Technical Specification

ISO/IEC TS 18013-7

First edition
2024-10

Personal identification — ISO-compliant driving licence —

Part 7:
Mobile driving licence (mDL) add-on functions

Identification des personnes — Permis de conduire conforme à l'ISO —

Partie 7: Fonctionnalités supplémentaires pour permis de conduire sur téléphone mobile

Reference number
ISO/IEC TS 18013-7:2024(en)

ISO copyright office
CP 401 • Ch. de Blandonnet 8
CH-1214 Vernier, Geneva
Phone: +41 22 749 01 11
Email: copyright@iso.org
Website: www.iso.org

Published in Switzerland

Contents

Foreword

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

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

ISO and IEC draw attention to the possibility that the implementation of this document may involve the use of (a) patent(s). ISO and IEC take no position concerning the evidence, validity or applicability of any claimed patent rights in respect thereof. As of the date of publication of this document, ISO and IEC had not received notice of (a) patent(s) which may be required to implement this document. However, implementers are cautioned that this may not represent the latest information, which may be obtained from the patent database available at www.iso.org/patents and https://patents.iec.ch. ISO and IEC shall not be held responsible for identifying any or all such patent rights.

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

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

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

A list of all parts in the ISO 18013 series can be found on the ISO website.

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

Introduction

ISO/IEC 18013-5 describes interface and related requirements to facilitate ISO-compliant driving licence functionality on a mobile device, standardizing the mobile driving licence (mDL) functionality.

This document augments the capabilities of the mDL by describing the interface and related requirements for presentation to a mDL reader over the internet.

A mobile document conforming to this document primarily conveys the driving privileges associated with a person. However, the transaction and security mechanisms in this document have been designed to support other types of mobile documents, specifically including identification documents.

NOTE ISO/IEC 18013-5 places the onus on the mDL verifier to match data received (in an mdoc) to the person presenting the mdoc. This version of this document does not change this.

Personal identification — ISO-compliant driving licence —

Part 7:
Mobile driving licence (mDL) add-on functions

1 Scope

This document augments the capabilities of the mobile driving licence (mDL) standardized in ISO/IEC 18013-5 with the following additional functionality:

- presentation of a mobile driving licence to a reader over the internet.

2 Normative references

The following documents are referred to in the text in such a way that some or all of their content constitutes requirements 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 18013-5, *Personal identification — ISO-compliant driving licence — Part 5: Mobile driving licence (mDL) application*

RFC 4648, *S. Josefsson, The Base16, Base32, and Base64 Data Encodings*

RFC 5280, *D. Cooper et al., Internet X.509 Public Key Infrastructure Certificate and Certificate Revocation List (CRL) Profile*

RFC 9101, *N. Sakimura, The OAuth 2.0 Authorization Framework: JWT-Secured Authorization Request (JAR*

RFC 9112, *R. Fielding et al., HTTP/1.1*

OID4VP (OpenID for Verifiable Presentations), *O. Terbu et al., Draft 18,* April 2023

3 Terms and definitions

For the purposes of this document, the terms and definitions given in ISO/IEC 18013-5 and the following apply.

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

- ISO Online browsing platform: available at https://www.iso.org/obp
- IEC Electropedia: available at https://www.electropedia.org/

3.1
mdoc reader
either device or service, or both, that can retrieve data from an mdoc and verify the authenticity of the data

Note 1 to entry: The mdoc reader includes, but is not limited to, the hardware and software components used.

4 Abbreviated terms

OID4VP	OpenID for Verifiable Presentations

5 Conformance requirement

An mDL is in conformance with this document if it meets all the requirements specified directly or by reference herein.

An mDL reader is in conformance with this document if it meets all the requirements specified directly or referenced herein.

NOTE Conformance of an mDL or an mDL reader with ISO/IEC 18013-5 is not required for conformance with this document, except for those clauses normatively referenced in this document. An mDL or an mDL reader conforming with this document can also be in conformity with ISO/IEC 18013-5.

6 mDL overview

6.1 Standards context

ISO/IEC 18013-5 describes the interface and related requirements to specifically facilitate ISO-compliant driving licence functionality on a mobile device. This document adds functionality by building on top of ISO/IEC 18013-5.

The transaction and security mechanisms in this document have been designed to also be applicable to other types of mobile documents besides the mobile driving licence.

6.2 Interfaces

Figure 1 shows the interfaces in scope for this document. The explanation of each interface is as follows:

- Interface 1 in Figure 1 is the interface between the issuing authority (IA) infrastructure and the mDL. This interface is out of scope for this document.
- Interface 2 in Figure 1 is the interface between the mDL and the mDL reader. This interface is specified in this document. The interface can be used for connection setup and for the device retrieval method.
- Interface 3 in Figure 1 is the interface between the IA infrastructure and the mDL reader. This interface is defined in ISO/IEC 18013-5. No new requirements are added in this document.

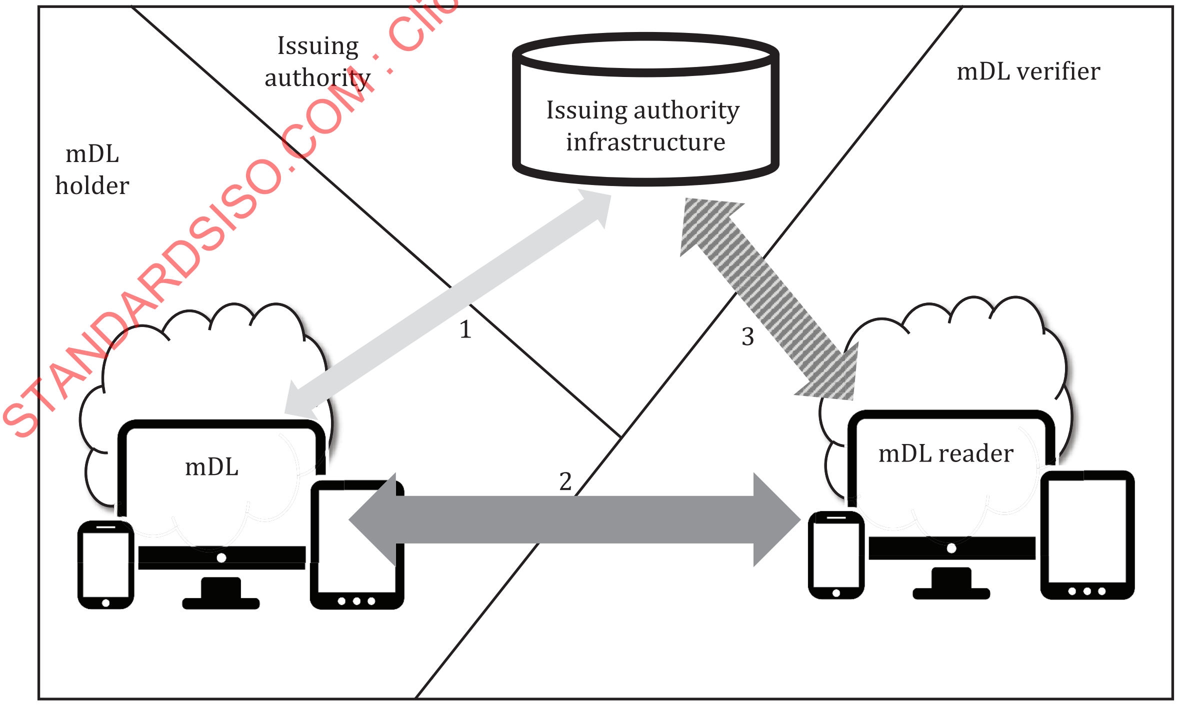

Figure 1 — mDL interfaces

6.3 Design objectives

The objectives underlying the requirements in this document include at least the following:

a) An mDL verifier together with an mDL reader is able to request and receive an mDL, and validate its integrity and authenticity.

b) An mDL verifier not associated with the IA is able to verify the integrity and authenticity of an mDL.

c) An mDL verifier is enabled to confirm the binding between the person presenting the mDL and the mDL holder.

d) The interface between the mDL and the mDL reader supports the selective release of mDL data to an mDL reader.

NOTE As in ISO 18013-5, the portrait image can be used for verifying that the person presenting the mDL is the mDL holder. Depending on the transaction details, in an unattended transaction this data element might not be able to serve the purpose of confirming that the person presenting the mDL is the mDL holder. Other methods can be used as well but are out of scope of this document. Other mechanisms are described in References [1] and [2].

6.4 Technical requirements

6.4.1 Data structures and data elements

The descriptions and requirements for Concise Binary Object Representation (CBOR), Concise Data Definition Language (CDDL), and version elements in ISO/IEC 18013-5 shall apply in this document.

Additionally, unless explicitly stated otherwise for a data structure, an mdoc or mdoc reader shall not give an error purely on the basis that it does not know the element. This requirement also applies when the CDDL definition of the data structure does not allow the presence of additional key-value pairs in the map, next to the specified ones.

6.4.2 Data model

The data model is described in Clause 7. It describes the identifier and format of the data elements.

6.4.3 Data exchange

6.4.3.1 Overview

An mDL or mDL reader shall support at least one of the flows and may support more:

a) Using the device retrieval messages structures and transmission channel as defined in 6.4.3.3 that is setup:

   1) using remote engagement, as defined in 6.4.3.2; or

   2) using an out-of-band mechanism, as defined in 6.4.3.2;

b) Using OID4VP as a transmission channel, as defined in Annex B.

The different flows are depicted in Figure 2.

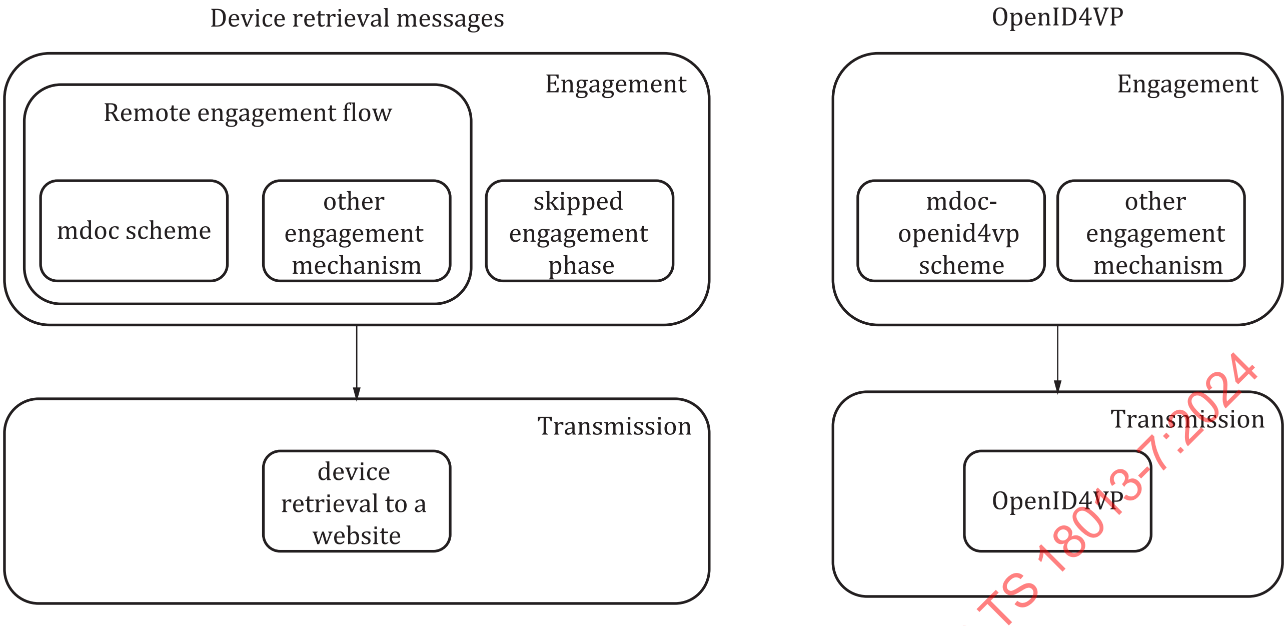

Figure 2 — Flows for unattended cases

An mDL and mDL reader shall support at least one of the following data retrieval methods and may support more. Table 1 shows the requirements.

- device retrieval as defined in 6.4.3.3;
- OID4VP as defined in Annex B.

Table 1 — Data retrieval methods

Data retrieval method	**Support**		**Reference in this document**
	mDL	mDL reader	
Device retrieval	C[a]	C[a]	6.4.3.3
OID4VP	C[a]	C[a]	Annex B

Key

C conditional

[a] Support for at least one of these methods is mandatory.

6.4.3.2 Device retrieval engagement phase

The engagement mechanism for remote engagement can be used to exchange the information required to set up a secure data retrieval mechanism between the mDL and mDL reader. When performing remote engagement, the following flow shall be used:

a) The mDL reader transmits the ReaderEngagement structure to the mDL.

b) The mDL sets up a data transmission channel with the mDL reader using the information from the ReaderEngagement structure.

c) The mDL sends a DeviceEngagement structure to the mDL reader using the newly setup data transmission channel.

The ReaderEngagement and DeviceEngagement structures are defined in A.1 and A.2. A possible mechanism for transmission of the ReaderEngagement structure is defined in A.4. Support for this transmission mechanism is recommended for the mDL and mDL reader, since this is the only mechanism currently provided in this document. However, other mechanisms for transmitting the ReaderEngagement structure, which are not defined in this document, can be used.

When the mDL and mDL reader have an existing two-way data transmission channel that is set up out-of-band for exchange of data, the device retrieval engagement phase can be skipped.

6.4.3.3 Device retrieval data retrieval phase

The general data retrieval architecture is described in ISO/IEC 18013-5. If an mDL or mDL reader supports the device retrieval data retrieval phase, they shall use the mdoc request and mdoc response structures as specified in ISO/IEC 18013-5.

A.6.2 defines a transmission technology for device retrieval that may be supported by an mDL or an mDL reader.

NOTE ISO/IEC 18013-5 defines the server retrieval data retrieval method. This document does not specify any additional requirements for server retrieval.

6.4.4 Security mechanisms

6.4.4.1 Security architecture

The security of mDL data exchanged with an mDL reader is designed to preserve the triad of confidentiality, integrity, and authenticity by design and by default.

The security architecture aims to achieve the following goals:

a) Protection against forgery: Data elements are signed by the IA. The degree of protection against forgery depends on the degree to which the IA's keys are protected. Minimizing the validity period of the data limits the value of the data.

b) Protection against cloning: The mDL produces a signature or message authentication code over session data. The private key used to authenticate the session data is stored only in the mDL. The corresponding public key in turn is signed by the IA. The degree of protection against cloning depends on the degree to which the mDL authentication key is protected. In addition to protecting DeviceKey by secure storage, an mdoc/mDL can require the user to be authenticated before this key is usable. This depends on the jurisdiction/issuer's policy (e.g. AAL as per eIDAS Regulation, NIST SP 800-63, ISO/IEC 29115).

c) Protection against eavesdropping: Communications between mDL and mDL readers are encrypted and authenticated. The mDL reader can detect man-in-the-middle (MITM) attacks by validating the anti-cloning signature or message authentication code, which is created using a key for which the public part is signed by the IA in the mobile security object (MSO), see ISO/IEC 18013-5. If mdoc reader authentication is used, the mDL can detect MITM attacks before returning any data.

d) Protection against unauthorized access: An mDL is protected from unauthorized access by an mDL reader by multiple mechanisms. When session encryption is used, the encryption key used for communications between the mDL and mDL reader is derived from an ephemeral key pair from both the mDL and mDL reader. The mDL can optionally authenticate the mDL reader by means of an mDL reader authentication certificate and a signature created by the mDL reader using the corresponding private key. The mDL reader certificate is signed by a certificate authority trusted by the mDL for this purpose.

e) Protection of the mDL holder against relayed engagement information: the mDL includes in the device engagement data, the origin information of the engagement channel or the data transmission channel for the mDL reader to confirm it. The origin is determined by the mDL independently from the information transmitted in the reader engagement structure. The transaction is cancelled by the mDL reader when the origin is different from the expected value.

Revocation of an mDL is out of scope for this document. However, the MSO includes update information and validity time frames which enable the mDL reader to check the freshness of the data. The IA shall define appropriate periods of validity that balance freshness with offline capability, considering that a shorter validity period mitigates certain security risks.

6.4.4.2 Security mechanisms support requirements

Table 2 describes the security mechanisms that can be implemented by an mDL or an mDL reader. Issuer data authentication, and mdoc authentication shall be implemented by the mDL and mDL reader. Session encryption shall be implemented if the device retrieval to a website mechanism, specified in A.6 is used. mdoc reader authentication is optional for the mDL and mDL reader.

mdoc authentication, mdoc reader authentication and session encryption shall use the session transcript as defined in A.8

A.5 contains further requirements on the use of mdoc mac authentication.

NOTE 1 ISO/IEC 18013-5 describes the use of the X.509 certificates when using mdoc reader authentication. Other mechanisms for providing the mDL reader public key and trust information can also be used.

The certificate and CRL profile requirements in ISO/IEC 18013-5 shall be applied for the following profiles: IACA root certificate, IACA link certificate, document signer certificate, mdoc reader authentication certificate, Online Certificate Status Protocol (OCSP) signer certificate, CRL profile.

All certificates issued by an IACA or another CA shall be validated according to ISO/IEC 18013-5.

An mDL reader needs access to the issuing authority's certificate authority (IACA) root certificate to verify issuer data authentication. An optional method to get access to these certificates is described in ISO/IEC 18013-5 using a verified issuer certificate authority list (VICAL) provider.

See the privacy and security recommendations in ISO/IEC 18013-5 for additional information on privacy and security.

Table 2 — Security mechanisms

Security mechanisms	Support	Reference
Session encryption	Conditional (see 6.4.3)	A.6
Issuer data authentication	Mandatory	ISO/IEC 18013-5
mdoc authentication	Mandatory	ISO/IEC 18013-5
mdoc reader authentication	Optional	ISO/IEC 18013-5

6.4.4.3 Additional verification requirements

If the OriginInfo as defined in A.3 contains the domain origin type as defined in A.3.2, the mDL reader shall verify whether it matches the domain of where the mDL was requested.

The behaviour of the mDL reader when it receives an empty string as value for the key "domain" in the "details" field of the domain origin OriginInfo type is out of scope of this document.

The mDL reader shall also verify any other elements in the OriginInfo that it understands and for which it can obtain the info to verify it. If the verification fails, the mDL reader shall terminate the transaction and invalidate any received data.

6.5 Protocol considerations

6.5.1 General

This clause reflects security and privacy considerations that implementers of flows and methods described in this document can consider.

6.5.2 Discovery and invocation of mdoc using a custom URI scheme

mdoc and mdoc-openID4VP URI schemes present limitations when used to invoke the wallet. Examples of limitations include (but are not limited to):

EXAMPLE 1 When using the custom URI scheme on iOS, the developer documentation notes that "If multiple apps register the same scheme, the app the system targets is undefined. There's no mechanism to change the app or to change the order apps appear in a Share sheet" (see Reference [19]).

EXAMPLE 2 Discussions are circulating around the possible deprecation of support for certain URI schemes that can cause implementations to break (see Reference [19]).

EXAMPLE 3 The user receives no assistance in selecting the appropriate wallet since the custom URI scheme only provides selection based on protocol. This can lead to the user making an incorrect wallet selection.

EXAMPLE 4 Custom URI schemes require apps to ensure protection from malformed input data. Further solutions that assist in executing this protection (see 6.5.3) can be helpful.

EXAMPLE 5 Custom URI schemes inhibit the capabilities of the browser to protect the user.

6.5.3 Possible attack

6.5.3.1 Attack description

A possible attack is when a victim authenticates for a session at the relying party that is under the attacker's control, or more specifically, when an attacker interacts with a relying party to generate a link to then forward that link to a victim to have the victim complete the process on behalf of the attacker.

6.5.3.2 Device retrieval to a website

For device retrieval to a website, the solution is for the user agent to provide the domain origin to the mdoc application. Certain browsers have settings that can prevent the domain origin information from being provided by the user agent. In addition, some browsers do not support providing the domain origin information via schemes. In situations like this, if the presentment is performed, engagement information can be forwarded by an attacker and the mDL holder is vulnerable to the above attack.

6.5.3.3 OID4VP

For OID4VP, a solution is for the mdoc reader to maintain the binding between the user session and the nonce authorization request parameter. While a reader is required to implement a mechanism to maintain the binding, this document does not define one. In addition, absent a list of trusted readers (that are confirmed to maintain the binding, and that can be used by the mdoc to make/inform decisions about the transaction), the mdoc does not have a way to check if the binding is maintained. If the binding is not maintained and the presentment is performed, engagement information can be forwarded by an attacker and the mDL holder is vulnerable to the above attack. There are ongoing discussions in the OpenID Foundation to propose a solution (see Reference [20]).

6.5.4 Additional flows and methods

Work is ongoing to improve the way in which an mdoc reader and an mdoc can interact when not in close proximity. This includes work on a browser API in an incubator group of the World Wide Web Consortium (W3C). Work is also ongoing in the OpenID Foundation on this subject (see Reference [3]).

7 mDL data model

The mDL data model descriptions and requirements in ISO/IEC 18013-5 shall apply in this document with the following exception: an mDL may require mdoc reader authentication as a precondition for the release of any of the mandatory data elements.

NOTE 1 This differs from the requirement from ISO/IEC 18013-5

NOTE 2 In order for an mDL and mDL reader to use mdoc reader authentication, a trust relationship between the parties involved must exist.

Annex A
(normative)

Mechanisms for device retrieval to a website

A.1 Reader engagement

The reader engagement structure contains the information to perform the engagement flow as defined in 6.4.3.2. The reader engagement structure shall be CBOR encoded and formatted as follows:

```
ReaderEngagement =
{
    0: tstr,            ; Version
    1: Security,
    ? 2: DeviceRetrievalMethods,
    * int => any
}

Security = [
    int,               ; Cipher suite identifier
    EReaderKeyBytes
]

DeviceRetrievalMethods = [
    + DeviceRetrievalMethod
]

DeviceRetrievalMethod = [
    uint,               ; Type
    uint,               ; Version
    RetrievalOptions    ; Specific option(s) to the type of retrieval method
]

RetrievalOptions = RestApiOptions / any
```

The reader engagement structure contains the key-value pairs described below. Additional positive key-value pairs within the engagement structure are RFU. An application-specific extension shall use a negative integer for the key. An mdoc or mdoc reader shall ignore any key-value pairs with a negative key value that it is not able to interpret.

0) `Version`: the version of the reader engagement structure, in the current version of this document its value shall be "1.1".

1) `Security`: an array that contains two mandatory elements. The first element is the cipher suite identifier, defined in ISO/IEC 18013-5. The second element is `EReaderKeyBytes`, defined in ISO/IEC 18013-5.

2) `DeviceRetrievalMethods`: an array that shall contain one or more `DeviceRetrievalMethod`. A `DeviceRetrievalMethod` array holds three mandatory values (`Type`, `Version`, `RetrievalOptions`). This document only specifies the values for the device retrieval to a website method from A.6. When using this method, the value for `Type` shall be 4, the value for `Version` shall be 1, and the value for `RetrievalOptions` shall be `RestApiOptions` as defined in A.6.

A.2 DeviceEngagement

The device engagement structure contains the information the mdoc reader needs to perform the engagement flow as defined in 6.4.3.2. The device engagement structure shall be CBOR encoded and formatted as follows:

```
DeviceEngagement =
{
    0: tstr,            ; Version
    1: Security,
    5: OriginInfos,
    ? 6: Capabilities,
    * int => any
}

Capabilities = {
    ? 0: MacKeysSupport,
    ? 1: MacKeysCurves,
    * int => any
}

MacKeysSupport = bool
MacKeysCurves = [*crv]
crv = int
```

The device engagement structure contains the key-value pairs described below. Additional positive key-value pairs within the engagement structure are RFU. An application-specific extension shall use a negative integer for the key. An mdoc or mdoc reader shall ignore any key-value pairs with a negative key value that it is not able to interpret.

0) `Version`: the version of the device engagement structure, in the current version of this document its value shall be “1.1”.

1) `Security`: This structure is defined in ISO/IEC 18013-5. A.7 describes how this structure is used.

5) `OriginInfos`: The `OriginInfos` structure provides information about the interface used to receive and deliver the engagement structure, it is defined in A.3. `OriginInfos` shall be present.

6) `Capabilities`: Contains a map of capabilities the mdoc supports. Currently two optional elements are defined: `MacKeysSupport` and `MacKeysCurves`. The contents of these two values are defined below.

The `MacKeysSupport` element in the `Capabilities` structure can be used by the mdoc to indicate whether it supports the `MacKeys` element in the mdoc request, as defined in A.5. When the value is set to `True`, the mdoc indicates support for the `MacKeys` element.

The `MacKeysCurves` element in the `Capabilities` structure may be used by the mdoc to indicate which curves it supports for mdoc mac authentication. If present it shall contain all the curves that are supported by the mdoc application for mdoc mac authentication. The value of `MacKeysCurves` is an array of `cry`, as defined in the IANA COSE Elliptic Curves registry. For privacy reasons, the contents of this field shall not be determined by actual presence of any mobile documents.

A.3 Origin info

A.3.1 General

The mdoc reader shall use the origin info to detect if engagement information from a different mdoc reader is forwarded by a malicious mdoc reader. This is done by indicating to the mdoc reader through what channel the mdoc received the engagement information.

The origin information structure that contains this information is `OriginInfos`. It shall be CBOR encoded and formatted as follows:

```
OriginInfos = [
    * OriginInfo
]

OriginInfo = {
    "cat" : Category,      ; Category
    "type" : Type,        ; Type
    "details" : Details    ; Details
}

Category = uint
Type = uint
Details = {
  + tstr: any
}
```

The `OriginInfos` is an array that consists of one or more `OriginInfo` structures. The value for `Category` shall be 1, any other value is RFU. The value of `Details` depends on the value of `Type`. This document defines the types “Domain origin” with type = 1, and “Other” with type = 0. Any other values for the type are RFU.

NOTE One of the reserved values can in the future be used for purposes of providing information about Qualified Website Authentication Certificate (QWAC) as defined in ETSI TS 119 495.

A.3.2 Domain origin

This type indicates the origin of the channel used to retrieve the engagement information.

For type 1, the “Details" map structure shall contain one field with “domain” as the identifier and the website domain as the value.

The value of the “domain” element may be an empty string. This signifies that the mdoc received the engagement information through a channel that has a domain, but the mdoc did not receive the value, or did not receive it from a source trusted by the mdoc.

In order for the domain origin field to mitigate phishing attacks, the value must be received from a source trusted by the mdoc. This document does not define what source the mdoc must receive the website domain from, but one of the options is to retrieve it from the referrer URL of the page from which the mdoc was requested.

The value shall not be determined using data from the ReaderEngagement structure.

EXAMPLE

Referrer URL = “https://gov.example.com/present/session1?hi=2"

Contents of Details = {

“domain“ = “gov.example.com”

}

A.3.3 Other

This type can be used to provide additional information about the channel via which the mdoc received the engagement information.

The value for this type shall be 0.

The value of the elements of `Details` is out of scope of this document.

A.4 mdoc scheme

To invoke the mdoc App through the mdoc scheme (mdoc://), the mdoc reader shall use:

- URI starts with “mdoc://”
- Followed by the `readerEngagement` data encoded base64 url-without-padding according to RFC 4648.

Note The URI scheme “mdoc://” has been reserved by ISO/IEC 18013-5 with Internet Assigned Numbers Authority (IANA).

A.5 MacKeys support

When performing mdoc authentication using a MAC as specified in ISO/IEC 18013-5, the ephemeral reader key and static device key are used to calculate the MAC. Since the static device key typically cannot be changed, this requires the ephemeral reader key to use the same curve as the static device key. In the flow specified in ISO/IEC 18013-5, this is solved by letting the mdoc generate the ephemeral device key for session encryption to be of the same curve as its static device key. The mdoc reader will then always generate an ephemeral reader key that uses the same curve.

However, in the flow specified in 6.4.3.2, the mdoc reader decides on the curve to use for the ephemeral reader key without having knowledge yet about the static device key of the mdoc. To mitigate this issue, this document defines an additional optional element, `MacKeys`, to the DeviceRequest structure as defined in ISO/IEC 18013-5. This provides the capability for the mdoc reader to send multiple keys with different curves which can be used for mdoc mac authentication. This mitigates the issue of the mdoc reader not knowing which key to send, or when the mdoc contains documents with different curves for mdoc mac authentication. The `MacKeys` element has the following definition:

```
? "macKeys"  : MacKeys
MacKeys = [+ COSE_Key]
```

This results in the following definition of DeviceRequest:

```
DeviceRequest = {
    "version" : tstr,                    ; Version of DeviceRequest structure
    "docRequests" : [+ DocRequest]       ; Requested documents
    ? "macKeys" : MacKeys
}
```

If present, the `MacKeys` element shall contain one or more ephemeral reader keys. The `MacKeys` element shall not contain more than one key for each curve. The mdoc reader shall not include the `MacKeys` element unless the mdoc indicated support for it in the DeviceEngagement structure as defined in A.2. When `MacKeys` are provided in the request structure, the version element shall have value “1.1”.

It is recommended for the mdoc reader to include all the curves supported by this document in the `MacKeys` element. However, if the mdoc uses the `MacKeysCurves` to indicate which keys it supports, the mdoc reader can use this information to limit the amount of different curves it should send.

If an mdoc reader provides the `MacKeys` element in the request structure, and the mdoc chooses to perform mdoc MAC authentication, the mdoc shall choose the applicable reader key from this element.

A.6 Device retrieval to a website

A.6.1 Engagement

To be able to set up the connection, the mdoc must know the URI at which the mdoc reader can be reached. When engagement is performed using the reader engagement structure, this information is included in `RestApiOptions` element in the reader engagement structure, which is formatted as follows:

```
RestApiOptions = {
    0: tstr,; URI of the website
}
```

The string shall contain the URI of the website in which to connect. The mdoc reader shall ensure that the URI contains the path that the mdoc reader uses to determine to which session the URI belongs.

A.6.2 Transport protocol

The mdoc shall use HTTP/1.1 POST according to RFC 9112 commands for sending `DeviceEngagementMessage` or `SessionData` and receiving `SessionEstablishment` or `SessionData` as follows:

HTTP POST messages shall have the following structure:

```
POST [path of URI] HTTP/1.1
Host: [host of URI]
Content-Length: [content length]
Content-Type: application/cbor

[
DeviceEngagementMessage or SessionData message]
```

HTTP successful response message shall have the following structure:

```
HTTP/1.1 200 OK
Content-length: [content length]
Content-type: application/cbor

[SessionEstablishment or SessionData message]
```

The content-type shall be application/cbor. Other header fields may be included.

The `SessionEstablishment` and `SessionData` messages are defined in ISO/IEC 18013-5, `DeviceEngagementMessage` is defined in A.7.

HTTP error responses are specified in RFC 9110:2022, section 15. Communication between the mdoc and the mdoc reader shall use Transport Layer Security with server authentication as specified in ISO/IEC 18013-5.

When Device Retrieval to a website is used as a transport mechanism, the "Domain origin" element shall be present in the OriginInfo structure.

A.7 Session encryption

When performing the remote engagement flow, and if session encryption is used, session encryption shall be implemented in accordance with ISO/IEC 18013-5 with the following changes:

The following step shall be added:

0) Reader engagement. The mdoc reader generates a new ephemeral key pair (EReaderKey.Priv, EReaderKey.Pub), and populates the `Security` element in the `ReaderEngagement` structure with the cipher suite identifier, the identifier of the elliptic curve to be used for key agreement and the ephemeral mdoc reader public key.

Step 1 and 2 shall be changed to:

1) Device engagement. The mdoc generates a new ephemeral key pair (EDeviceKey.Priv, EDeviceKey.Pub). To do so, the mdoc should use the curve of the ephemeral reader public key it received, if it supports that curve. The mdoc shall populate the `Security` element in the `DeviceEngagement` with the cipher suite identifier, the identifier of the elliptic curve to be used for key agreement and the ephemeral pub key. If the curves used by the mdoc and mdoc reader ephemeral keys are the same, the mdoc shall derive the session keys as defined in ISO/IEC 18013-5. The mdoc shall send the device engagement structure to the mdoc reader in a `DeviceEngagementMessage` as specified below.

2) Session establishment. The mdoc reader shall derive the session keys as defined in ISO/IEC 18013-5. using the cipher suite, the elliptic curve and the ephemeral device public key received in the `deviceEngagement` structure. If this curve is the same as the curve of the mdoc reader ephemeral key pair the mdoc reader generated in step 0, the mdoc reader shall use the associated private key for session key derivation. If not, the mdoc reader shall generate a new ephemeral key pair (EReaderKey.Priv, EReaderKey.Pub) using the elliptic curve identified by the mdoc and shall use that private key.

   The mdoc reader shall encrypt the mdoc request with the appropriate session key. If the mdoc reader derived a new ephemeral reader key pair in this step, it shall send the encrypted mdoc request to the mdoc in a SessionEstablishment message, together with EReaderKey.Pub. If the mdoc reader used the ephemeral reader key pair from step 0 instead, it shall send the encrypted mdoc request in a `SessionData` message, leaving out the "`status`" pair.

If the mdoc did not yet derive the session keys in step 1, it shall do so now, using the mdoc reader ephemeral public key in the SessionEstablishment message. If the mdoc derived session keys in step 1 and it received a new ephemeral reader key in step 2, the mdoc shall terminate the session. The mdoc shall decrypt the mdoc request. The session shall continue as described in ISO/IEC 18013-5.

The Device Engagement message in step 1 shall be CBOR encoded and formatted as follows:

```
DeviceEngagementMessage = {
               "deviceEngagementBytes": DeviceEngagementBytes  ;
}
DeviceEngagementBytes = #6.24(bstr .cbor DeviceEngagement); see Annex A.2
```

A.8 Session transcript

The SessionTranscript as defined in ISO/IEC 18013-5 shall be used with the following changes:

The handover element is defined as:

```
Handover = EngagementToApp / any
EngagementToApp  = ReaderEngagementBytesHash

ReaderEngagementBytesHash = bstr

ReaderEngagementBytes = #6.24(bstr .cbor ReaderEngagement)
```

Where `ReaderEngagementBytesHash` is the SHA-256 hash of `ReaderEngagementBytes`.

`EngagementToApp` shall be used when the remote engagement flow is used.

When an mdoc and mdoc reader uses an out-of-band mechanism to set up the device retrieval phase, the contents of the `Handover` element is out of scope of this document.

When DeviceEngagement is not applicable to the transfer mechanism used, the value shall be null.

When EReaderKey is not applicable to the transfer mechanism used, the value shall be null.

Annex B
(normative)

Use of OID4VP to retrieve an mdoc

B.1 Mechanism description

B.1.1 General

This document supports presentation of mdoc using OID4VP, an extension to OAuth 2.0 as defined in Reference [8]. It enables the mdoc holder to present an mdoc directly to the mdoc reader. This document is an mdoc-specific profile of OID4VP defined in OID4VP, Draft 18. It clarifies mandatory-to-implement features for the options mentioned in OID4VP for the implementers who wish to present an mdoc response. Presentation of the mdoc shall use the same-device flow as defined in OID4VP, Section 3 with the Response Mode defined in OID4VP, Section 6.2.

NOTE 1 Cross-device flows are prone to engagement relay attacks which is the reason cross-device flow is not included in this document. This issue is not specific to OID4VP only.

NOTE 2 When implementing all requirements in Annex B, the mdoc app acts as an OAuth 2.0 Authorization Server towards the remote mdoc reader which acts as an OAuth 2.0 Client. OID4VP uses the terms Verifier for OAuth 2.0 Client and Wallet for OAuth 2.0 Authorization Server.

B.1.2 Sequence diagram

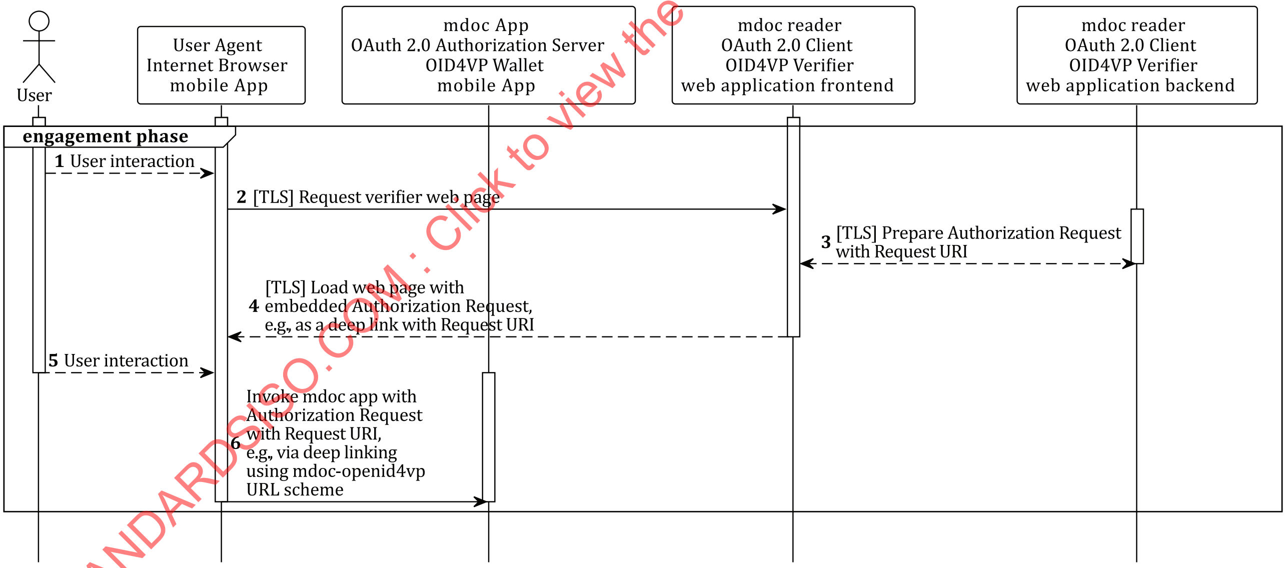
User
User Agent
Internet Browser
mobile App
mdoc App
OAuth 2.0 Authorization Server
OID4VP Wallet
mobile App
mdoc reader
OAuth 2.0 Client
OID4VP Verifier
web application frontend
mdoc reader
OAuth 2.0 Client
OID4VP Verifier
web application backend
engagement phase
1 User interaction
2 [TLS] Request verifier web page
3 [TLS] Prepare Authorization Request with Request URI
4 [TLS] Load web page with embedded Authorization Request, e.g. as a deep link with Request URI
5 User interaction
6 Invoke mdoc app with Authorization Request with Request URI, e.g., via deep linking using mdoc-openid4vp URL scheme

Figure B.1 — Engagement phase sequence diagram

Figure B.1 and Figure B.2 show the informative sequence diagram of the engagement phase and device retrieval phase. During the engagement phase, the mdoc app receives the Authorization Request including the `request_uri` which is used by the mdoc app to connect to the mdoc reader.

In steps 1 to 6, OID4VP does not define when the Request URI is generated and how the Authorization Request including the Request URI is presented on the web page. In step 4, a session was established between the user agent and the mdoc reader. In step 6, the user agent invokes the mdoc app, for example, using the mdoc-openid4vp URL scheme and provides the Request URI as a URL query string parameter to the mdoc app.

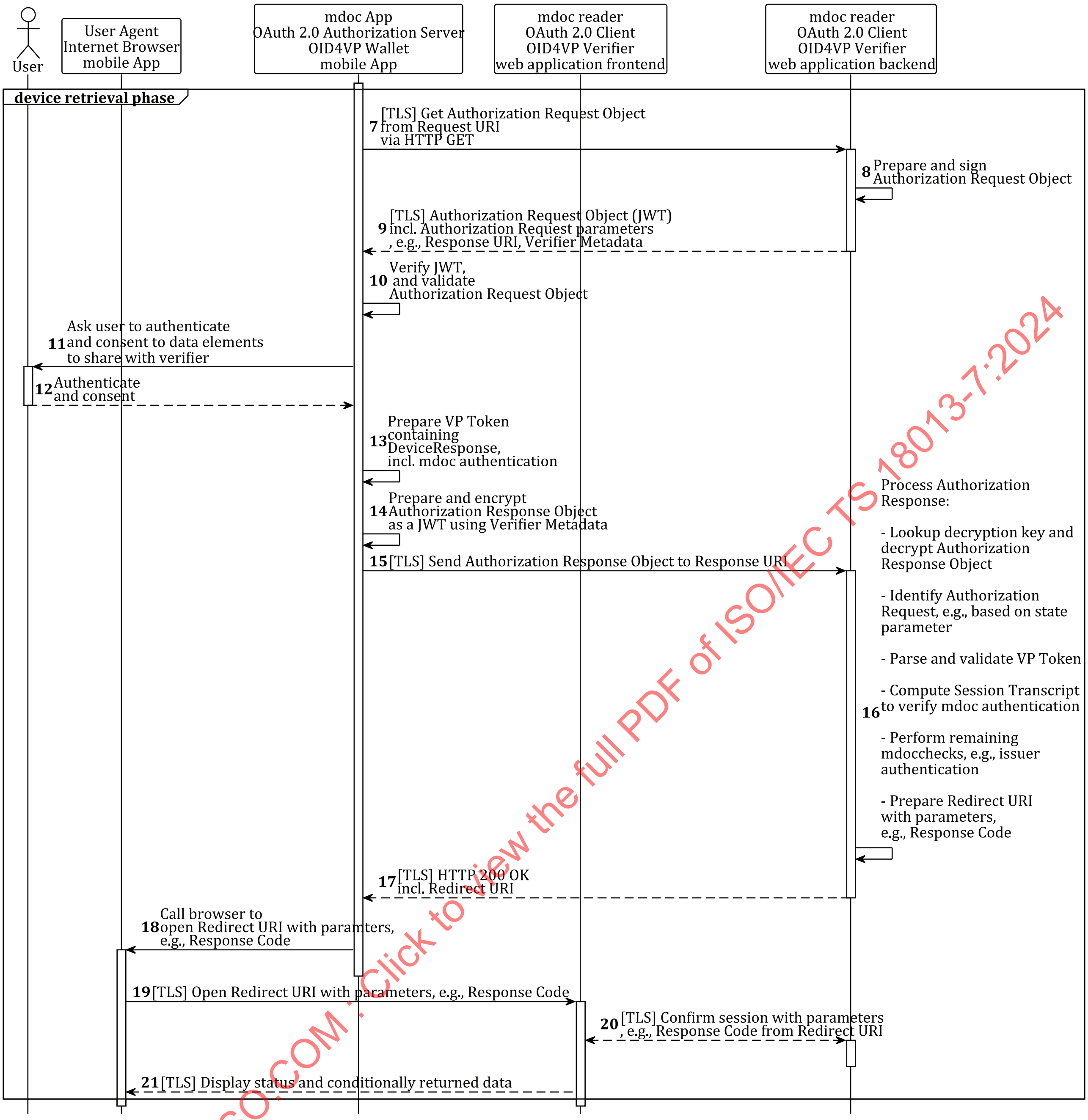
User
User Agent
Internet Browser
mobile App
mdoc App
OAuth 2.0 Authorization Server
OID4VP Wallet
mobile App
mdoc reader
OAuth 2.0 Client
OID4VP Verifier
web application frontend
mdoc reader
OAuth 2.0 Client
OID4VP Verifier
web application backend
device retrieval phase
7 [TLS] Get Authorization Request Object from Request URI via HTTP GET
8 Prepare and sign Authorization Request Object
9 [TLS] Authorization Request Object (JWT) incl. Authorization Request parameters , e.g., Response URI, Verifier Metadata
10 Verify JWT, and validate Authorization Request Object
11 Ask user to authenticate and consent to data elements to share with verifier
12 Authenticate and consent
13 Prepare VP Token containing DeviceResponse, incl. mdoc authentication
14 Prepare and encrypt Authorization Response Object as a JWT using Verifier Metadata
15 [TLS] Send Authorization Response Object to Response URI
16 Process Authorization Response:
- Lookup decryption key and decrypt Authorization Response Object
- Identify Authorization Request, e.g., based on state parameter
- Parse and validate VP Token
- Compute Session Transcript to verify mdoc authentication
- Perform remaining mdocchecks, e.g., issuer authentication
- Prepare Redirect URI with parameters, e.g., Response Code
17 [TLS] HTTP 200 OK incl. Redirect URI
18 Call browser to open Redirect URI with paramters, e.g., Response Code
19 [TLS] Open Redirect URI with parameters, e.g., Response Code
20 [TLS] Confirm session with parameters , e.g., Response Code from Redirect URI
21 [TLS] Display status and conditionally returned data

Figure B.2 — Device retrieval phase sequence diagram

During the device retrieval phase:

- In step 7, the mdoc app fetches the Authorization Request Object from the Request URI received using a HTTP GET over Transport Layer Security (TLS) from the Request URI that was received in the Authorization Request (see B.4.2.2).
- In step 8, the mdoc reader receives the fetch request and then prepares and signs the Authorization Request Object associated with the Request URI (see B.4.2)
- In step 9, the mdoc reader returns the Authorization Request Object to the mdoc app over TLS.
- In step 10, the mdoc app verifies the JWT signature and validates the Authorization Request Object which includes determining whether the mdoc reader is trustworthy according to OID4VP, Section 5 and in RFC 9101:2021, Section 6.2.

- In step 11, the mdoc app asks the user to authenticate and to consent to sharing the requested data elements with the mdoc reader.

  NOTE 1 This step can be implemented in different ways.

- In step 12, the user authenticates and gives consent.
- In step 13, the mdoc app performs mdoc authentication, generates the `DeviceResponse` and prepares the VP Token.
- In step 14, the mdoc app prepares the Authorization Response parameters and encrypts the Authorization Response Object as a JWE using the Verifier Metadata from the Authorization Request Object (see B.4.2.3.2).
- In step 15, the mdoc app sends the encrypted Authorization Response Object (JWT) to the Response URI provided in the Authorization Request Object.
- In step 16, the mdoc reader processes the Authorization Response Object.
  - First, the mdoc reader decrypts the JSON Web Encryption (JWE).
  - Then, the mdoc reader identifies the associated Authorization Request. This can be done by using a potential `state` Authorization Request parameter. Then, the mdoc reader verifies that the Authorization Response corresponds to the Authorization Request. This involves the verification of the provided nonce in the Authorization Response Object.
  - Then, the mdoc reader parses and validates the VP Token containing the `DeviceResponse`. This includes validation of contained document types and data elements.
  - Then, the mdoc reader computes the `SessionTranscript` to verify mdoc authentication.
  - Then, the mdoc reader performs other remaining checks such as issuer authentication.
  - Finally, the mdoc reader prepares a Redirect URI including parameters required to verify the session binding. This can be a Response Code associated with the session (see in OID4VP, Draft 18, section 11.2).
- In step 17, the mdoc reader returns the Redirect URI including parameters to the mdoc app over TLS.
- In step 18, the mdoc app directs the user agent to open the Redirect URI including potential parameters.
- In step 19, the user agent requests the Redirect URI including potential parameters from the mdoc reader over TLS. In this step, the user agent uses the existing session with the mdoc reader.
- In step 20, the mdoc reader receives the Redirect URI including potential parameters and confirms the request is associated with the existing session between the user agent and the mdoc reader. This could include the verification of additional parameters such as a Response Code. The mdoc reader looks up the status and potential data of the Authorization Response associated with the current Authorization Request (i.e. transaction) and session.
- In step 21, the mdoc reader returns the status and potential data to the user agent which then displays the data to the user.

NOTE 2 The AuthorizationRequest in the OID4VP protocol is sent directly from the request endpoint of the mdoc reader to the mdoc app without any intermediary applications. Since this message requires TLS for transport layer encryption, the DeviceRequest does not use application layer encryption. The exposure of DeviceRequest is limited to the connection between the TLS termination endpoint and the mDL Reader application.

B.2 Wallet invocation

The mdoc reader has the choice of the following mechanisms to invoke a mdoc app acting as a Wallet:

- custom URL scheme as an `authorization_endpoint` (e.g. `mdoc-openid4vp://` as defined in B.3.2.3.2);

— other mechanisms not described in this document.

B.3 Metadata exchange

B.3.1 General

OID4VP utilizes Wallet and Verifier Metadata, defined in OID4VP:2023, Section 8 and 9, to inform both the mdoc app and the mdoc reader about each other's capabilities for the data exchange. This information includes details such as supported signature and encryption algorithms, and more.

The subsequent sections outline specific requirements and default values for Wallet and Verifier Metadata, particularly when the mdoc reader lacks prior knowledge of the invoked mdoc app.

NOTE 1 The mdoc reader uses the Wallet Metadata, while the mdoc app relies on the Verifier Metadata at different stages in the OID4VP flow.

NOTE 2 Since the OID4VP specification references the OAuth2 specification, all requirements in the Oauth2 specification with regards to HTTP connection settings and parameters apply to this profile as well.

B.3.2 Wallet Metadata

B.3.2.1 General

Before generating and sending an Authorization Request to the mdoc app, an mdoc reader requires the Wallet Metadata.

B.3.2.2 Wallet Metadata parameters

An mdoc app and mdoc reader shall support the following Wallet Metadata parameters as defined in Table B.1.

Table B.1 — Wallet Metadata parameters

Parameter name	Value defined in this profile	Reference
`issuer`	The value for `issuer` is the Wallet issuer identifier.	[15]
`authorization_endpoint`	The value for `authorization_endpoint` is the OAuth 2 Authorization Endpoint where the mdoc reader sends the Authorization Request.	[15]
`response_types_supported`	The value for `response_types_supported` shall contain the `vp_token` Response Type as defined in OID4VP, Section 5.4.	[15]
`vp_formats_supported`	The value for `vp_formats_supported` shall contain the `mso_mdoc` Credential Format Identifier.	OID4VP, Draft 18, Section 8.1
`client_id_schemes_supported`	The value for `client_id_schemes_supported` shall contain the `x509_san_dns` Client Identifier scheme. For example, `[ "x509_san_dns" ]`.	OID4VP, Draft 18, Section 8.1
`request_object_signing_alg_values_supported`	The value for `request_object_signing_alg_values_supported` shall contain all signature algorithms from Table B.8	RFC 9101
`authorization_encryption_alg_values_supported`	The value for `authorization_encryption_alg_values_supported` shall contain ECDH-ES and support all curves defined in Table B.8 unless the mdoc reader knows the set of curves that the mdoc that it interacts with supports.	[5]
`authorization_encryption_enc_values_supported`	The value for `authorization_encryption_enc_values_supported` shall at least contain `A256GCM`. For example, `[ "A256GCM" ]`.	[5]

B.3.2.3 Obtaining Wallet Metadata

B.3.2.3.1 General

An mdoc reader utilizing this specification has multiple options to obtain the Wallet Metadata of an mdoc app, as defined in OID4VP:2023, Section 8.2:

— The mdoc reader has a static and pre-configured set of Wallet Metadata parameters of the mdoc app (see B.3.2.3.2).

— Alternatively, the mdoc reader can obtain the Wallet Metadata of the mdoc app dynamically, for example, using Reference [15] or other out-of-band mechanisms.

B.3.2.3.2 Static set of Wallet Metadata

The mdoc app shall support the following static set of Wallet Metadata parameters bound to the mdoc-OID4VP scheme (`mdoc-openid4vp://`):

```
{
  "issuer": "https://self-issued.me/v2",
  "authorization_endpoint": "mdoc-openid4vp://",
  "response_types_supported": [
    "vp_token"
  ],
  "vp_formats_supported": {
    "mso_mdoc": {}
  },
  "client_id_schemes_supported": [
    "x509_san_dns"
  ],
  "authorization_encryption_alg_values_supported": [
    "ECDH-ES"
  ],
  "authorization_encryption_enc_values_supported": [
    "A256GCM"
  ]
}
```

NOTE 1 `https://self-issued.me/v2` above is a symbolic string.

NOTE 2 In the static set of Wallet Metadata above, `mso_mdoc` does not contain any JSON members which indicates that the mdoc app can support any of the supported cryptographic algorithms for issuer and mdoc authentication defined in ISO/IEC 18013-5.

NOTE 3 If the mdoc reader has no prior knowledge of the mdoc app or if the mdoc reader cannot discover the Wallet Metadata dynamically, it is expected that the mdoc reader uses the static set of Wallet Metadata as defined above.

B.3.2.3.3 mdoc-openid4vp URL scheme

This document registers the mdoc-openid4vp URI scheme defined in the IANA Uniform Resource Identifier (URI) Schemes registry, defined in Reference [13]:2023, Section B.3.2.3.1.

— URI scheme name: mdoc-openid4vp

— Status: permanent

— Applications/protocols that use this scheme name: the mdoc-openid4vp URI scheme is intended to be used by mobile document reader applications.

— Change controller: ISO/IEC JTC1/SC17

— Reference: ISO/IEC TS 18013-7 (this document)

B.3.3 Verifier Metadata

B.3.3.1 General

An mdoc needs the Verifier Metadata when generating and sending an Authorization Response to the mdoc reader.

An mdoc shall support receiving Verifier Metadata using the `client_metadata` Authorization Request parameter, as defined in OID4VP:2023, Section 5.

Both the mdoc app and mdoc reader shall use the Verifier Metadata parameters from Table B.2.

Table B.2 — Verifier Metadata parameters

Parameter name	Value defined in this profile	Reference
`jwks`	The value for `jwks` contains the public key(s) of the mdoc reader. Each curve defined in Table B.8 shall be included unless the mdoc reader is aware of the set of keys supported by the mdoc it interacts with. The public keys are encoded as a JSON Web Key Set (JWKS) as defined in Reference [10]. The `jwks` shall contain the keys used for mdoc mac authentication as defined in ISO/IEC 18013-5. The keys used for this purpose shall set the “use” JWK paremeters to “enc” (see RFC 7515) and “kty”, “crv” to the corresponding values of “kty”, “crv” for each curve defined in Table B.8 unless the mdoc reader is aware of the set of keys supported by the mdoc it interacts with.	[12]
`authorization_encrypted_response_alg`	The value for `authorization_encrypted_response_alg` is used for the value in the `alg` JWT (JWE) header parameter in the encrypted Authorization Response.	[5]
`authorization_encrypted_response_enc`	The value for `authorization_encrypted_response_enc` is used for the value in the `enc` JWT (JWE) header parameter in the encrypted Authorization Response.	[5]
`vp_formats`	The value for `vp_formats` shall include a JSON object with the key `mso_mdoc`.	OID4VP:2023, Section 9.1

Additional mechanisms to obtain the Verifier Metadata and Verifier Metadata parameters may be supported but are out-of-scope of this specification.

The mdoc reader is not required to pre-register with the mdoc app.

B.4 Data Exchange

B.4.1 Data Model

The data model shall be used as specified in Clause 7.

B.4.2 Authorization request

B.4.2.1 General

The mdoc reader shall generate and deliver the Authorization Request as defined in OID4VP:2023, Section 5 to the mdoc and applies the additional rules of the OID4VP mdoc-specific profile defined in this document.

B.4.2.2 Engagement phase

The mdoc app shall support receiving the OID4VP Authorization Request at the `authorization_endpoint` specified by the Wallet Metadata. The Authorization Request shall pass an Authorization Request Object by reference using the `request_uri` parameter as defined in RFC 9101.

The value for `request_uri` contains the HTTPS-based URL where the mdoc retrieves the Authorization Request Object that contains the Authorization Request parameters (see RFC 9101).

The following is a non-normative example of an Authorization Request using the mdoc-openid4vp scheme:

```
mdoc-openid4vp://
 ?client_id=example.com
 &request_uri=https%3A%2F%2Fexample.com%2F567545564
```

The `client_id` Authorization Request parameter is included in the Authorization Request as a percent-encoded (as defined in Reference [6]:2023, Section 2.1) URL query string parameter. The `client_id` is also included in the Authorization Request Object as defined in RFC 9101.

B.4.2.3 Device retrieval phase

B.4.2.3.1 General

The mdoc app uses the `request_uri` Authorization Request parameter to retrieve the signed Authorization Request Object as defined in RFC 9101:2023, Section 5.2.3.

The following is a non-normative example of a HTTPS request to fetch the signed Authorization Request:

```
  GET /567545564 HTTP/1.1
  Host: example.com
```

The following is a non-normative example of the HTTPS response:

```
  HTTP/1.1 200 OK
  Date: Thu, 20 Aug 2020 23:52:39 GMT
  Server: Apache/2.4.43 (example.com)
  Content-type: application/oauth-authz-req+jwt
  Content-Length: 797
  Last-Modified: Wed, 19 Aug 2020 23:52:32 GMT

  eyJ4NW...
```

NOTE The example above does not show the entire HTTP response body to improve readability. The HTTP response body contains the signed Authorization Request Object encoded as a JWT as defined in B.4.2.3.4.

B.4.2.3.2 Authorization Request parameters

The mdoc reader shall include the Authorization Request parameters from Table B.3 in the Authorization Request Object.

The Authorization Request Object may contain the `state` Authorization Request parameter as defined in Reference [8].

NOTE As defined in Reference [8]:2012, Section 3.1 the mdoc must ignore other unrecognized Authorization Request parameters.

Table B.3 — Authorization Request parameters (passed in Request Object)

Parameter Name	Value defined in this profile	Reference
`response_type`	The value for `response_type` is a Response Type value that is included in the Wallet Metadata parameter `response_types_supported`. For example, `vp_token`.	[8]
`presentation_definition`	The value for `presentation_definition` is the Presentation Definition object that specifies the mdoc data being requested. B.4.2.3.3 defines the content of the Presentation Definition object for this mdoc-specific profile of OID4VP.	OID4VP:2023, Section 5.1
`client_metadata`	The value for `client_metadata` is the Verifier Metadata parameter of the mdoc reader as defined in this document.	OID4VP:2023, Section 5
`nonce`	The value for `nonce` is a cryptographic nonce value and shall follow the requirements in B.5.3.	OID4VP:2023, Section 5
`client_id`	The value for `client_id` is the Client Identifier of the mdoc reader that corresponds to the Client Identifier scheme.	[8]
`client_id_scheme`	The value for `client_id_scheme` is a Client Identifier Scheme that is included in `client_id_schemes_supported` Wallet Metadata parameter, e.g., `x509_san_dns`.	OID4VP:2023, Section 5
`response_mode`	The value for `response_mode` shall be `direct_post.jwt` as defined in OID4VP, clause 6.3.1.	[8]
`response_uri`	The value for `response_uri` is the HTTPS URL that represents the HTTPS POST endpoint for submitting the encrypted Authorization Response required by the Response Mode `direct_post.jwt`.	OID4VP:2023, Section 6.2
`aud`	The value for `aud` is the audience of the Authorization Request Object and is set to the `issuer` Wallet Metadata parameter.	OID4VP:2023, Section 5.6

An example of of Authorization Request parameters included in an Authorization Request Object using the `x509_san_dns` Client Identifier scheme is provided in B.6.2.

B.4.2.3.3 Presentation Definition

A Presentation Definition object (as defined in OID4VP:2023, Section 5) used by this OID4VP mdoc-specific profile shall contain the JSON members from Table B.4.

Table B.4 — Presentation Definition JSON members

Presentation Definition JSON member	Value defined in this profile
`id`	The value for `id` is a JSON String that is unique in the Presentation Definition of the Authorization Request.
`input_descriptors`	The value for `input_descriptors` is a JSON array of Input Descriptor objects. The Presentation Definition shall have at least one Input Descriptor object.

Each Input Descriptor object in a Presentation Definition object shall contain the JSON members from Table B.5.

Table B.5 — Presentation Definition JSON members

Input Descriptor JSON member	Value defined in this profile
`id`	The value for `id` shall be set to the requested document type. This indicates that all requested data elements shall be selected from that document type. For example, ISO/IEC 18013-5 defines `org.iso.18013.5.1.mDL` as the document type for mDL. The Input Descriptor `id` shall be unique per Presentation Definition object. This implies that a document type can only be used once within the Presentation Definition. Example: `"id": "org.iso.18013.5.1.mDL"`
`format`	The value for `format` is a JSON object which shall contain a JSON object with the key `mso_mdoc`. `mso_mdoc` shall contain a JSON member `alg` that contains a list of supported algorithms for issuer and mdoc authentication (as defined in ISO/IEC 18013-5) as listed in Table B.8.

`constraints`	The value for `constraints` is a JSON object with the following JSON members: `limit_disclosure`, `fields`.
	`limit_disclosure` whose value shall be set to the JSON String value `required`.
	`fields` is a JSON array of Field objects where each Field is a JSON object with the following JSON members: `path`, `intent_to_retain`.
	`path` is a JSON array where each entry is a JSON String containing a requested data element from the requested document type as follows: `$['<namespace>']['<data element identifier>']`. For example, to request a data element with an data element identifier `family_name` from the namespace `org.iso.18013.5.1`, the following JSON String is used: `$['org.iso.18013.5.1']['family_name']`.
	`intent_to_retain` whose value shall be set to `true` or `false`.
	Example:
	`"constraints": {`
	`  "limit_disclosure":"required",`
	`  "fields": [{`
	`    "path":[`
	`      "$['org.iso.18013.5.1']['family_name']"`
	`    ],`
	`    "intent_to_retain":true`
	`  }]`
	`}`

An mdoc and mdoc reader are only required to implement the syntax requirements related to the parameters of the Presentation Definition object that are specified in this document. An mdoc reader shall not include any parameters and values in the Presentation Definition object that are not defined in this document.

An example of an Authorization Request is provided in B.6.2.

When data elements from multiple document types are requested, separate Input Descriptor objects per document type shall be present.

B.4.2.3.4 Client Identifier scheme

The mdoc reader shall select a Client Identifier scheme for the Authorization Request that is supported by the Wallet Metadata.

B.4.2.3.5 Authorization Request signing

The mdoc reader shall select a Client Identifier scheme that requires Authorization Request signing.

If the selected Client Identifier scheme requires Authorization Request signing, e.g. `x509_san_dns`, the mdoc reader shall encode the Authorization Request parameters as a signed Authorization Request Object encoded as a JWT. The Authorization Request Object shall contain and set the Authorization Request parameter `require_signed_request_object` to `true`. The signed Authorization Request Object shall contain an alg JWT (JWS) header parameter that matches one of the supported signature algorithms specified by the request_object_signing_alg_values_supported Wallet Metadata parameter.

For the x509_san_dns Client Identifier scheme, the Client Identifier shall be a DNS name and match a dNSName Subject Alternative Name (SAN) according to RFC 5280 entry in the leaf certificate passed with the Authorization Request Object. The Authorization Request Object shall be signed with the private key corresponding to the public key in the leaf X.509 certificate of the certificate chain added to the x5c JWT (JWS) header parameter of the signed Authorization Request Object. The Wallet shall validate the signature and the trust chain of the X.509 certificate. All Verifier metadata other than the public key shall be obtained from the client_metadata parameter. If the Wallet can establish trust in the Client Identifier authenticated through the certificate, e.g. because the Client Identifier is contained in a list of trusted Client Identifiers, it may allow the client to freely choose the response_uri Authorization Request parameter value. If not, the FQDN of the response_uri value shall match the Client Identifier.

NOTE 1 The certificates included in the x5c JWT (JWS) header parameter are base64-encoded (RFC 4648:2006, Section 4) and not base64url-encoded. Further note that if the certificate chain includes only one certificate, the x5c JWT (JWS) header parameter is a JSON array with one entry.

NOTE 2 Since AuthorizationRequest in OID4VP is signed, a trust anchor is needed to enable trust in the signature. Self-signed mdoc reader certificates can support trust when they are already present in the mDL via an out-of-band manner (which effectively elevates such a certificate to a trust anchor) e.g. dynamic registration.

B.4.3 Authorization Response

B.4.3.1 General

The mdoc shall generate and deliver the Authorization Response as defined in OID4VP:2023, Section 6 to the mdoc reader and applies the additional rules of the OID4VP mdoc-specific profile defined in this document.

The Authorization Response is sent to the response_uri endpoint as defined in OID4VP:2023, Section 6.3.1 using the Response Mode direct_post.jwt. The Authorization Response is encoded as a JWT (JWE) without nesting (see B.4.3.3.2). The JWT (JWE) is included in the response parameter as defined in OID4VP:2023, Section 6.3.1.

The following is a non-normative example of an Authorization Response sent to the response_uri endpoint:

```
POST /post HTTP/1.1
  Host: client.example.org
  Content-Type: application/x-www-form-urlencoded

  response=eyJra...9t2LQ
```

NOTE The example above does not show the entire value for the response Authorization Response parameter to improve readability.

The response of the response_uri endpoint shall include the redirect_uri parameter as defined in OID4VP:2023, Section 6.2.

B.4.3.2 Authorization Response parameters

An Authorization Response shall include the Authorization Response parameters from Table B.6.

If present, the mdoc shall copy the state Authorization Request parameter from the Authorization Request Object to the Authorization Response Object (as defined in Reference [8]).

The Authorization Response is delivered encrypted to the mdoc reader encoded as a JWT using JWE compact serialization (see B.4.3.3.2).

Table B.6 — Authorization Response parameters

Parameter Name	Value defined in this profile	Reference
vp_token	The value for vp_token shall contain the base64url-encoded-without-padding DeviceResponse data structure as defined in ISO/IEC 18013-5.	OID4VP:2023, Section 6.1
presentation_submission	The value for presentation_submission shall contain the Presentation Submission object as described in B.4.3.3.	OID4VP:2023, Section 6.1

An example of the Authorization Response Object Parameters can be found in B.6.6.

B.4.3.3 Presentation Submission

B.4.3.3.1 General

A Presentation Submission object (as defined in OID4VP:2023, Section 6.1) used by this OID4VP mdoc-specific profile shall contain the JSON members from Table B.7.

Table B.7 — Presentation Submission JSON members

Presentation Submission JSON member	Value defined in this profile
id	The value for id is a String with a value that is unique in the Authorization Response.
definition_id	The value for definition_id corresponds to the id value of the Presentation Definition object in the Authorization Request.
descriptor_map	The value for descriptor_map shall be a JSON array of Descriptor Map objects with one or more entries where each entry corresponds to an Input Descriptor object in the Authorization Request. Each Descriptor Map is a JSON object with the following JSON members: id, format, path. The value for each id corresponds to the id value of the Input Descriptor object the Descriptor Map object corresponds to. The value for format shall be the static JSON String value mso_mdoc. The value for path shall be the static JSON String value $ if the VP Token contains a single JSON String or JSON object.

An example of a Presentation Submission object referring to a VP Token with a single entry representing a DeviceResponse with a single document type can be found in B.6.4.

B.4.3.3.2 Authorization Response encryption

The mdoc shall encode the Authorization Response as a JWT (JWE only) as defined in OID4VP:2023, Section 6.3. The JWT uses the JWE compact serialization as defined in Reference [9]:2015, Section 3.1.

NOTE The JWT does not contain a nested JWS.

To generate the JWE, the mdoc shall use the jwks, authorization_encrypted_response_alg and authorization_encrypted_response_enc from the Verifier Metadata. The JWE shall encrypt a JSON Object containing the Authorization Response parameters from Table B.6. The mdoc shall generate a new JWE Initialization Vector (IV) for each encryption operation.

For example, if the authorization_encrypted_response_alg is set to ECDH-ES and authorization_encrypted_response_enc is set to A256GCM, this means Eliptic Curve Diffie-Hellman in Direct Key Agreement mode as specified in Reference [11] is used to encrypt the payload of the JWE.

The jwks Authorization Request parameter is used to convey the ephemeral public key information of the mdoc reader to the mdoc app which is required for key agreement. The mdoc app generates a new ephemeral key pair and sets the value for the epk JWT (JWE) header parameter to the public key of the generated key pair-encoded as a JSON Web Key (see Reference [10]). When generating the ephemeral key pair, the mdoc app has to ensure that the curve of the epk matches the curve of the mdoc reader public key specified by the crv and kty JSON members (as defined in Reference [10]) of the JWK.

The mdoc reader shall set the use JWK parameter (public key use) to the static JSON String value enc and set the alg JWK parameter to the static JSON String value ECDH-ES to indicate which JWK in the jwks Authorization Request parameter can be used for key agreement to encrypt the response (see Reference [10]).

The mdoc app shall support at least one of the cryptographic curves as defined in Table B.8. The mdoc reader shall support all of the cryptographic curves defined in Table B.8.

The mdoc shall set the apu JWT (JWE) header parameter to the base64url-encoded-with-no-padding value of the mdocGeneratedNonce of the SessionTranscript as defined in B.4.4.

The mdoc shall set the apv JWT (JWE) header parameter to the base64url-encoded-with-no-padding value of the utf-8 encoded nonce Authorization Request parameter from the Authorization Request Object.

The mdoc shall set the kid JWT (JWE) header parameter to the value of the kid JWK parameter of the public key that was used for key agreement to encrypt the response.

An example of the JWT (JWE) header of an encrypted Authorization Response can be found in B.6.8.

An example of an encrypted Authorization Response encoded as a JWT (JWE) can be found in B.6.7.

B.4.3.4 Error Handling

Error handling shall be supported as defined in OID4VP.

B.4.4 Session Transcript

The SessionTranscript as defined in ISO/IEC 18013-5 shall be used with the following changes:

— DeviceEngagementBytes is replaced with null,

— EReaderKeyBytes is replaced with null

The Handover element is defined as:

```
Handover = OID4VPHandover
OID4VPHandover = [
  clientIdHash
  responseUriHash
  nonce
]

clientIdHash = bstr
responseUriHash = bstr
clientIdToHash = [clientId, mdocGeneratedNonce]
responseUriToHash = [responseUri, mdocGeneratedNonce]

mdocGeneratedNonce = tstr
clientId = tstr
responseUri = tstr
nonce = tstr
```

where clientIdHash is the SHA-256 hash of clientIdToHash and responseUriHash is the SHA-256 hash of the responseUriToHash.

The mdoc app shall set the value for mdocGeneratedNonce to a cryptographically random number with sufficient entropy (see B.5.3).

clientId shall be the client_id, responseUri shall be the response_uri and nonce shall be the nonce Authorization Request parameter from the Authorization Request Object.

An example of the SessionTranscript can be found in B.6.9.

B.4.5 mdoc MAC authentication

To perform mdoc MAC authentication as defined in ISO/IEC 18013-5, the mdoc shall use on of the ephemeral public keys from the “jwks” Verifier Metadata parameter as the “EReaderKey.Pub” with having the “use” JWK parameter set to “enc”.

NOTE “EReaderKey.Pub” acts as input to ECKA-DH to compute the shared secret to derive the MAC Key.

B.5 Security mechanisms

B.5.1 General

The following security architecture goals from 6.4.4.1 apply:

- Security mechanisms a), b) and c) apply directly.
- Security mechanism d) applies by sending the Authorization Response encrypted to the mdoc reader.
- The security mechanism defined in e) does not apply, however the protection against relayed engagement information is still achieved. The mdoc reader must maintain a binding between the user session bound and the `nonce` Authorization Request parameter. When the end-user is redirected back to the mdoc reader with the same `nonce` (included in `SessionTranscript`), the mdoc reader can detect whether there was no MITM attack.

B.5.2 Cryptographic curves

Besides the requirements for cryptographic algorithms from ISO/IEC 18013-5 for the mdoc and mdoc reader.

An mdoc app and mdoc reader shall use one of the ECDH and/or signature algorithms from Table B.8 for Request signing and Response encryption / decryption. The mdoc reader shall support all curves and algorithms defined in Table B.8 for response object decryption.

Table B.8 — Cryptographic curves

Definition	Specification	Key type (`kty` value)	Signature algorithm (`alg` value)	Curve identifier (`crv` value)	Purpose
NIST P-256	Reference [11]	`EC`	`ES256`	`P-256`	ECDSA/ECDH
NIST P-384	Reference [11]	`EC`	`ES384`	`P-384`	ECDSA/ECDH
NIST P-521	Reference [11]	`EC`	`ES512`	`P-521`	ECDSA/ECDH
BrainpoolP256r1	Reference [7]	`EC`	`ESB256` (with SHA-256)	`BP-256`	ECDSA/ECDH
BrainpoolP320r1	Reference [7]	`EC`	`ESB320` (with SHA-384)	`BP-320`	ECDSA/ECDH
BrainpoolP384r1	Reference [7]	`EC`	`ESB384` (with SHA-384)	`BP-384`	ECDSA/ECDH
BrainpoolP512r1	Reference [7]	`EC`	`ESB512` (with SHA-512)	`BP-512`	ECDSA/ECDH
Curve25519	Reference [14]	`OKP`	`EdDSA`	`Ed25519/X25519`	EdDSA/ECDH
Curve448	Reference [14]	`OKP`	`EdDSA`	`Ed448/X448`	EdDSA/ECDH

B.5.3 Entropy of the nonce

The mdoc `DeviceResponse` is securely bound to a particular session based on the fact that the nonces are not learned or guessed by the attacker. As such, nonces shall be an unpredictable random or pseudorandom value. Nonces shall have a minimum entropy of 16 bytes. A new nonce value shall be chosen for each transaction.

NOTE This applies to `nonce` and `mdocGeneratedNonce`.

B.6 Examples

B.6.1 OID4VP example — Presentation Definition

The following is an example of the Presentation Definition.

```
--------------------------------
Example: Presentation Definition
--------------------------------
{
  "id": "mDL-sample-req",
  "input_descriptors": [
    {
      "id": "org.iso.18013.5.1.mDL",
      "format": {
        "mso_mdoc": {
          "alg": [
            "ES256",
            "ES384",
            "ES512",
            "EdDSA",
            "ESB256",
            "ESB320",
            "ESB384",
            "ESB512"
          ]
        }
      },
      "constraints": {
        "fields": [
          {
            "path": [
              "$['org.iso.18013.5.1']['birth_date']"
            ],
            "intent_to_retain": false
          },
          {
            "path": [
              "$['org.iso.18013.5.1']['document_number']"
            ],
            "intent_to_retain": false
          },
          {
            "path": [
              "$['org.iso.18013.5.1']['driving_privileges']"
            ],
            "intent_to_retain": false
          },
          {
            "path": [
              "$['org.iso.18013.5.1']['expiry_date']"
            ],
            "intent_to_retain": false
          },
          {
            "path": [
              "$['org.iso.18013.5.1']['family_name']"
            ],
            "intent_to_retain": false
          },
          {
            "path": [
              "$['org.iso.18013.5.1']['given_name']"
            ],
            "intent_to_retain": false
          },
          {
            "path": [
              "$['org.iso.18013.5.1']['issue_date']"
            ],
            "intent_to_retain": false
          },
          {
            "path": [
              "$['org.iso.18013.5.1']['issuing_authority']"
            ],
            "intent_to_retain": false
          },
          {
            "path": [
              "$['org.iso.18013.5.1']['issuing_country']"
            ],
            "intent_to_retain": false
          },
          {
            "path": [
              "$['org.iso.18013.5.1']['portrait']"
            ],
            "intent_to_retain": false
          },
          {
            "path": [
              "$['org.iso.18013.5.1']['un_distinguishing_sign']"
            ],
            "intent_to_retain": false
          }
        ],
        "limit_disclosure": "required"
      }
    }
  ]
}
```

B.6.2 OID4VP example — Authorization Request Object parameters

The following is an example of the Authorization Request Object parameters. In this example and subsequent examples, the mdoc reader only contains a key for a single curve in the jwks structure. This is permissible only because for these examples it is assumed the mdoc reader is aware of the set of keys supported by the mdoc with which it is interacting.

```
------------------------------------------------
Example: Authorization Request Object parameters
------------------------------------------------
{
  "aud": "https://self-issued.me/v2",
  "response_type": "vp_token",
  "presentation_definition": {
    "id": "mDL-sample-req",
    "input_descriptors": [
      {
        "id": "org.iso.18013.5.1.mDL",
        "format": {
          "mso_mdoc": {
            "alg": [
              "ES256",
              "ES384",
              "ES512",
              "EdDSA",
              "ESB256",
              "ESB320",
              "ESB384",
              "ESB512"
            ]
          }
        },
        "constraints": {
          "fields": [
            {
              "path": [
                "$['org.iso.18013.5.1']['birth_date']"
              ],
              "intent_to_retain": false
            },
            {
              "path": [
                "$['org.iso.18013.5.1']['document_number']"
              ],
              "intent_to_retain": false
            },
            {
              "path": [
                "$['org.iso.18013.5.1']['driving_privileges']"
              ],
              "intent_to_retain": false
            },
            {
              "path": [
                "$['org.iso.18013.5.1']['expiry_date']"
              ],
              "intent_to_retain": false
            },
            {
              "path": [
                "$['org.iso.18013.5.1']['family_name']"
              ],
              "intent_to_retain": false
            },
            {
              "path": [
                "$['org.iso.18013.5.1']['given_name']"
              ],
              "intent_to_retain": false
            },
            {
              "path": [
                "$['org.iso.18013.5.1']['issue_date']"
              ],
              "intent_to_retain": false
            },
            {
              "path": [
                "$['org.iso.18013.5.1']['issuing_authority']"
              ],
              "intent_to_retain": false
            },
            {
              "path": [
                "$['org.iso.18013.5.1']['issuing_country']"
              ],
              "intent_to_retain": false
            },
            {
              "path": [
                "$['org.iso.18013.5.1']['portrait']"
              ],
              "intent_to_retain": false
            },
            {
              "path": [
                "$['org.iso.18013.5.1']['un_distinguishing_sign']"
              ],
              "intent_to_retain": false
            }
          ],
          "limit_disclosure": "required"
```