primesign

digital signing, simple as that.

PrimeSign RKSV v2 Signature Cards

Author: PrimeSign GmbH
office@prime-sign.com

Document Version: 1.0.0
Date of Issue: 2026-06-01

PUBLIC

PrimeSign GmbH
Wielandgasse 2 . 8010 Graz . Austria
T +43 (316) 25 830-0 . **E** office@prime-sign.com

cryptas.com . prime-sign.com . cryptoshop.com

Vienna | Graz | Düsseldorf | Hengelo | Stockholm

TABLE OF CONTENTS

About this document

Thank you very much for your confidence in PrimeSign RKSV signature cards. With this document, we guide you through the first steps to integrate our signature card in your products.

You receive the signature card already fully activated. The APDU sequences for reading the RKSV signing certificate from the smart card, issuing a signature, and changing the signature PIN are described in the following sections.

Document History

Date	Version	Author	Comment
2026-05-22	0.9.9	Simon Roth	Initial document
2026-06-01	1.0.0	Thomas Knall	Review and final adjustments

1. Usage

By default, the signature application is pre-selected upon card reset; therefore, the following sections assume it is already selected.

If you need to select other components on the card (for example, to perform the optional unique identification process shown in Section 1.3.2), you must navigate back to the SSCD Application before initiating any subsequent processes, such as triggering a signature or reading a certificate. Example APDUs for this re-selection process can be found in Section 1.3.2.

1.1. Applet Structure

The applet's file system is structured as follows:

- SSCD Application (ChipDoc Signature Application) (AID: `E828BD080F01`)
  - Master File / MF (ID: `3F00`)
    - SSCD Application DF (ID: `0001`)
      - Digital Signature DF (ID: `0002`)
        - Certificate EF (ID: `0004`)

1.2. Signature Creation

The application expects the *Data To Be Signed* to be pre-hashed.

1.2.1. APDU sequences for Signature Creation

Assuming the card has just been reset and no other directories have been selected (which leaves the signature application default-selected), the following APDU sequence is sufficient to authenticate the user (using the Signature PIN) and to create a digital signature:

Step 1: Select SSCD Application Dedicated File (ADF)
Command APDU: `00 A4 00 0C 02 00 01`

Step 2: Select Digital Signature Dedicated File (DF)
Command APDU: `00 A4 00 0C 02 00 02`

Step 3: Verify Signature PIN (RAD)
Command APDU: `00 20 00 02 06` **`313131313131`**

Note: `00 20` *is the VERIFY command. The payload* **`313131313131`** *represents the sample Signature PIN value* `"111111"` *encoded in ASCII Hex. This value must be adapted to the actual Signature PIN of your card.*

Step 4: Perform Digital Signature
Command APDU:
`00 2A 9E 9A 20` **`2D2DA1...EF30`** `40`

Note: `00 2A` *is the PERFORM SECURITY OPERATION (SIGN) command.*

The payload is structured as follows:

- `20` (Lc): *Specifies that exactly 32 bytes of payload data will follow.*
- `2D2DA1...EF30` (Data): *The exact 32 bytes of payload data.* ***Please note that this is just an example hash value.*** *The card processes these 32 bytes as an already finalized digest (e.g., SHA-256) of the Data to be Signed instead of raw text. This significantly saves processing time and communication overhead.*
  Section 1.2.3 provides an example for digest calculation with Java.
- `40` (Le): *Specifies the expected response length of 64 bytes. For 256-bit Elliptic Curve Cryptography (ECC/ECDSA) keys, the returned signature (consisting of the concatenated values r and s in plain format) is exactly twice the length of the key size (2 x 32 bytes = 64 bytes).*

1.2.2. Error Handling for PIN Verification

When executing the Verify Signature PIN (Step 3 above) command, it is crucial to properly handle the Status Word returned by the smart card. The card enforces a strict retry policy to protect the signature key against brute-force attacks.

The terminal software should parse the response and handle the following Status Word values appropriately:

- `9000` **(Success)**
  The PIN was verified successfully, and the security status is granted.
- `63Cx` **(Verification Failed - Retries Remaining)**
  The provided PIN was incorrect. The last nibble (`x`) indicates the exact number of verification attempts remaining. For example, `63C2` means 2 retries are left. If the card returns `63C0`, the PIN was incorrect, the retry counter has reached zero, and the PIN is now **blocked**.
- `6983` **(Authentication Key Locked)**
  The verification failed because the PIN is already **blocked**.
- `6984` **(Data Invalidated)**
  The referenced data is invalidated (this may also indicate a blocked or corrupted state).
- `6700` **(Wrong Length)**
  The length of the provided PIN payload does not match the expected format or length constraints.
- `6A88` **(Data Object Not Found)**
  The specified key reference (PIN ID) does not exist on the card.

1.2.3. Example: Hash Calculation in Java

Before performing the Digital Signature (Step 4 above), the SHA-256 hash of the *Data To Be Signed* must be computed.

The following code snippet shows calculating the SHA-256 using Java:

```
byte[] dataToBeSigned = ...;
byte[] hashValue = null;
try {
    MessageDigest md = MessageDigest.getInstance("SHA-256");
    hashValue = md.digest(dataToBeSigned);
} catch (Exception cause) {
    throw new RuntimeException(cause);
}
// Build APDU using hashValue
CommandAPDU signCmd = new CommandAPDU(0x00, 0x2A, 0x9E, 0x9A, hashValue, 64);
ResponseAPDU signResp = card.getBasicChannel().transmit(signCmd);  // card
instanceof javax.smartcardio.Card
if (signResp.getSW() != 0x9000) {
    throw new RuntimeException("Signature creation failed.");
}
byte[] signatureValue = signResp.getData();
```

1.3. PrimeSign RKSV v2 Card Identification

If the cash register (POS system) needs to operate with smart cards other than the PrimeSign RKSV v2 cards, the following methods can be used to identify our cards:

1.3.1. Identification via ATR (Answer To Reset)

Our current cards feature the following ATR:

`3BD518FF81B1FE451FC38073C821106F`

This ATR identifies the underlying hardware and operating system. It is not guaranteed to be strictly unique to PrimeSign and might be used by other card issuers utilizing the same chip platform.

1.3.2. Unique Identification via Card Manager

To reliably distinguish PrimeSign cards from other cards sharing the same ATR, a unique identifier can be read directly from the GlobalPlatform Card Manager.

- **Step 1: Select Card Manager (Issuer Security Domain)**
  Command APDU: `00 A4 04 00 08 A0 00 00 01 51 00 00 00`

(If the card responds with an error instead of `9000` (OK), *it is not a GlobalPlatform card and thus not a PrimeSign card).*

- **Step 2: Get CIN (Card Image Number, Tag `45`)**
  Command APDU: `80 CA 00 45 00`
- **Step 3: Verify the Response**
  Convert the returned response byte array into a Hex String and compare it with the exact reference value:

  `450750532D524B2D32`

  Technical detail: This response represents a TLV (Tag-Length-Value) structure where Tag `45` *with Length* `07` *contains the Hex-encoded ASCII string:* `PS-RK-2`

State Change after Identification:
Explicitly selecting the Card Manager overrides the "default selected" state of the ChipDoc signature application. If you intend to perform signature operations right after this identification process (at least without resetting the card), you must manually re-select the ChipDoc application and its Master File (MF) before starting the regular signature sequence:

- **Step 4: Re-select ChipDoc Application**
  Command APDU: `00 A4 04 0C 06 E8 28 BD 08 0F 01`
- **Step 5: Select Master File (MF)**
  Command APDU: `00 A4 00 0C 02 3F 00`

1.4. Reading the Certificate from the Card

It might not always be necessary to read the signing certificate from the smart card. However, especially during the integration it may be helpful.

To extract the public certificate from the smart card, the exact file path must be selected first. The certificate is stored in a dedicated Elementary File (EF) within the Digital Signature environment.

Assuming the card has just been reset or is in an unknown directory state, the following APDU sequence selects the necessary directories and reads the certificate in one go:

Step 1: Select SSCD Application Dedicated File (ADF)
Command APDU: `00 A4 00 0C 02 00 01`

Step 2: Select Digital Signature Dedicated File (DF)
Command APDU: `00 A4 00 0C 02 00 02`

Step 3: Select Certificate Elementary File (EF)
Command APDU: `00 A4 00 0C 02 00 04`

Step 4: Read Binary (Entire File)
Command APDU: `00 B0 00 00 00 00 00`

> **Extended Length APDU:**
> The `READ BINARY` command above uses an Extended Length Expected field (`Le = 00 00 00`). This allows the terminal to read the entire certificate in a single command, avoiding the overhead of multiple read cycles. The maximum expected file size for the certificate is 1500 bytes.

1.4.1. Example: Parsing the Certificate in Java

Once the raw certificate bytes are successfully retrieved from the card, they can be parsed into a standard `X.509` certificate object using the Java `CertificateFactory`, for example.

```
byte[] certBytes = response.getValue();
try (ByteArrayInputStream is = new ByteArrayInputStream(certBytes)) {
    CertificateFactory fact = CertificateFactory.getInstance("X.509");
    Certificate cert = fact.generateCertificate(is);
} catch (Exception cause) {
    // Process exception
}
```

1.5. Changing the Signature PIN

To change the signature PIN, the current PIN must be verified first to satisfy the required access condition. The new signature PIN **must have a length of 6 digits**.

Step 1: Select SSCD Application Dedicated File (ADF)
Command APDU: `00 A4 00 0C 02 00 01`

Step 2: Select Digital Signature Dedicated File (DF)
Command APDU: `00 A4 00 0C 02 00 02`

Step 3: Verify Current Signature PIN
Command APDU: `00 20 00 02 06 313131313131`
Note: The payload is the current PIN encoded in ASCII Hex.
The sample payload `313131313131` *reflects the PIN:* `"111111"`

Step 4: Change PIN
Command APDU: `00 24 01 82 06 323232323232`
Note: The payload is the new PIN encoded in ASCII Hex.
The sample payload `323232323232` *reflects the PIN:* `"222222"`

Note that the differing P2 value compared to VERIFY (`82` vs. `02`) is intentional. CHANGE PIN encodes the PIN reference as `0x80|id`, so reference `02` becomes `82`. Both commands address the same Signature PIN.