipher-128-bit(2) }

id-bc64-tdea      OID ::= { id-bc64 tdea(1) }

id-bc64-misty1    OID ::= { id-bc64 misty1(2) }

id-bc64-cast128   OID ::= { id-bc64 cast128(3) }

id-bc128-aes      OID ::= { id-bc128 aes(1) }

id-bc128-camellia OID ::= { id-bc128 camellia(2) }

id-bc128-seed     OID ::= { id-bc128 seed(3) }
```

B.2.3 AES (128-bit key)

```
id-aes128-ECB ::= { joint-iso-ccitt(2) country(16) us(840) organization(1)
                    gov(101) csor(3) nistAlgorithm(4) 1 1 }

id-aes128-CBC ::= { joint-iso-ccitt(2) country(16) us(840) organization(1)
                    gov(101) csor(3) nistAlgorithm(4) 1 2 }

id-aes128-OFB ::= { joint-iso-ccitt(2) country(16) us(840) organization(1)
                    gov(101) csor(3) nistAlgorithm(4) 1 3 }

id-aes128-CFB ::= { joint-iso-ccitt(2) country(16) us(840) organization(1)
                    gov(101) csor(3) nistAlgorithm(4) 1 4 }
```

B.2.4 AES (192-bit key)

```
id-aes192-ECB ::= { joint-iso-ccitt(2) country(16) us(840) organization(1)
                    gov(101) csor(3) nistAlgorithm(4) 1 21 }

id-aes192-CBC ::= { joint-iso-ccitt(2) country(16) us(840) organization(1)
                    gov(101) csor(3) nistAlgorithm(4) 1 22 }

id-aes192-OFB ::= { joint-iso-ccitt(2) country(16) us(840) organization(1)
                    gov(101) csor(3) nistAlgorithm(4) 1 23 }

id-aes192-CFB ::= { joint-iso-ccitt(2) country(16) us(840) organization(1)
                    gov(101) csor(3) nistAlgorithm(4) 1 24 }
```

B.2.5 AES (256 bit key)

```
id-aes256-ECB ::= { joint-iso-ccitt(2) country(16) us(840) organization(1)
                    gov(101) csor(3) nistAlgorithm(4) 1 41 }

id-aes256-CBC ::= { joint-iso-ccitt(2) country(16) us(840) organization(1)
                    gov(101) csor(3) nistAlgorithm(4) 1 42 }

id-aes256-OFB ::= { joint-iso-ccitt(2) country(16) us(840) organization(1)
                    gov(101) csor(3) nistAlgorithm(4) 1 43 }

id-aes256-CFB ::= { joint-iso-ccitt(2) country(16) us(840) organization(1)
                    gov(101) csor(3) nistAlgorithm(4) 1 44 }
```

B.2.6 DES

```
DES-ECB ::= { iso(1) identified-organization(3) oiw(14) secsig(3) algorithm(2) 6 }

DES-CBC ::= { iso(1) identified-organization(3) oiw(14) secsig(3) algorithm(2) 7 }

-- Carries an IV as a parameter.

DES-OFB ::= { iso(1) identified-organization(3) oiw(14) secsig(3) algorithm(2) 8 }

-- Carries an FBParameter as a parameter.

where,

FBParameter ::= SEQUENCE           {
                        iv         IV,
                        numberOfBits  NumberOfBits
                        }

and,

IV ::= OCTET STRING

NumberOfBits ::= INTEGER -- number of feedback bits (1-64)
```

B.2.7 Others

```
des-EDE3-CBC ::= { iso(1) member-body(2) us(840) rsadsi(113549)
                   encryptionAlgorithm(3) 7}

rc2CBC       ::= { iso(1) member-body(2) us(840) rsadsi(113549)
                   encryptionAlgorithm(3) 2}

rc5-CBC-PAD  ::= { iso(1) member-body(2) us(840) rsadsi(113549)
                   encryptionAlgorithm(3) 9}
```

B.3 Asymmetric Algorithms

B.3.1 Public Keys

```
RSA public key: 1.2.840.113549.1.1.1

DSA public key: 1.2.840.10040.4.1

EC public key: 1.2.840.10045.2.1
```

B.3.2 Asymmetric Encryption

```
RSA-OAEP ::= 1.2.840.113549.1.1.7
```

B.3.3 Signatures

```
DSA with SHA-1: 1.2.840.10040.4.3

md2WithRSAEncryption     ::= 1.2.840.113549.1.1.2

md5WithRSAEncryption     ::= 1.2.840.113549.1.1.4

sha-1WithRSAEncryption   ::= 1.2.840.113549.1.1.5

sha224WithRSAEncryption  ::= 1.2.840.113549.1.1.14

sha256WithRSAEncryption  ::= 1.2.840.113549.1.1.11

sha384WithRSAEncryption  ::= 1.2.840.113549.1.1.12

sha512WithRSAEncryption  ::= 1.2.840.113549.1.1.13

RSASSA-PSS ::= 1.2.840.113549.1.1.10

-- same OID for all hash algorithms (hash algorithm is specified as a parameter).
```

B.4 Elliptic Curve Algorithms

B.4.1 Elliptic Curves

```
ansiX9p192r1 ::= { iso(1) member-body(2) us(840) 10045 curves(3) prime(1) 1 }
ansiX9t163k1 ::= { iso(1) identified-organization(3) certicom(132) curve(0)  1 }
ansiX9t163r2 ::= { iso(1) identified-organization(3) certicom(132) curve(0) 15 }
ansiX9p224r1 ::= { iso(1) identified-organization(3) certicom(132) curve(0) 33 }
ansiX9t233k1 ::= { iso(1) identified-organization(3) certicom(132) curve(0) 26 }
ansiX9t233r1 ::= { iso(1) identified-organization(3) certicom(132) curve(0) 27 }
ansiX9p256r1 ::= { iso(1) member-body(2) us(840) 10045 curves(3) prime(1) 7 }
ansiX9t283k1 ::= { iso(1) identified-organization(3) certicom(132) curve(0) 16 }
ansiX9t283r1 ::= { iso(1) identified-organization(3) certicom(132) curve(0) 17 }
ansiX9p384r1 ::= { iso(1) identified-organization(3) certicom(132) curve(0) 34 }
ansiX9t409k1 ::= { iso(1) identified-organization(3) certicom(132) curve(0) 36 }
ansiX9t409r1 ::= { iso(1) identified-organization(3) certicom(132) curve(0) 37 }
ansiX9p521r1 ::= { iso(1) identified-organization(3) certicom(132) curve(0) 35 }
ansiX9t571k1 ::= { iso(1) identified-organization(3) certicom(132) curve(0) 38 }
ansiX9t571r1 ::= { iso(1) identified-organization(3) certicom(132) curve(0) 39 }
```

B.4.2 Signatures

```
ECDSA with SHA-1:   1.2.840.10045.4.1
ECDSA with SHA-224: 1.2.840.10045.4.3.1
ECDSA with SHA-256: 1.2.840.10045.4.3.2
ECDSA with SHA-384: 1.2.840.10045.4.3.3
ECDSA with SHA-512: 1.2.840.10045.4.3.4
```

B.5 Hash Functions

```
md2    ::= { iso(1) member-body(2) US(840) rsadsi(113549) digestAlgorithm(2) 2 }

md5    ::= { iso(1) member-body(2) US(840) rsadsi(113549) digestAlgorithm(2) 5 }

sha1   ::= { iso(1) identified-organization(3) oiw(14) secsig(3) algorithms(2)
             26 }

sha224 ::= { joint-iso-itu-t(2) country(16) us(840) organization(1) gov(101)
             csor(3) nistalgorithm(4) hashalgs(2) 4 }

sha256 ::= { joint-iso-itu-t(2) country(16) us(840) organization(1) gov(101)
             csor(3) nistalgorithm(4) hashalgs(2) 1 }

sha384 ::= { joint-iso-itu-t(2) country(16) us(840) organization(1) gov(101)
             csor(3) nistalgorithm(4) hashalgs(2) 2 }

sha512 ::= { joint-iso-itu-t(2) country(16) us(840) organization(1) gov(101)
             csor(3) nistalgorithm(4) hashalgs(2) 3 }
```

B.6 Message Authentication Codes

```
id-hmacWithSHA1   ::= { iso(1) member-body(2) us(840) rsadsi(113549) 2 7}

id-hmacWithSHA224 ::= { iso(1) member-body(2) us(840) rsadsi(113549) 2 8}

id-hmacWithSHA256 ::= { iso(1) member-body(2) us(840) rsadsi(113549) 2 9}

id-hmacWithSHA384 ::= { iso(1) member-body(2) us(840) rsadsi(113549) 2 10}

id-hmacWithSHA512 ::= { iso(1) member-body(2) us(840) rsadsi(113549) 2 11}

HMAC-MD5          ::= { iso(1) org(3) dod(6) internet(1) security(5)
                        mechanisms(5) ipsec(8) isakmpOakley(1) 1 }
```

B.7 Key Establishment

```
dhSinglePass-stdDH-sha1kdf-scheme      ::= { iso(1) identified-organization(3)
                                             tc68(133) country(16) x9(840)
                                             x9-63(63) schemes(0) 2}

dhSinglePass-cofactorDH-sha1kdf-scheme ::= { iso(1) identified-organization(3)
                                             tc68(133) country(16) x9(840)
                                             x9-63(63) schemes(0) 3}

mqvSinglePass-sha1kdf-scheme           ::= { iso(1) identified-organization(3)
                                             tc68(133) country(16) x9(840)
                                             x9-63(63) schemes(0) 16}

x9-63-scheme                           ::= { iso(1) member-body(2) US(840)
                                             ansi-x9-63(63) schemes(0) }

secg-scheme                            ::= { iso(1) identified-organization(3)
                                             certicom(132) schemes(1) }

dhSinglePass-stdDH-sha1kdf             ::= { x9-63-scheme 2 }

dhSinglePass-cofactorDH-sha1kdf        ::= { x9-63-schemem 3 }

dhSinglePass-cofactorDH-recommendedKDF ::= { secg-scheme 1 }

dhSinglePass-cofactorDH-specifiedKDF   ::= { secg-scheme 2 }
```

Annex C
(normative)

ASN.1 Representation

C.1 General

```
ASN1 for Annex C of ISO_IEC_24727-31.txt
ISO24727API -- {iso(1) standard(0) iso24727(24727) part3(3) }
--
-- Version 1.60
--
DEFINITIONS IMPLICIT TAGS ::=
BEGIN
```

-- C.1.1 Constants

```
size-max-NameLength INTEGER ::= 255
size-max-NodePathLength          INTEGER ::= 255
size-max-Padding                 INTEGER ::= 16
size-max-SecurityCondition       INTEGER ::= 255
```

-- C.1.2 The Data Types

```
ByteValue ::=                    INTEGER (FROM (0..255) )
ConnectionHandle ::=             INTEGER
ApplicationIdentifier ::=        [APPLICATION 15] OCTET STRING
ObjectIdentifier ::=             OBJECT IDENTIFIER
Name ::=                         VisibleString (SIZE(1..size-max-NameLength))
CardApplicationName ::=          ApplicationIdentifier
CardApplicationNameList ::=      SET OF CardApplicationName
DSIName ::=                      Name
DSINameList ::=                  SET OF DSIName
DSIContent ::=                   OCTET STRING
DataSetName ::=                  Name
DataSetNameList ::=              SET OF DataSetName

ProtocolTerminationPoint ::=     OCTET STRING
TransactionIdentifier ::=        OCTET STRING
DIDScope ::= CHOICE {
    local [0] NULL,
    global [1] NULL
}

CardApplicationServiceLoadPackage ::= OCTET STRING

ExecuteActionRequest ::= OCTET STRING
ExecuteActionConfirmation ::= OCTET STRING

CipherBuffer ::= OCTET STRING
MessageBuffer ::= OCTET STRING
HashBuffer ::= OCTET STRING
SignatureBuffer ::= OCTET STRING

RandomDataBuffer ::= OCTET STRING
```

-- C.1.3 The Data Structures

```
DifferentialIdentityAuthenticationState ::= SEQUENCE {
    dIDName      DIDName,
    dIDScope     DIDScope,
    dIDState     BOOLEAN
}

SecurityCondition ::= CHOICE {
    didAuthentication        [1] DifferentialIdentityAuthenticationState,
    always                   [2] BOOLEAN (FROM (TRUE)),
    never                    [3] BOOLEAN (FROM (FALSE)),
    not                      [4] SecurityCondition,
    and                      [5] SEQUENCE SIZE (1..size-max-SecurityCondition) OF
SecurityCondition,
    or                       [6] SEQUENCE SIZE (1..size-max-SecurityCondition) OF
SecurityCondition
}

AccessRule ::= SEQUENCE {
    cardApplicationService   CardApplicationServiceName,
    action                   ActionName,
    securityCondition        SecurityCondition
}
AccessControlList ::= SET OF AccessRule

ReaderAction ::= ENUMERATED { reset(0), unpower(1), eject(2), confiscate(3) }

CardApplicationPath ::= SEQUENCE {
    pathSecurity         PathSecurityType OPTIONAL,
    -- The pathSecurity element specifies the protection
    -- between the local dispatcher and the remote dispatcher
    -- which is located at channelHandle.protocolTerminationPoint
    channelHandle        ChannelHandleType OPTIONAL,
    contextHandle        ContextHandleType OPTIONAL,
    iFDName              UTF8String OPTIONAL,
    slotIndex            INTEGER OPTIONAL,
    cardApplication      ApplicationIdentifier
}

PathSecurityType ::= SEQUENCE {
    protocol      PATHSECURITYPARAMETERS.&id({SupportedPathSecurityProtocols}),
    parameters    PATHSECURITYPARAMETERS.&Type({SupportedPathSecurityProtocols} {@protocol}) OPTIONAL
}

unknownPathSecurityProtocolOID OBJECT IDENTIFIER ::= { iso(1) standard(0) iso24727(24727) part3(3)
annex-c(2) pathSecurityProtocols(0) unknown(0) }

unknownPathSecurityProtocol PATHSECURITYPARAMETERS ::= {
    ID  unknownPathSecurityProtocolOID
    TYPE NULL
}

SupportedPathSecurityProtocols PATHSECURITYPARAMETERS ::= {unknownPathSecurityProtocol}

PATHSECURITYPARAMETERS ::= CLASS {
    &id     OBJECT IDENTIFIER,
    &Type
}   WITH SYNTAX {ID &id
        TYPE &Type}
```