PKOC SPECIFICATION

PKOC Core Specification

Public Key Open Credential — transport-independent core

Version: 2.0.1

Status / Classification: Public — Approved for open implementation

Date: August 13, 2026

Maintained by: Physical Security Interoperability Alliance (PSIA)

Document Revision History

Version	Date	Author	Description
1.0.0	Jan 10, 2022	PSIA SCI Working Group	Initial PKOC credential concept and public-key model.
1.1.0	Dec 8, 2023	PSIA SCI Working Group	Consolidated credential and derived-identifier definitions across transports.
2.0.0	Jun 1, 2026	PSIA SCI Working Group	Restructured PKOC into a transport-independent Core plus transport Profiles. Unified glossary, cryptographic foundations, registration model, and common requirements previously duplicated in the BLE and NFC specifications.
2.0.1	Aug 13, 2026	PSIA SCI Working Group	Editorial harmonization pass, and incorporation of two previously separate specifications as normative Core sections: the PKOC Credential Specification (credential and derived-identifier derivation, now Section 4) and the PKOC-CVC Attestation Certificate Specification (now Section 5). Standardized front-matter order (Revision History → Contents → Glossary → Conformance), added the profile-extensibility framework and the registration sequence diagram, and aligned requirement language and formatting with the BLE and NFC Transport Profiles. All profiles now reference a single source for credential derivation and attestation. Stated explicitly that the Validated trust model and the PKOC-CVC are optional and that PKOC conformance is vendor-neutral and royalty-free, and reaffirmed the unconditional NIST P-256 Standard-mode baseline. No change is made to any wire protocol or to any credential derivation, so all existing PKOC implementations remain conformant.

Table of Contents

Glossary

The following terms are used throughout the PKOC Core Specification and all PKOC Transport Profiles. Profile documents add only the terms specific to their transport and otherwise defer to this glossary.

Term	Definition
Attestation Certificate	A certificate in which an issuer signs a binding between a subject public key and the device that holds the corresponding private key in non-extractable form, attesting that use of the subject key requires possession of that device. Realized in PKOC as the PKOC-CVC (Section 5).
Credential	The party that holds a PKOC key pair and presents it to a Reader. Realized as a mobile Device (BLE) or a smart card (NFC). Referred to as the Device or Card in the relevant profile.
CSPRNG	Cryptographically Secure Pseudo-Random Number Generator.
DER	Distinguished Encoding Rules; a binary encoding for ASN.1 structures, including ECDSA signatures produced by many libraries and the SubjectPublicKeyInfo and PKOC-CVC structures.
EAC-CVC	The Extended Access Control Card Verifiable Certificate format (BSI TR-03110-3; ICAO Doc 9303 Part 11) on which the PKOC-CVC profile is based.
ECDSA	Elliptic Curve Digital Signature Algorithm.
ECDHE	Elliptic Curve Diffie-Hellman Ephemeral; ECDH using single-use key pairs. Used by the BLE profile for Perfect Forward Secrecy.
IIR	Issuer Identification Reference; a 16-character identifier (CVC tag 42) naming the Issuer Key used to verify a PKOC-CVC signature (Section 5.4).
NIST P-256	The 256-bit prime-field elliptic curve defined in FIPS 186-5, also known as secp256r1 or prime256v1. The mandatory baseline curve for PKOC.
PACS	Physical Access Control System — the panel, controller, and backend that make access decisions.
PFS	Perfect Forward Secrecy; compromise of long-term keys does not expose past session traffic.
PKOC	Public Key Open Credential.
PKOC Credential	A 32-octet string derived from a PKOC public key, output from the Reader to the PACS (Section 4).
PKOC-CVC	PKOC Card Verifiable Certificate; the PKOC attestation-certificate format defined in Section 5, used to realize the Validated trust model.
PKOC Derived Identifier	An octet string of 8 to 31 octets derived from the PKOC Credential, output from the Reader to the PACS (Section 4).
PKOC Private Key	The Credential's long-term EC private key, held only in the Credential's secure storage and never transmitted.
PKOC Public Key	The Credential's long-term EC public key, presented to the Reader during a transaction.
PKOC Validated Credential	A 32-octet string derived from the public key attested by the PKOC-CVC, output from the Reader to the PACS (Section 4.6).
PKOC Validated Derived Identifier	An octet string of 4 to 31 octets derived from the PKOC Validated Credential, output from the Reader to the PACS (Section 4).

Term	Definition
Profile	A transport-binding specification (e.g., BLE, NFC) that maps the transport-independent PKOC Core onto a concrete wire protocol.
Provisioning / Issuer Backend	The service that enrolls credentials, records public keys with the PACS, and distributes trust anchors. Called the Provisioning Server (BLE) or Card Issuer (NFC).
Reader	The access-control endpoint that challenges a Credential, verifies its signature, and reports the result to the PACS.
SPKI	SubjectPublicKeyInfo; the ASN.1/DER structure (algorithm identifier plus key material) hashed to derive a PKOC Credential V2 (Section 4.4).
Standard (Enrollment) Trust	A trust model where a Credential is trusted because its public key was enrolled into the PACS. No attestation certificate is required.
Trust Anchor	An authoritative public key a Reader or Credential uses to validate an attestation over another party's key. Bound to a transport-specific artifact by each profile (Reader Certificate in BLE, PKOC-CVC Issuer Key in NFC).
Validated (Attested) Trust	A trust model where a Credential or Reader key is validated against a Trust Anchor via a signed attestation, enabling verification without prior enrollment of the individual key.

Conformance

The key words **MUST**, **MUST NOT**, **REQUIRED**, **SHALL**, **SHALL NOT**, **SHOULD**, **SHOULD NOT**, **RECOMMENDED**, **MAY**, and **OPTIONAL** in this document are to be interpreted as described in RFC 2119.

An implementation is conformant to a given profile only if it satisfies every **MUST** and **MUST NOT** requirement defined by the PKOC Core Specification and by that profile. Requirements are highlighted in **bold** throughout this document.

The Standard trust model is the whole of what PKOC conformance requires. The Validated trust model (Section 2.3), the PKOC-CVC (Section 5), Trust Anchors, and Issuer Keys are **OPTIONAL** capabilities that a Transport Profile **MAY** realize and that an implementation **MAY** elect to support. Where this Core or any Transport Profile states a **MUST**, **MUST NOT**, **SHALL**, or **SHALL NOT** in connection with the Validated model, that requirement is conditional on the implementation having elected to support it, and is never in itself a condition of PKOC conformance. An implementation that supports only Standard Mode is fully PKOC conformant.

PKOC conformance is vendor-neutral. No requirement in this Core or in any Transport Profile may be satisfiable only through a particular vendor's product, applet, chip, card operating system, certificate hierarchy, key-distribution service, registry, or intellectual property. PKOC is published for open, royalty-free implementation by any party, and any Credential, Reader, or Issuer meeting the requirements of this Core and a Transport Profile **MUST** be accepted as conformant regardless of its supplier.

The PKOC-CVC is the only attestation certificate. Where a profile realizes the Validated trust model over an attested Credential, the attestation **MUST** be a PKOC-CVC as defined in Section 5, signed by a PKOC Issuer Key. A profile **MUST NOT** define, require, or accept any other certificate format as a PKOC trust artifact. A certificate issued by a card manufacturer, chip vendor, or personalizer **MAY** be verified by an Issuer at personalization time, but **MUST NOT** be required by a Reader at transaction time and **MUST NOT** be a condition of PKOC conformance.

Backward compatibility is preserved. Version 2.0.1 introduces no change to any wire protocol and no change to any credential or derived-identifier derivation. A given public key yields the same PKOC Credential and the same PKOC Derived Identifier as it did under the specifications this edition consolidates,

so implementations built to PKOC NFC v1.1 or PKOC BLE v3.1.2 remain conformant without modification and enrolled credential records do not need to be re-derived or re-enrolled.

1. Introduction

1.1 Purpose and Scope

PKOC (Public Key Open Credential) is an open, royalty-free credential standard that uses public-key cryptography to authenticate a Credential to a Reader in a Physical Access Control System (PACS). This Core Specification defines the concepts, cryptography, credential model, attestation-certificate format, registration process, and common requirements that are **identical across every transport**. Transport-specific behavior — how PKOC is carried over a particular link such as Bluetooth Low Energy or NFC — is defined in a separate **Transport Profile** that references this Core.

This document is normative for all PKOC implementations. Where a Transport Profile and this Core appear to conflict on a transport-independent matter, this Core governs.

1.2 PKOC Objectives

Every PKOC profile is designed to satisfy the following objectives:

- **Interoperability.** Any PKOC-compliant Credential can be verified by any PKOC-compliant Reader for the same profile, regardless of manufacturer.
- **Openness and vendor neutrality.** PKOC is published for open, royalty-free implementation. Everything needed to build a conforming Credential, Reader, or Issuer is specified in this Core and its Transport Profiles, and conformance never depends on a particular vendor's product, certificate hierarchy, service, registry, or licence.
- **Backward compatibility.** A new edition does not invalidate a deployed one. Existing wire protocols and credential derivations are carried forward unchanged, and new capabilities are added as optional elections rather than as new obligations.
- **Security.** Authentication is based on asymmetric cryptography (ECDSA on NIST P-256 as the baseline). The PKOC private key never leaves the Credential's secure storage, and no shared secrets are distributed to Readers in the Standard trust model.
- **Privacy.** The protocol minimizes disclosure of personally identifiable information. A Reader learns the Credential's public key (or a derived identifier), not the identity of the holder.
- **Forward secrecy (profile-dependent).** Where a profile defines an ephemeral key-agreement flow, compromise of long-term keys does not expose past sessions.
- **Migration-friendly deployment.** PKOC works alongside existing PACS infrastructure and can emit legacy identifier formats, minimizing or eliminating panel and backend changes.

1.3 Changes in Version 2.0.1

Version 2.0 restructured PKOC from two independently-authored transport specifications into a single transport-independent Core plus a set of Transport Profiles. Content that had been duplicated — the glossary, cryptographic foundations, key encodings, the credential registration model, and the common Credential/Reader requirements — now lives here once and is referenced by every profile.

Version 2.0.1 is an editorial harmonization release that also **incorporates two previously separate specifications** so that all profiles draw on a single normative source. The credential and derived-identifier derivation methods (formerly the PKOC Credential Specification) are now **Section 4**, and the PKOC-CVC attestation-certificate format (formerly the PKOC-CVC Attestation Certificate Specification) is now **Section 5**. This edition also standardizes the front-matter order used by all PKOC documents, introduces the profile-extensibility framework (Section 8) that governs how new transports such as DUOX are added, adds the registration sequence diagram (Section 6.2), and aligns requirement wording and document formatting across the Core, BLE, and NFC documents so the set reads as a single coherent standard.

2. Architecture

This section defines the transport-independent architecture shared by all PKOC profiles. A profile maps these roles and concepts onto a concrete transport; it does not redefine them.

2.1 Roles

Role	Description
Credential	Holds a PKOC key pair and proves possession of the private key on demand. A mobile **Device** in the BLE profile; a smart **Card** in the NFC profile.
Reader	Challenges the Credential, verifies the returned signature, applies trust-model checks, and reports the outcome to the PACS.
PACS	The panel/controller/backend that stores credential records and renders access decisions. PKOC replaces the physical token, not the PACS.
Provisioning / Issuer Backend	Enrolls credentials, records public keys with the PACS, and distributes trust anchors (and, where applicable, revocation data) to Credentials and Readers.

2.2 The PKOC Credential

A PKOC Credential is fundamentally an EC key pair. The **private key** is generated on, and never leaves, the Credential's hardware-backed secure storage. The **public key** is the credential's identity: it is enrolled with the PACS at registration and presented to Readers at transaction time. Because authentication is a proof of possession of the private key over a fresh, reader-supplied challenge, a captured transaction cannot be replayed and no secret is exposed to the Reader.

From a PKOC public key, this Core derives a fixed 32-octet **PKOC Credential** and, where a PACS requires a compact identifier, a shorter **PKOC Derived Identifier**. Both derivations are transport-independent and are defined in Section 4.

2.3 Operating Modes and Trust Models

PKOC supports two trust models. Every profile realizes at least the Standard model and **MAY** realize the Validated model. The terminology below is common; each profile names the concrete artifacts that implement it.

Trust model	How trust is established	Typical use
Standard (Enrollment)	The Credential's public key is enrolled directly into the PACS. A Reader verifies proof-of-possession and the PACS recognizes the enrolled key. No attestation certificate is required.	Baseline interoperable access. Mandatory in every profile.
Validated (Attested)	The Credential's (or Reader's) key is validated against a **Trust Anchor** via a signed attestation, so a party can be verified without prior enrollment of that individual key.	Attestation, legacy-identifier issuance, and bounding the blast radius of key compromise.

> **Note:** The Validated model is realized differently by each transport, but the underlying concept — an authoritative Trust Anchor validating a signed attestation — is identical. In the NFC profile the attestation is a card-bound PKOC-CVC (Section 5); in the BLE profile it is a Site Issuer-signed Reader Certificate that authenticates the Reader to the Credential. Profiles bind the abstract Trust Anchor to these concrete artifacts.
>
> The Validated model is **OPTIONAL** in every profile. It is elected by a deployment, and no Credential, Reader, or PACS is required to support it in order to be PKOC conformant. Any organization **MAY** operate its own Trust Anchor and issue its own attestations: PKOC defines no central issuing authority, no registry, and no approval process, and no vendor holds a privileged position in the trust model.

2.4 Transport Profiles and Extensibility

A Transport Profile binds this Core to a specific link technology. The BLE and NFC profiles are the first two; additional profiles (for example, a future DUOX profile) attach to the same Core without changing it. The rules governing profiles are defined normatively in Section 8.

3. Cryptographic Foundations

These cryptographic rules apply to every PKOC profile. A profile **MAY** extend the permitted algorithm set for its Validated model (for example, the NFC profile permits additional curves and RSA/ML-DSA for issuer attestations), but the baseline defined here is mandatory.

3.1 Elliptic Curve Baseline

1. Every PKOC Credential **MUST** support the NIST P-256 curve (secp256r1 / prime256v1) for its Standard-mode key pair.
2. Private keys are 32-byte scalars. Public keys are points on the curve, encoded per Section 3.3.
3. A profile **MAY** permit additional curves or algorithms for attestation or Validated-mode keys; such extensions **MUST** be specified in the profile and **MUST NOT** weaken the P-256 baseline required for Standard-mode interoperability.
4. The P-256 Standard-mode baseline is unconditional. A profile **MUST NOT** define Standard-mode operation over a Credential whose key pair is not on NIST P-256, and **MUST NOT** define a mode of operation in which interoperability depends on a key pair, curve, or attestation controlled by a single supplier.

3.2 Digital Signatures

1. All baseline ECDSA signatures **MUST** use NIST P-256 with SHA-256 (ECDSA-SHA256), per FIPS 186-5.
2. Signatures on the wire **MUST** be transmitted in raw R ‖ S format: 64 bytes total, R and S each a 32-byte big-endian unsigned integer.
3. Implementations **MUST NOT** transmit ASN.1/DER-wrapped signatures. Any DER encoding produced by a library **MUST** be converted to raw R ‖ S before transmission (Section 3.3).

3.3 Key Formats and Encoding

EC public keys are encoded in one of two point formats:

- **Uncompressed:** `0x04` ‖ `X (32 bytes)` ‖ `Y (32 bytes)` — 65 bytes.
- **Compressed:** `0x02` (even Y) or `0x03` (odd Y) followed by `X (32 bytes)` — 33 bytes.

A DER-encoded ECDSA signature has the structure:

```
30 <len> 02 <r_len> [00] <r_bytes> 02 <s_len> [00] <s_bytes>
```

To convert to raw R ‖ S, strip the SEQUENCE and INTEGER tags and lengths, remove any leading `0x00` padding byte (present when the high bit of R or S is set), then left-pad R and S to exactly 32 bytes each and concatenate them.

3.4 Randomness

1. All private keys, ephemeral keys, challenges, and nonces **MUST** be generated with a CSPRNG.
2. Where the Credential hardware supports it, the Standard-mode private key **MUST** be generated inside, and be non-exportable from, a hardware-backed secure element.

4. PKOC Credentials and Derived Identifiers

This section defines how a **PKOC Credential** and optional **PKOC Derived Identifiers** are derived from a public key. The derivation is transport-independent: a given public key yields the same PKOC Credential and the same Derived Identifiers whether presented over BLE, NFC, or any future transport.

The output that the Reader sends to the PACS can be the full PKOC Credential, or a shorter PKOC Derived Identifier truncated or generated from the PKOC Credential.

4.1 Credential and Identifier Definitions

1. A PKOC Credential **MUST** be a 32-octet string derived from a public key per this section.
2. A PKOC Derived Identifier **MUST** be an octet string derived by truncating a PKOC Credential (Section 4.5), with a minimum length of 8 octets and a maximum length of 31 octets.
3. A PKOC Validated Derived Identifier **MUST** be an octet string derived by truncating a PKOC Credential, with a minimum length of 4 octets and a maximum length of 31 octets (see Section 4.6 for the conditions on lengths below 8 octets).
4. A PKOC Derived Identifier length **MUST** be an integer number of octets.

The following terms apply in this section: an **octet string** is a sequence of 8-bit octets; **big-endian** means most-significant byte first; and **lower N bits** means the least-significant N bits of the big-endian integer value.

4.2 Credential Representations

A PKOC Credential or Derived Identifier is a binary octet string. When it must be represented in text, a web context, decimal, or a data object, the following representations apply.

4.2.1 Base16 (Hex)

In textual form the value **MUST** be encoded using Base16 (hex) per RFC 4648 Section 8, representing the full octet-string value with no prefix (for example, no `0x`).

4.2.2 Base64url (Web)

For use in URLs or web contexts the value **MUST** be encoded using Base64url per RFC 4648 Section 5, with padding characters (=) omitted.

4.2.3 Decimal

In decimal form the value **MUST** be interpreted as an unsigned integer in big-endian byte order (OS2IP, RFC 8017 Section 4.2) and encoded in base 10 using ASCII digits `0`–`9` with no leading zeros, except that the value zero **MUST** be represented as `0`.

4.2.4 BER-TLV

As a Tag-Length-Value data object (ISO/IEC 7816-4 Section 6.3 BER-TLV), the Tag **MUST** be `04` (ASN.1 OCTET STRING); the Value **MUST** be the credential or identifier octet string. The Value **MUST NOT** be a SEQUENCE.

4.2.5 OID / Value (Discretionary Data Template)

As an OID/Value pair the object **MUST** be encoded as an ASN.1 sequence — a Discretionary Data Template (DDT) — as follows:

Field	Tag	ASN.1 type
Discretionary Data Template	7F4E	SEQUENCE
└ PKOC OID	06	OBJECT IDENTIFIER
└ PKOC Value	53	OCTET STRING

The DDT **MUST** contain exactly two data objects in the order shown: the PKOC OID object followed by the PKOC Value object. The DDT length **MUST** equal the total encoded length of those two objects. The PKOC OID object **MUST** use tag `06` and **MUST** contain the applicable encoded OID value from the table below. The PKOC Value object **MUST** use tag `53`, **MUST** contain the PKOC Credential or PKOC Derived Identifier octets directly, and **MUST NOT** contain a Base16 textual encoding of those octets.

The arc and DER encoding of each OID are:

PKOC type	OID arc	Encoded OID
PKOC Credential	1.3.6.1.4.1.65071.12.1	2B0601040183FC2F0C01
PKOC Derived Identifier	1.3.6.1.4.1.65071.12.2	2B0601040183FC2F0C02

4.3 PKOC Credential V1: ECC P-256 Derivation

PKOC Credential V1 is a deterministic derivation from a public key on NIST secp256r1 (P-256). The PKOC public key **MUST** be encoded in SEC1 uncompressed point form as a 65-octet string: octet 0 is `0x04` (uncompressed-point indicator per ANSI X9.62); octets 1–32 are the X coordinate; octets 33–64 are the Y coordinate. The key **MUST** represent a valid point on secp256r1, and compressed encodings (`0x02`/`0x03`) **MUST NOT** be used.

1. For PKOC V1, the PKOC Credential **MUST** be derived by extracting the 32-octet X coordinate (octets 1–32) of the uncompressed SEC1 public key.
2. The extracted 32-octet value **MUST** be used directly as the PKOC Credential without modification.

4.4 PKOC Credential V2: Hashed Public Key

PKOC Credential V2 applies to keys not covered by V1: ECC public keys on any supported curve other than NIST P-256 (for example P-384, P-521, or Brainpool); RSA public keys of any supported size (2048, 3072, 4096 bits); and other public-key algorithms for which a canonical DER-encoded SubjectPublicKeyInfo (SPKI) representation is defined. A NIST P-256 (secp256r1) public key **MUST** be derived using V1 (Section 4.3) and **MUST NOT** use V2, so that a given key yields a single canonical PKOC Credential and remains consistent with the v1.1 baseline.

The PKOC public key **MUST** be encoded as a SubjectPublicKeyInfo (SPKI) structure. The resulting SPKI **MUST** be DER-encoded. The SHA-256 hash function **MUST** be applied to the DER-encoded SPKI, per FIPS 180-4. The resulting 32-octet hash output is the PKOC Credential.

4.5 PKOC Derived Identifier Truncation

A PKOC Derived Identifier is derived by truncating a PKOC Credential. Truncation **MUST** retain the **least-significant octets** (the rightmost bytes) of the PKOC Credential and **MUST NOT** alter the order of octets. The resulting length **MUST** be an integer number of octets within the range defined in Section 4.1.

4.6 Validated Credentials and Derived Identifiers

In the Validated model a Credential or Identifier is additionally backed by a signed attestation — a PKOC-CVC (Section 5) — verifiable against a Trust Anchor. A PKOC Validated Credential **MUST** be derived from the attested public key using Section 4.3 or Section 4.4. A PKOC Validated Derived Identifier **MUST** be derived from the corresponding PKOC Validated Credential using the truncation procedure in Section 4.5.

A PKOC Validated Derived Identifier **MAY** have a length of 4 to 7 octets. In that case the issuer of the PKOC-CVC **MUST** ensure, at the time the PKOC-CVC is provisioned, that the identifier does not collide with any other identifier the issuer has assigned under the same IIR; the issuer's signature over the PKOC-CVC attests that uniqueness. Such an identifier is unique only within the scope of that IIR, and a Reader **MUST** treat it as scoped to the IIR of the validated PKOC-CVC.

4.7 Worked Examples

PKOC Credential V1 — from an uncompressed 65-octet EC point, the credential is the 32-octet X coordinate:

```
EC Public Key:    04 BEA02AA1320054CFF1DFD2F88FA583B5B059833BA87CEC415ABDAE0791F0EC66

                     A913C7104A725F6497B8C08FF91217B106FEF7B51ACD4ADF6645E765E4E88D84

PKOC Credential:  BEA02AA1320054CFF1DFD2F88FA583B5B059833BA87CEC415ABDAE0791F0EC66
```

PKOC Derived Identifier (8 octets) — the rightmost 8 octets of the credential above:

```
5ABDAE0791F0EC66
```

The same 8-octet identifier as a DDT (OID + value) object:

```
7F4E16 060A2B0601040183FC2F0C02 53085ABDAE0791F0EC66
```

5. PKOC-CVC Attestation Certificate

5.1 Overview and Purpose

The **PKOC-CVC** is an attestation-certificate format used with PKOC credentials at the door and throughout the card lifecycle. It is based on **EAC-CVC**, the Card Verifiable Certificate standard used worldwide with passports and eID cards (BSI TR-03110-3; ICAO Doc 9303 Part 11), and follows the same ASN.1 structure while defining its own certificate fields optimized for physical access.

As an attestation certificate, the issuer signs a binding between a **subject public key** and the device or card that holds the corresponding private key in non-extractable form. The certificate attests that using the subject key requires possession of that specific device, so a relying party can trust that a signature produced with the subject key originates from that device. This is the concrete realization of the Core's Validated trust model (Section 2.3): a Reader holding the issuer's public key as a Trust Anchor validates the PKOC-CVC without any traditional PKI, revocation lists, or prior enrollment of the individual card key.

Use of the PKOC-CVC is **OPTIONAL**. It applies only where a deployment elects the Validated trust model; Standard-mode PKOC operates without it, and every requirement in this section is conditional on an implementation having elected to issue or validate PKOC-CVCs. The format is open: any organization **MAY** generate PKOC-CVCs using its own Issuer Key and its own Issuer Identification Reference, and any Reader **MAY** validate them, without licence, royalty, registration, or vendor approval.

This section defines the container format — its structure, fields, and encoding — but not the semantics of issuer-specific data. The core binding (subject public key ↔ device, vouched for by the issuer) is fixed; the identifying and descriptive data around it is carried in the Issuer Identification Reference, the Subject Reference, and the Certificate Extension Data, and is interpreted within an issuer's deployment profile. The profile is self-descriptive (it carries its own tag numbers and field lengths), consists of BER-TLV data objects per ISO/IEC 7816-4, uses ASN.1 Application-class tags from the 7816 tag space, uses interindustry data objects per ISO/IEC 7816-6 Table 7, and includes a discretionary data template for PSIA-specific OIDs and values.

5.2 Data Objects and Encoding

Distinguished Encoding Rules (DER) **MUST** be used to encode both the ASN.1 data structures and the application-specific data objects, producing a Tag-Length-Value structure. The Tag is encoded in one or two octets; the Length is an unsigned integer in one, two, or three octets (maximum 65,535, using the minimum number of octets); the Value is zero or more octets. The following ASN.1 universal-class tags are referenced:

Tag	ASN.1 type
0x03	BIT STRING
0x04	OCTET STRING
0x06	OBJECT IDENTIFIER
0x0C	UTF8String
0x30	SEQUENCE (tag number 16, constructed)

The interindustry data objects used in the PKOC-CVC (some renamed to clarify function) are:

Tag	Name	Len	Type / comment
06	Object Identifier	V	OBJECT IDENTIFIER
42	Issuer Identification Reference (IIR)	16	Character String — Key ID of Issuer Key
53	Discretionary Data	V	OCTET STRING — arbitrary data
65	Certificate Extension Data	V	SEQUENCE — nests certificate extensions
73	Discretionary Data Template	V	SEQUENCE — nests arbitrary data objects
5F20	Subject Identification Reference	16	Character String — Key ID of subject key
5F24	Valid To	6	Date — certificate expiration
5F25	Valid From	6	Date — certificate generation
5F29	Certificate Profile Identifier	1	Unsigned integer — profile number
5F37	Signature	V	OCTET STRING — static internal authentication
7F21	CV Certificate	V	SEQUENCE — nests body and signature
7F49	Public Key	V	SEQUENCE — key value and domain parameters

Tag	Name	Len	Type / comment
7F4E	Certificate Body	V	SEQUENCE — nests body data objects

5.3 Certificate Profile

The certificate profile specifies the structure of the PKOC-CVC. The data objects **MUST** appear in the order shown in the following table. Profile Version 1 is as follows (m = mandatory, o = optional):

Field (Profile v1)	Tag	M/O	Comment
CV Certificate	7F21	m	Envelope for body + signature.
└ Certificate Body	7F4E	m	Concatenates all body fields.
└ Certificate Profile Identifier	5F29	m	Value 0 for Profile v1.
└ Issuer Identification Reference	42	m	Names the Issuer Key (Section 5.4).
└ Public Key	7F49	m	Subject public key (Section 5.5).
└ Subject Identification Reference	5F20	m	Reference for the subject key.
└ Certificate Effective Date	5F25	m	Valid-from date.
└ Certificate Expiration Date	5F24	m	Valid-to date.
└ Certificate Extension Data	65	o	Extensions, when present (Section 5.7).
└ Signature	5F37	m	Signature over the Certificate Body (Section 5.8).

The Certificate Profile Identifier (tag `5F29`) is `0` for Profile v1; note that other specifications using the CVC structure may use the same profile-identifier value.

5.4 Issuer Identification Reference

The Issuer Identification Reference (IIR, tag `42`) identifies the public key used to verify the certificate signature. It **MUST** be a 16-character string. Each character **MUST** be an uppercase letter `A`–`Z` or a digit `0`–`9`. An IIR **MUST** use either the standard format or the private format defined below: a **standard format** using a country code and IANA Private Enterprise Number (a globally meaningful namespace), or a **private format** for issuer-defined identifiers (an issuer-scoped namespace).

Standard format	Private format
Chars 1–2: ISO 3166-1 alpha-2 country code of the issuer.	Chars 1–2: **MUST** be `01`.
Chars 3–4: reserved for future use.	Chars 3–4: reserved for future use.
Chars 5–10: IANA Private Enterprise Number, zero-padded to 6.	Chars 5–16: issuer-defined; **MUST NOT** be all zeroes.
Chars 11–16: issuer-defined (typically a sequence number).	Example: `01000ACMECORP001`
Example: `US00044986000001`	

Issuers using the private format **MUST** ensure their IIRs are unique within the deployments where their certificates are used.

5.5 Public Key Encodings

The Public Key field (tag `7F49`) is a sequence consisting of an Object Identifier (tag `06`) identifying the algorithm, followed by one or more algorithm-specific value objects carrying the key material. Implementations **MUST** reject certificates whose public-key OID they do not support. Other algorithms

MAY be used, provided the algorithm is identified by an OID in tag 06, the key material uses context-specific tags in the range 81–86, and a corresponding signature encoding is defined. Three encodings are given as examples.

5.5.1 Elliptic Curve

Tag	Data object	Type	Option
06	Object Identifier (curve name)	OBJECT IDENTIFIER	m
86	Public Point (Q)	EC Point	m

The default and mandatory-to-support curve is secp256r1 (P-256 / prime256v1), OID 1.2.840.10045.3.1.7. brainpoolP256r1 (OID 1.3.36.3.3.2.8.1.1.7, RFC 5639) **MAY** also be supported, as **MAY** other named curves. Only the short form (OID + public point) is used; explicit domain parameters **MUST NOT** be present. The Public Point (tag 86) is the uncompressed point 04 ‖ X ‖ Y (65 bytes for secp256r1 and brainpoolP256r1).

5.5.2 RSA

Tag	Data object	Type	Option
06	Object Identifier	OBJECT IDENTIFIER	m
81	Composite Modulus (n)	Unsigned Integer	m
82	Public Exponent (e)	Unsigned Integer	m

The OID (tag 06) is typically rsaEncryption (1.2.840.113549.1.1.1, RFC 8017 / RFC 3279). The modulus (tag 81) and exponent (tag 82) are unsigned integers whose lengths follow the key size (for example 256 bytes for RSA-2048, 384 bytes for RSA-3072).

5.5.3 ML-DSA

Tag	Data object	Type	Option
06	Object Identifier (parameter set)	OBJECT IDENTIFIER	m
86	Public Key (pk)	Octet String	m

The OID identifies the parameter set (ML-DSA-44, -65, or -87) per NIST CSOR registrations 2.16.840.1.101.3.4.3.17–19; the public-key length is fixed by the parameter set. ML-KEM is out of scope: CVC public keys are used for signature verification only.

5.6 Subject Reference and Validity Dates

The **Subject Reference** (tag 5F20) **MUST** be a fixed 16-character string defined by the issuer. It provides a reference for the card public key — an identification number or other reference.

The Certificate Effective Date (tag 5F25) and Certificate Expiration Date (tag 5F24) **MUST** each be encoded as six digits in **YYMMDD** format using GMT. Each digit **MUST** be encoded as unpacked BCD. The year YY **MUST** be interpreted as 20YY, representing years 2000 through 2099. The effective date is the date from which the certificate may be used; the expiration date is the date after which it expires.

5.7 Certificate Extension Data

Certificate Extension Data (tag 65) **MAY** be present. When present, it **MUST** contain one or more Discretionary Data Template objects (tag 73). Each tag 73 object **MUST** contain exactly one Object Identifier object (tag 06) followed by exactly one Discretionary Data value object (tag 53), in that order:

Field	Tag
Certificate Extension Data	65
└ Discretionary Data Template	73
└ PSIA OID	06
└ Value	53

The Profile v1 PSIA OIDs are:

Name	OID	DER type	Length (bytes)
KeyID	1.3.6.1.4.1.59685.7.1	OCTET STRING	32
UUID	1.3.6.1.4.1.59685.8.1	OCTET STRING	16
4ByteUID	1.3.6.1.4.1.59685.8.2	OCTET STRING	4
7ByteUID	1.3.6.1.4.1.59685.8.3	OCTET STRING	7
10ByteUID	1.3.6.1.4.1.59685.8.5	OCTET STRING	10
BinaryID	1.3.6.1.4.1.59685.8.7	BIT STRING	Variable
CardInfo	1.3.6.1.4.1.59685.8.8	UTF8String	Variable
PKOC	1.3.6.1.4.1.59685.8.9	OCTET STRING	32
CardType	1.3.6.1.4.1.59685.8.16	UTF8String	Variable

> **Note:** These OIDs are the source of the PKOC-CVC credential types a Reader can output in the Validated mode. The specific transport profile identifies which subset a conforming Reader must support.

5.8 Certificate Signature

The Signature (tag `5F37`) authenticates the integrity and origin of the Certificate Body. It is computed over the complete DER-encoded Certificate Body — the entire TLV structure of tag `7F4E`, including its tag and length octets — using the issuer's private key. The signature algorithm corresponds to the algorithm of the issuer's key; the CVC carries no explicit signature-algorithm identifier. A relying party uses the IIR (tag `42`) to locate the issuer's public key from a configured Trust Anchor and thereby determines the algorithm, hash, and encoding needed to verify the signature.

1. **Assemble the Certificate Body.** Concatenate and DER-encode all fields of tag `7F4E` in the order defined in Section 5.3.
2. **Sign.** Apply the issuer's private-key signing operation, as specified by the issuer key's algorithm, over the complete DER-encoded Certificate Body.
3. **Encode the signature value.** Place the raw signing output in tag `5F37` with no additional ASN.1 wrapping (encodings below).
4. **Wrap as TLV.** Encode the value in a TLV with tag `5F37` and append it after the Certificate Body inside the CV Certificate (tag `7F21`).

The raw signature value per algorithm is:

- **ECDSA:** the fixed 64-byte concatenation `r ‖ s`, each a 32-byte big-endian unsigned integer; the DER `Ecdsa-Sig-Value` SEQUENCE **MUST NOT** be used.
- **RSA:** the RSASSA-PKCS1-v1_5 or RSASSA-PSS signature octet string (RFC 8017), length equal to the modulus size.

- **ML-DSA:** the raw FIPS 204 ML-DSA.Sign output, fixed by parameter set — 2,420 bytes (ML-DSA-44), 3,309 bytes (ML-DSA-65), or 4,627 bytes (ML-DSA-87).

Verification is performed over the complete DER-encoded Certificate Body using the issuer's public key, following the algorithm's verification procedure. Hash selection and padding follow the algorithm specification: ECDSA per FIPS 186-5 / ANSI X9.62 / SEC 1; RSA per RFC 8017; ML-DSA per FIPS 204.

5.9 Certificate Example

The following is a complete PKOC-CVC using Profile v1 (Section 5.3) with a secp256r1 subject key and an ECDSA issuer signature. Values are illustrative and are not valid for production use.

The certificate carries one extension — the PKOC OID (Section 5.7) whose 32-octet value is the PKOC Credential derived from the subject public key per Section 4.3, demonstrating how an attested credential value is bound into the certificate.

Field breakdown

Field	Tag	Len	Value
CV Certificate	7F21	82 01 04	Envelope; 260 octets of content.
└ Certificate Body	7F4E	81 BD	189 octets; the signed structure.
└ Certificate Profile Identifier	5F29	01	`00` — Profile v1.
└ Issuer Identification Reference	42	10	`US00044986000001` (standard format, Section 5.4).
└ Public Key	7F49	4D	OID + uncompressed point, below.
└ Object Identifier	06	08	`2A 86 48 CE 3D 03 01 07` — secp256r1.
└ Public Point (Q)	86	41	`04 ‖ X ‖ Y`, 65 octets.
└ Subject Identification Reference	5F20	10	`ACME000000000042`.
└ Certificate Effective Date	5F25	06	`02 06 00 07 00 09` — 2026-07-09, unpacked BCD.
└ Certificate Expiration Date	5F24	06	`03 01 00 07 00 08` — 2031-07-08, unpacked BCD.
└ Certificate Extension Data	65	30	One Discretionary Data Template, below.
└ Discretionary Data Template	73	2E	
└ PSIA OID	06	0A	`2B 06 01 04 01 83 D2 25 08 09` — PKOC, 1.3.6.1.4.1.59685.8.9.
└ Value	53	20	The 32-octet PKOC Credential.
└ Signature	5F37	40	64-octet raw `r ‖ s` over the complete DER-encoded Certificate Body.

Subject public key (uncompressed, 65 octets)

```
04 3DAFF8FF0A0420289ABE240D0DDED80A9400169676C818BE4F5283CFE1B5CECD

  CB3ECA3B72713F1AAF09D252C954AE6FD87F1600ADB23488D425D2BF804CA10F
```

Derived PKOC Credential (Section 4.3 — the X coordinate)

```
3DAFF8FF0A0420289ABE240D0DDED80A9400169676C818BE4F5283CFE1B5CECD
```

Certificate Body (tag 7F4E, 193 octets including tag and length) — the signature input

```
7F4E81BD5F290100421055533030303434393836303030300
30317F494D06082A8648CE3D0301078641043DAFF8FF0A04
20289ABE240D0DDED80A9400169676C818BE4F5283CFE1B5
CECDCB3ECA3B72713F1AAF09D252C954AE6FD87F1600ADB2
3488D425D2BF804CA10F5F201041434D4530303030303030
30303034325F25060206000700095F240603010007000865
30732E060A2B0601040183D225080953203DAFF8FF0A0420
289ABE240D0DDED80A9400169676C818BE4F5283CFE1B5CE
CD
```

Complete PKOC-CVC (tag 7F21, 265 octets)

```
7F218201047F4E81BD5F290100421055533030303434393938
363030303030317F494D06082A8648CE3D0301078641043D
AFF8FF0A0420289ABE240D0DDED80A9400169676C818BE4F
5283CFE1B5CECDCB3ECA3B72713F1AAF09D252C954AE6FD8
7F1600ADB23488D425D2BF804CA10F5F201041434D453030
303030303030303034325F25060206000700095F24060301
000700086530732E060A2B0601040183D225080953203DAF
F8FF0A0420289ABE240D0DDED80A9400169676C818BE4F52
83CFE1B5CECD5F3740722E223AD3D90EF265A4955CD72C5D
BC6005A13BC7B8EFBA013281266AC1E4070D3D34116529FD
F605253AC7719CCE3F9A2DB82353DED23EA6311B0E2F4C9B
24
```

Issuer public key — for verifying the example signature. In deployment a Reader obtains this key through the Trust Anchor identified by the IIR (Section 6.3).

```
04 D5B2FE92A22F031D754720109D16BC259A7F174DC86F3894555B78FBD12E5D99

   7883F0762CEB8D477C48793A5E7C24E12EC55BF3DFF916E656B5136FCA2EF2BD
```

6. Credential Registration and Provisioning

Registration is transport-independent: a Credential generates its key pair, enrolls the public key with the Provisioning/Issuer backend, and receives the trust anchors it needs. Profiles reuse this process unchanged and add only transport-specific delivery details.

6.1 Recommended Application Registration Process

1. **Key generation.** The Credential generates an EC key pair (NIST P-256 baseline) using a CSPRNG. The private key is stored in hardware-backed secure storage where available.
2. **Enrollment request.** The Credential transmits its public key — with any device attestation — to the Provisioning/Issuer backend over a secure (TLS 1.2+) channel.
3. **Validation.** The backend validates the device, the user identity, and any applicable entitlements.
4. **Credential record.** The backend records the public key with the user's access rights in the PACS.
5. **Acknowledgment.** The backend returns success, optionally including a credential identifier or token.
6. **Trust-anchor provisioning.** For each site the credential may access, the backend provisions the applicable trust anchor(s). Profiles specify the concrete anchor (Section 6.3).
7. **Revocation data (optional).** Where a profile defines revocation, the backend may provision initial revocation data over the same channel and refresh it periodically.

6.2 Registration Sequence Diagram

The following sequence illustrates the transport-independent registration and provisioning exchange. Profiles carry the same steps; only the concrete trust-anchor artifact differs.

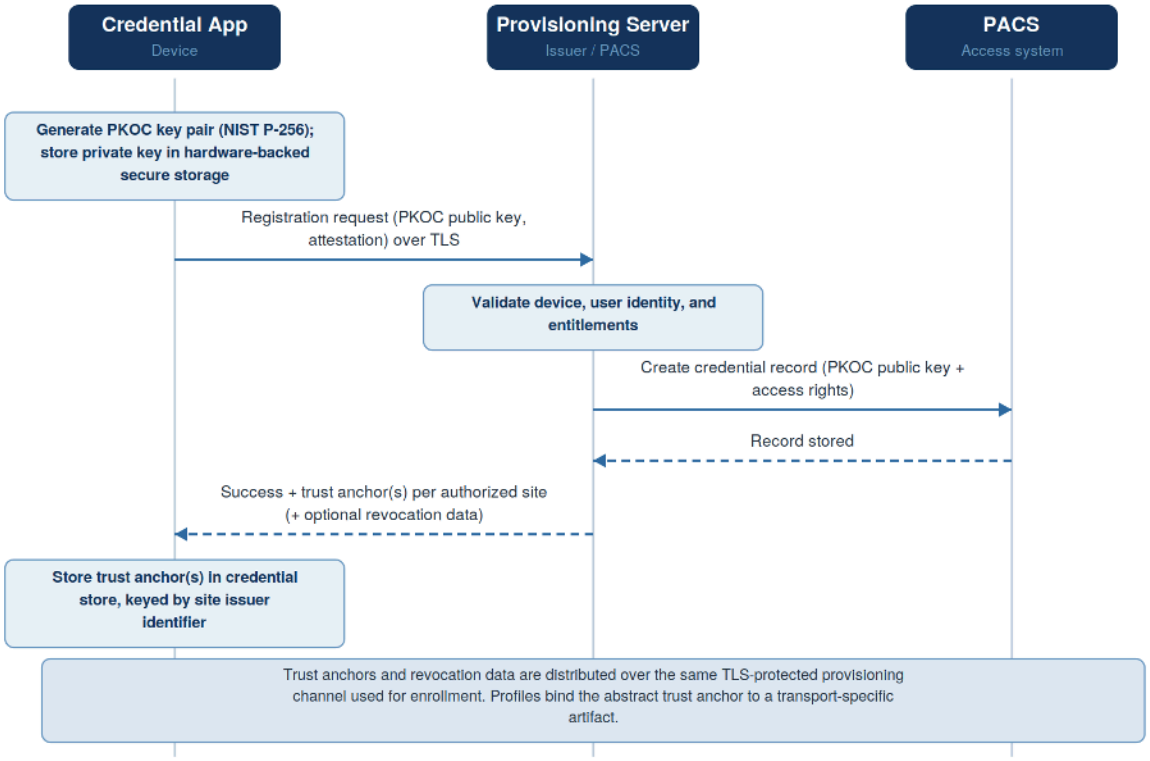

Figure 6-1. Credential registration and trust-anchor provisioning sequence.

6.3 Trust-Anchor Provisioning

A trust anchor is delivered out-of-band over the same TLS-protected provisioning channel used for enrollment, or by QR code or administrator configuration. The Credential stores anchors in its secure credential store, keyed by a profile-defined issuer identifier. The concrete anchor per profile is:

Profile	Concrete trust anchor	What it validates
NFC	Issuer Key (typically published in a JWKS), identified by an Issuer Identification Reference (IIR).	The PKOC-CVC signature that attests the card's public key (Section 5).
BLE	Site Issuer public key, provisioned per site.	The Reader Certificate that authenticates the Reader to the Credential.

7. Common Requirements

These requirements apply to every PKOC implementation. Each Transport Profile restates them in transport-specific terms and adds requirements unique to its link; a conformant implementation **MUST** satisfy both the common requirements below and its profile's requirements.

7.1 Credential (Device / Card) Requirements

1. The Credential **MUST** implement the NIST P-256 curve for its Standard-mode key pair.
2. The Credential **MUST** store the PKOC private key in hardware-backed secure storage where the hardware supports it, and **MUST NOT** transmit the private key over any interface.
3. The Credential **MUST** sign a fresh, Reader-supplied challenge for every transaction, so that captured transactions cannot be replayed.

4. The Credential **MUST** implement at minimum the Standard operating mode of its profile.
5. The Credential **SHOULD** support its profile's Validated operating mode where the deployment requires attestation.

7.2 Reader Requirements

1. The Reader **MUST** implement the NIST P-256 curve for all baseline asymmetric operations.
2. The Reader **MUST** verify the Credential's challenge signature against the presented public key before granting access.
3. The Reader **MUST** generate a fresh, unpredictable challenge (CSPRNG) for every transaction.
4. The Reader **MUST** implement at minimum the Standard operating mode of its profile.
5. A Reader that implements the Validated model **MUST** verify the attestation against a configured Trust Anchor (a PKOC-CVC Issuer Key or a Reader Certificate signer) before honoring the credential.

7.3 Cryptographic Requirements

1. All baseline ECDSA signatures **MUST** use ECDSA-SHA256 on NIST P-256 and **MUST** be transmitted as raw R ‖ S (Section 3.2).
2. Uncompressed public keys **MUST** use `0x04` ‖ X ‖ Y (65 bytes); compressed public keys **MUST** use `0x02/0x03` ‖ X (33 bytes).
3. All random values and ephemeral keys **MUST** be generated with a CSPRNG.

8. Conformance and Profiles

8.1 Profile Conformance

A PKOC implementation always conforms to **this Core plus exactly one or more Transport Profiles**. Conformance to a profile requires satisfying every **MUST**/**MUST NOT** in this Core and every **MUST**/**MUST NOT** in that profile. The Core alone is not directly implementable: it has no wire format. A profile alone is not complete: it inherits the definitions, cryptography, credential derivation, attestation format, and common requirements defined here.

8.2 Adding a New Transport Profile

The Core is deliberately transport-agnostic so that new links can be added without disturbing existing deployments. A new Transport Profile (for example, a future DUOX profile) **MUST**:

1. **Reuse the Core unchanged.** It **MUST** adopt the roles (Section 2), cryptographic foundations (Section 3), credential and derived-identifier model (Section 4), attestation-certificate format (Section 5), and registration process (Section 6) without redefining them.
2. **Realize the Standard trust model.** It **MUST** define a Standard operating mode built on the P-256 baseline and the common Credential/Reader requirements (Section 7).
3. **Bind, not reinvent, the Validated model.** If it offers attestation, it **MUST** map the abstract Trust Anchor (Section 2.3) onto a concrete transport artifact — reusing the PKOC-CVC (Section 5) where the credential itself is attested by a PKOC issuer — rather than defining a new trust concept. A profile **MUST NOT** define a trust artifact whose issuance, signing hierarchy, or key distribution is controlled by a single supplier, and **MUST NOT** make conformance to the profile dependent on such an artifact.
4. **Define only transport-specific content:** the wire/framing format, the command or message set, framing/fragmentation, error signaling, and any transport-specific requirements and sequence diagrams.
5. **Follow the common document structure** (Section 8.3) and the shared formatting so that the profile is visually and structurally consistent with the Core, BLE, and NFC documents.

6. **State its heritage and backward-compatibility** in its revision history and a “Changes in this version” section, mirroring the existing profiles.
7. **Preserve existing implementations.** It **MUST NOT** alter the wire protocol, credential derivation, or conformance obligations of any other profile, and **MUST NOT** make a previously conformant implementation non-conformant. New capabilities **MUST** be added as optional elections.

> **Note:** Because the Core carries all shared concepts — including credential derivation and the attestation-certificate format — a new profile such as DUOX is scoped to its transport binding alone. Implementers of an existing profile are unaffected by the addition of a new one: no Core change is required to introduce a new transport.

8.3 Common Document Structure

Every PKOC document — this Core and each profile — uses the same front-matter order and section conventions so the set reads as one specification:

1. Cover page (title, version, classification, date, maintainer, copyright).
2. Document Revision History.
3. Table of Contents.
4. Glossary (immediately after the Table of Contents; profiles list only transport-specific terms and defer to the Core glossary).
5. Conformance (RFC 2119 keywords).
6. Numbered body sections, with the high-level process overview presented before detailed requirements and command definitions.
7. Appendices and references.

9. References

The following references apply across all PKOC documents. Profiles add references specific to their transport.

- **RFC 2119** — Key words for use in RFCs to Indicate Requirement Levels.
- **FIPS 186-5** — Digital Signature Standard (DSS).
- **FIPS 180-4** — Secure Hash Standard (SHS / SHA-256).
- **FIPS 204** — Module-Lattice-Based Digital Signature Standard (ML-DSA).
- **SEC 1** — Elliptic Curve Cryptography, Version 2.0.
- **ANSI X9.62** — Public Key Cryptography for the Financial Services Industry: Elliptic Curve Cryptography.
- **ITU-T X.690** — ASN.1 encoding rules (BER, CER, DER).
- **RFC 4648** — The Base16, Base32, and Base64 Data Encodings.
- **RFC 8017** — PKCS #1: RSA Cryptography Specifications Version 2.2.
- **RFC 3279** — Algorithms and Identifiers for the Internet X.509 PKI (RSA/ECDSA OIDs).
- **RFC 5639** — Elliptic Curve Cryptography (ECC) Brainpool Standard Curves.
- **ISO/IEC 7816-4:2020** — Organization, security, and commands for interchange (BER-TLV).
- **ISO/IEC 7816-6:2023** — Interindustry data elements for interchange.
- **ISO/IEC 7816-9:2017** — Commands for card management.
- **BSI TR-03110-3** — Advanced Security Mechanisms for MRTDs (EAC-CVC).
- **ICAO Doc 9303 Part 11** — Machine Readable Travel Documents: security mechanisms.
- **NIST CSOR** — Computer Security Objects Register (algorithm registrations).
- **PKOC BLE Transport Profile**, Version 2.0.1 — PSIA.

- **PKOC NFC Transport Profile**, Version 2.0.1 — PSIA.

End of PKOC Core Specification, Version 2.0.1.