EMV® Specification Bulletin No. 243
First Edition October, 2021

Introduction of XDA/ODE for EMV Specifications

This Specification Bulletin explains the additional requirements to support XDA/ODE for current EMV Contact Specifications.

Applicability

This Specification Bulletin applies to:

- *EMV Integrated Circuit Card Specifications for Payment Systems, Book 2 – Security and Key Management, Version 4.3, November 2011*
- *EMV Integrated Circuit Card Specifications for Payment Systems, Book 3 – Application Specification, Version 4.3, November 2011*
- *EMV Integrated Circuit Card Specifications for Payment Systems, Book 4 – Cardholder, Attendant, and Acquirer Interface Requirements, Version 4.3, November 2011*

Related Documents

- *None*

Effective Date

- *January, 2023*

Description

This bulletin describes the requirements to support XDA (Extended Data Authentication) and ODE (Offline Data Encipherment) using ECC (Elliptic Curve Cryptography) for contact EMV transactions. Editorial updates are also included in this bulletin.

In some sections, the whole section, paragraph, or table is listed in this Bulletin for readability. Unless otherwise noted, underlined text indicates revisions and strikethroughs indicate deleted text.

<Changes to Book 2>

In section 1.2, the following description is added:

Part II covers:

- Offline static data authentication (SDA)
- Offline dynamic data authentication (DDA and CDA and XDA)

In section 2, the following references are added:

FIPS 202	SHA-3 Standard: Permutation-Based Hash and Extendable-Output Functions
IEEE P1363	Standard Specifications For Public-Key Cryptography
ISO/IEC 11770-6	Information technology – Security techniques – Key management — Part 6: Key derivation
ISO/IEC 14888-3	Information technology – Security techniques – Digital signatures with appendix — Part 3: Discrete logarithm based mechanisms
ISO/IEC 15946-1	Information technology – Security techniques – Cryptographic techniques based on elliptic curves — Part 1: General
ISO/IEC 15946-5	Information technology – Security techniques – Cryptographic techniques based on elliptic curves — Part 5: Elliptic curve generation
ISO/IEC 19772	Information technology – Security techniques – Authenticated encryption
SEC 1	Elliptic Curve Cryptography (available at http://www.secg.org)

In section 3, Definitions, delete the following terms:

~~**Accelerated Revocation**~~	~~A key revocation performed on a date sooner than the published key expiry date.~~
~~**Key Expiry Date**~~	~~The date after which a signature made with a particular key is no longer valid. Issuer certificates signed by the key must expire on or before this date. Keys may be removed from terminals after this date has passed.~~

~~**Key Life Cycle**~~	~~All phases of key management, from planning and generation, through revocation, destruction, and archiving.~~
~~**Key Replacement**~~	~~The simultaneous revocation of a key and introduction of a key to replace the revoked one.~~
~~**Key Revocation**~~	~~The key management process of withdrawing a key from service and dealing with the legacy of its use. Key revocation can be as scheduled or accelerated.~~
~~**Key Revocation Date**~~	~~The date after which no legitimate cards still in use should contain certificates signed by this key, and therefore the date after which this key can be deleted from terminals. For a planned revocation the Key Revocation Date is the same as the key expiry date.~~
~~**Planned Revocation**~~	~~A key revocation performed as scheduled by the published key expiry date.~~

In section 4.1, the following Abbreviations are added in alphabetical order:

AAD	Additional Authenticated Data
AES	Advanced Encryption Standard
EC-SDSA	Elliptic Curve Schnorr Digital Signature Algorithm
ECC	Elliptic Curve Cryptography
ICCD	Issuer Certified Card Data
KD	Key Derivation
N_{FIELD}	Length of a finite field element
N_{HASH}	Output length of a hash function
N_{SIG}	Length of an ECC Digital Signature
ODA	Offline Data Authentication
ODE	Offline Data Encipherment
SHA-2	Secure Hash Algorithm 2 (includes SHA-256 and SHA-512)
SHA-3	Secure Hash Algorithm 3
UN	Unpredictable Number
XDA	Extended Data Authentication

Section 5 is updated as below:

This section describes SDA, an offline data authentication mechanism that utilises RSA public key cryptography.

Offline static data authentication is performed by the terminal using a digital signature scheme based on public key techniques to confirm the legitimacy of critical ICC-resident static data. This detects unauthorised alteration of data after personalisation.

The only form of offline static data authentication defined is Static Data Authentication (SDA) that verifies the data identified by the Application File Locator (AFL) and by the optional Static Data Authentication Tag List.

SDA requires the existence of a certification authority, which is a highly secure cryptographic facility that 'signs' the issuer's public keys.

Every terminal conforming to this specification shall contain the appropriate certification authority's public key(s) for every application recognised by the terminal.

This specification permits multiple AIDs to share the same 'set' of Certification Authority Public Keys. The relationship between the data and the cryptographic keys is shown in Figure 1.

Section 5 Table 1 and its description are updated as below:

Required Data Element	Length	Description
Certification Authority Public Key Index	1	Contains a binary number that indicates which of the application's Certification Authority Public Keys and its associated algorithm that reside in the terminal is to be used with this ICC.
Issuer Public Key Certificate	var.	Provided by the appropriate certification authority to the card issuer. When the terminal verifies this data element, it authenticates the Issuer Public Key plus additional data as described in section 5.3.
Signed Static Application Data	var.	Generated by the issuer using the private key that corresponds to the public key authenticated in the Issuer Public Key Certificate. It is a digital signature covering critical ICC-resident static data elements, as described in section 5.4.
Issuer Public Key Remainder	var.	The presence of this data element in the ICC is conditional. See section 5.1 for further explanation.
Issuer Public Key Exponent	var.	Provided by the issuer. See section 5.1 for further explanation.

Table 1: Required ICC Data Elements for SDA

To support SDA, each terminal shall be able to store six Certification Authority Public Keys per Registered Application Provider Identifier (RID) and shall associate with each such key the key-related information to be used with the key (so that terminals can in the future support multiple algorithms and allow an evolutionary transition from one to another, as discussed in section 11.2.2). The terminal shall be able to locate any such key (and the key-related information) given the RID and Certification Authority Public Key Index as provided by the ICC.

Section 6 is updated as below:

6 Offline Dynamic Data Authentication (DDA, CDA)

This section describes DDA and CDA, two offline dynamic data authentication mechanisms that utilise RSA public key cryptography.

Offline dynamic data authentication is performed by the terminal using a digital signature scheme based on public key techniques to authenticate the ICC and confirm the legitimacy of critical ICC-resident/generated data and data received from the terminal. This precludes the counterfeiting of any such card.

Two forms of RSA offline dynamic data authentication exist:

- Dynamic Data Authentication (DDA) executed before card action analysis, where the ICC generates a digital signature on ICC-resident/generated data identified by the ICC Dynamic Data and data received from the terminal identified by the Dynamic Data Authentication Data Object List (DDOL).
- Combined Dynamic Data Authentication/Application Cryptogram Generation (CDA) executed at issuance of the first and second GENERATE AC commands. In the case of a Transaction Certificate (TC) or Authorisation Request Cryptogram (ARQC), the ICC generates a digital signature on ICC-resident/generated data identified by the ICC Dynamic Data, which contains the TC or ARQC, and an Unpredictable Number generated by the terminal[10].

The AIP denotes the options supported by the ICC.

Offline dynamic data authentication requires the existence of a certification authority, a highly secure cryptographic facility that 'signs' the Issuer's Public Keys. Every terminal conforming to this specification shall contain the appropriate certification authority's public key(s) for every application recognised by the terminal. This specification permits multiple AIDs to share the same 'set' of Certification Authority Public Keys. The relationship between the data and the cryptographic keys is shown in Figure 2.

Section 6 Table 8 and its description are updated as below:

ICCs that support DDA or CDA shall contain the data elements listed in Table 8:

Required Data Element	Length	Description
Certification Authority Public Key Index	1	Contains a binary number that indicates which of the application's Certification Authority Public Keys and its associated algorithm that reside in the terminal is to be used with this ICC.
Issuer Public Key Certificate	var.	Provided by the appropriate certification authority to the card issuer. When the terminal verifies this data element, it authenticates the Issuer Public Key plus additional data as described in section 6.3.
ICC Public Key Certificate	var.	Provided by the issuer to the ICC. When the terminal verifies this data element, it authenticates the ICC Public Key plus additional data as described in section 6.4.
Issuer Public Key Remainder	var.	See section 6.4 for further explanation.
Issuer Public Key Exponent	var.	Provided by the issuer. See section 6.4 for further explanation.
ICC Public Key Remainder	var.	See section 6.4 for further explanation.
ICC Public Key Exponent	var.	Provided by the issuer. See section 6.4 for further explanation.
ICC Private Key	var.	ICC internal. Used to generate the Signed Dynamic Application Data as described in sections 6.5 and 6.6.

Table 8: Required ICC Data Elements for offline dynamic data authentication

Section 6 Table 9 and its description are updated as below:

ICCs that support DDA or CDA shall generate the data element listed in Table 9:

Data Element	Length	Description
Signed Dynamic Application Data	var.	Generated by the ICC using the private key that corresponds to the public key authenticated in the ICC Public Key Certificate. This data element is a digital signature covering critical ICC-resident/generated and terminal data elements, as described in sections 6.5 and 6.6.

Table 9: Data Element Generated for offline dynamic data authentication

To support offline dynamic data authentication, each terminal shall be able to store six Certification Authority Public Keys per RID and shall associate with each such key the key-related information to be used with the key (so that terminals can in the future support multiple algorithms and allow an evolutionary transition from one to another, see section 11.2.2). The terminal shall be able to locate any such key (and key-related information) given the RID and Certification Authority Public Key Index as provided by the ICC.

DDA and CDA shall use a reversible algorithm as specified in Annex A2.1 and Annex B2. Section 11.2 contains an overview of the keys and certificates involved in the offline dynamic data authentication process. Sections 6.2 to 6.4 specify the initial steps in the process, namely:

- Retrieval of the Certification Authority Public Key by the terminal.
- Retrieval of the Issuer Public Key by the terminal.
- Retrieval of the ICC Public Key by the terminal.

If DDA or CDA fails then the TVR bit indicating failure of the attempted method shall be set as follows:

- If the attempted method is DDA then the terminal shall set the 'DDA failed' bit in the TVR to 1.
- If the attempted method is CDA then the terminal shall set the 'CDA failed' bit in the TVR to 1.

Sections 6.5 and 6.6 specify the dynamic signature generation and verification processes for each method.

Section 6.6.1, Description number 1 is updated as below:

1. The terminal issues a first or second GENERATE AC command with the 'CDA signature requested' bits in the GENERATE AC command set to '10' according to sections 6.5.5.4 and 9.3 of Book 3.

In Section 7, the first two paragraphs are updated as below:

7 Personal Identification Number and Biometric Data Encipherment using RSA

This section relates only to PIN and Biometric Data Encipherment using RSA.

If supported, Personal Identification Number (PIN) encipherment for offline PIN verification is performed by the terminal using an RSA based encipherment mechanism in order to ensure the secure transfer of a PIN from a ~~secure tamper-evident PIN pad~~ CVM capture device to the ICC.

The entirety of Section 10 is replaced with the following two paragraphs:

The contents of this section have been deleted.

With the maximum size of RSA keys already in use, there is no longer a need for a structured certification authority process to replace shorter keys, as was required for earlier versions of this specification. Some minor residual content has been incorporated into section 11.

Section 11.2 is updated as below:

This section specifies the requirements for the management by acquirers of the Certification Authority Public Keys in the terminals. The requirements cover the following phases:

- Introduction of a Certification Authority Public Key in a terminal
- Storage of a Certification Authority Public Key in a terminal
- Usage of a Certification Authority Public Key in a terminal
- Withdrawal of a Certification Authority Public Key from a terminal

Terminal management functions shall allow for the installation and withdrawal of Certification Authority Public Keys by acquirers or their agents. The terminal management processes need to ensure the authenticity of the keys and protect the integrity of the installed keys. Processes also need to confirm the authenticity of the instructions to introduce or withdraw keys and to confirm that the action has been completed.

Section 11.2.1 is updated as below:

~~When a payment system has decided that~~For the introduction of a new Certification Authority Public Key ~~is to be introduced~~, ~~a process is executed that ensures the distribution of the new key from the~~ payment system~~s make the keys available~~ to ~~each acquirer~~ Acquirers. It is ~~then~~ the acquirer's responsibility to ensure that the new Certification Authority Public Key and its related data (see section 11.2.2) is ~~conveyed to~~ installed in its terminals. The Certification Authority Public Key will also be made available to Issuers to allow the verification of Issuer Public Key Certificates requested by an Issuer.

~~The following principles apply to the introduction of a Certification Authority Public Key from an acquirer to its terminals:~~

- ~~The terminal shall be able to verify that it received the Certification Authority Public Key and its related data error-free from the acquirer.~~
- ~~The terminal shall be able to verify that the received Certification Authority Public Key and related data originated from its legitimate acquirer.~~
- ~~The acquirer shall be able to confirm that the new Certification Authority Public Key was introduced correctly in its terminals.~~

One process for the introduction of Certification Authority Public Keys into terminals is to install the keys during terminal manufacture and deployment. The keys may be made available by the payment systems to terminal vendors, or acquirers may specify the keys necessary for their particular market segments.

On being informed of a new key being available, Acquirers are expected to install it in their terminal base within six months.

During installation into the terminal the integrity of the Certification Authority Public Key and its related data is confirmed, see section 10.2 of Book 4.

For RSA, the integrity of the Certification Authority Public Key may be protected using a Check Sum calculated in the same manner as is used to protect the integrity in storage as described in section 11.2.2.

For ECC, the Certification Authority Public Key will be self-signed for distribution according to Table 26a. A diagram showing the construction of a self-signed Certification Authority Public Key Certificate is shown in Figure 13a.

Successful verification of the self-signed Certification Authority Certification Authority Public Key Certificate also ensures that the terminal supports the algorithm suite associated with the Certification Authority Public Key and that will be used for verifying Issuer Public Key Certificates.

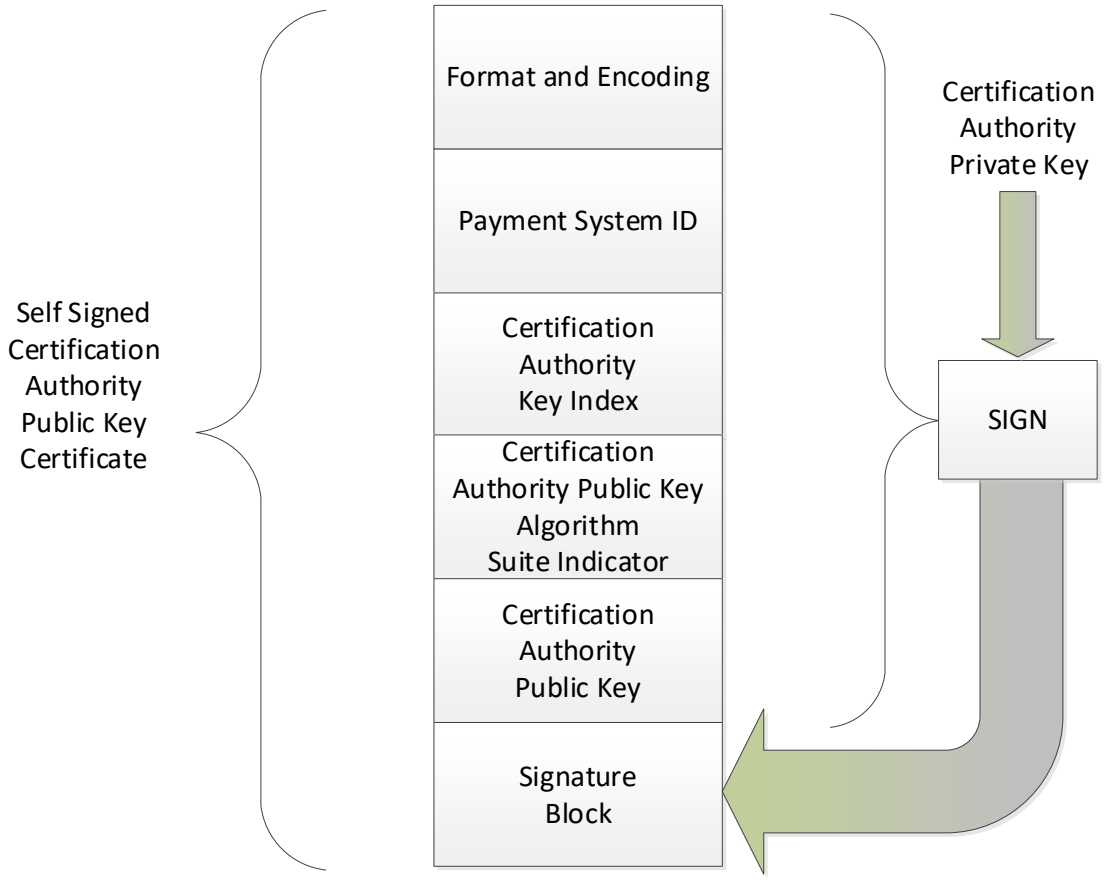


Figure 13a: ECC Self-signed Certification Authority Public Key Diagram

As shown in Figure 13a, the self-signed Certification Authority Public Key Certificate consists of a plain text prefix followed by a signature block. The certificate is specified in the following table where for this version of the specifications the encoding of the certificate is binary (i.e. value, value, value... with no additional identifier or length fields).

	Name	Description	Length (bytes)
1	Certificate Format	Always Hex value '20'.	1
2	Certificate Encoding	Always Hex value '00' for this version of the specifications.	1
3	RID	Identifies the Payment System to which the Certification Authority public key is associated. The RID is identified in the AID (first 5 bytes).	5
4	Certification Authority Public Key Index	When combined with the RID, uniquely identifies the Certification Authority Public Key.	1
5	Certification Authority Public Key Algorithm Suite Indicator	Indicates the algorithms to be used with the Certification Authority Public Key.	1
6	Certification Authority Public Key	Representation of Certification Authority Public Key (x-coordinate of Certification Authority public key point) on the curve identified by the Certification Authority Public Key Algorithm Suite Indicator.	NFIELD
7	Certification Authority Public Key Certificate Signature	Output of digital signature ECC algorithm on concatenated first six data objects using the Certification Authority private key on the elliptic curve identified by the Certification Authority Public Key Algorithm Suite Indicator.	NSIG

Table 26a: ECC Self-Signed Certification Authority Public Key & Related Data (Binary Encoding)

At the end of the installation, after verifying the integrity of the public key and its related data, the terminal shall apply storage integrity protection of the public key and its related data to allow the subsequent re-verification of their integrity. This may be achieved using the Certification Authority Public Key Check Sum in Tables 27 and 27a or a proprietary mechanism.

Section 11.2.2 is updated as below:

Terminals that support RSA-based offline ~~static or dynamic~~ data authentication, offline PIN encipherment, or biometric data encipherment shall provide support for at least six RSA Certification Authority Public Keys per RID for ~~EMVCo member~~ EMV-based payment systems' debit/credit applications based on this specification.

Terminals that support XDA or ECC ODE shall provide support for the ECC domain parameters for the two ECC curves specified in Annex B2.2.2 and at least ten ECC Certification Authority Public Keys for either the same or different curves (and related key information) per RID for EMV-based payment systems' payment applications based on this specification.

Each Certification Authority Public Key is uniquely identified by the 5-byte RID that identifies the payment system in question, and the 1-byte Certification Authority Public Key Index, unique per RID and assigned by that payment system to a particular Certification Authority Public Key.

For each Certification Authority Public Key, the minimum set of data elements that shall be available in the terminal is specified in Table 27 (for RSA) or Table 27a (for ECC).

~~The RID and the Certification Public Key Index together uniquely identify the Certification Authority Public Key and associate it with the proper payment system.~~

For RSA, the Certification Authority Public Key Algorithm Indicator identifies the digital signature algorithm to be used with the corresponding Certification Authority Public Key. The only acceptable value at this moment is hexadecimal '01', indicating the usage of the RSA algorithm in the digital signature scheme as specified in Annex A2.1 and Annex B2.1. The Hash Algorithm Indicator specifies the hashing algorithm to produce the Hash Result in the digital signature scheme. The only acceptable value at this moment is hexadecimal '01', indicating the usage of the SHA-1 algorithm.

For ECC, the Certification Authority Public Key Algorithm Suite Indicator identifies the Algorithm Suite to be used with the corresponding Certification Authority Public Key. The only acceptable values at this moment are those specified in Annex B2.4.1. Like the Certification Authority Public Keys and related data, the integrity of the ECC domain parameters associated with each suite shall be protected by the terminal (as shall other sensitive terminal data and code).

The Certification Authority Public Key Index is unique and not duplicated within a RID irrespective of the algorithm associated with the Certification Authority Public Key. An RSA key and an ECC key for the same RID cannot have the same Index.

~~The Certification Authority Public Key Check Sum is derived using the technique specified in section 10.2 of Book 4, to ensure that a Certification Authority Public Key and its related data are received error-free. The terminal may use this data element to subsequently re-verify the integrity of a Certification Authority Public Key and its related data. Alternately, the terminal may use another technique to ensure the integrity of this data.~~

The integrity of the stored Certification Authority Public Keys and their related data should be verified periodically. This may be achieved using the Certification Authority Public Key Check Sum in Table 27 and Table 27a or a proprietary mechanism.

Field Name	Length	Description	Format
Registered Application Provider Identifier (RID)	5	Identifies the payment system to which the Certification Authority Public Key is associated	b

Field Name	Length	Description	Format
Certification Authority Public Key Index	1	Identifies the Certification Authority Public Key in conjunction with the RID	b
Certification Authority Hash Algorithm Indicator	1	Identifies the hash algorithm used to produce the Hash Result in the digital signature scheme	b
Certification Authority Public Key Algorithm Indicator	1	Identifies the digital signature algorithm to be used with the Certification Authority Public Key	b
Certification Authority Public Key Modulus	~~V~~var. (max 248)	Value of the modulus part of the Certification Authority Public Key	b
Certification Authority Public Key Exponent	1 or 3	Value of the exponent part of the Certification Authority Public Key, equal to 3 or $2^{16} + 1$	b
Certification Authority Public Key Check Sum [14]	~~20~~var.	A check value calculated on the concatenation of all parts of the Certification Authority Public Key (RID, Certification Authority Public Key Index, Certification Authority Public Key Modulus, Certification Authority Public Key Exponent) using SHA-1. Alternatively, one of the hash algorithms from section B2.3 may be used	b

Table 27: Minimum Set of Certification Authority Public Key Related Data Elements to be Stored in Terminal (RSA)

[40] Only necessary if used to verify the integrity of the Certification Authority Public Key.

Field Name	Length	Description	Format
Registered Application Provider Identifier (RID)	5	Identifies the payment system to which the Certification Authority Public Key is associated	b
Certification Authority Public Key Index	1	Identifies the Certification Authority Public Key in conjunction with the RID	b
Certification Authority Public Key Algorithm Suite Indicator	1	Identifies the algorithm suite to be used with the Certification Authority Public Key. See section B2.4.1.	b
Certification Authority Public Key	$2*N_{FIELD}$	Representation of Certification Authority Public Key (xy-coordinates of Certification Authority Public Key point) on the curve identified by the Certification Authority Public Key Algorithm Suite Indicator.	b
Certification Authority Public Key Check Sum [40]	var.	A check value calculated on the concatenation of the above data elements using a hash algorithm. The same hash algorithm may be used for all keys in the terminal or the hash algorithm associated with the Certification Authority Public Key Algorithm Suite may be used	b

Table 27a: Minimum Set of Certification Authority Public Key Related Data Elements to be Stored in Terminal (ECC)

Section 11.2.3 is updated and combined with section 11.2.4 as below:

11.2.3 Certification Authority Public Key ~~Usage~~ Withdrawal

~~The usage of a Certification Authority Public Key during a transaction shall be as specified in this specification.~~

When a payment system has decided to withdraw ~~revoke~~ one of its Certification Authority Public Keys, an acquirer shall ensure that this Certification Authority Public Key can no longer be used in its terminals for offline static and dynamic data authentication during transactions as of a certain date.

It is recommended that acquirers do not implement ~~expiry~~ expiration dates coded into the terminals but instead remove keys according to the ~~expiry~~ expiration dates as indicated by the payment systems.

The following principles apply for the withdrawal by an acquirer of Certification Authority Public Keys from its terminals:

- The terminal shall be able to verify that it received the withdrawal notification error-free from the acquirer (or its agent).
- The terminal shall be able to verify that the received withdrawal notification originated from its legitimate acquirer (or its agent).
- The acquirer shall be able to confirm that a specific Certification Authority Public Key was indeed withdrawn correctly from its terminals.

~~For more details on Certification Authority Public Key revocation and the corresponding timescales involved, see section 10.~~

Other than in the extreme case of a Certification Authority Public Key compromise, keys are expected to be withdrawn on the previously publicised expiration dates.

To assist with this, EMVCo will conduct annual review sessions for Certification Authority Public Key pair strength evaluation, using state of the art information and analysis from the fields of computer science, cryptography, and data security.

All Certification Authority Public Keys will have December 31st as planned expiration date.

Public Key expiration dates will be set well in advance to allow card portfolio lifetimes to meet business needs and Issuers should ensure that IC Cards expire no later than the expiration date of the Issuer Public Key Certificates.

Acquirers need to remove the Certification Authority Public Keys from service in their terminals within a specific grace period after expiration. This is expected to be six months starting from the planned expiration date (until June 30th of the following calendar year) but may be deferred at payment system discretion.

It may be noted that much of this material relates to the introduction of longer RSA keys, necessary as the shorter keys lose security strength. This sequence will end when the 1984 bit keys reach an expiration date. Whilst ECC keys are expected to be stable in the long term, the ability to install and withdraw keys is still necessary to allow for infrastructure changes, such as the emergence of a new domestic payment system.

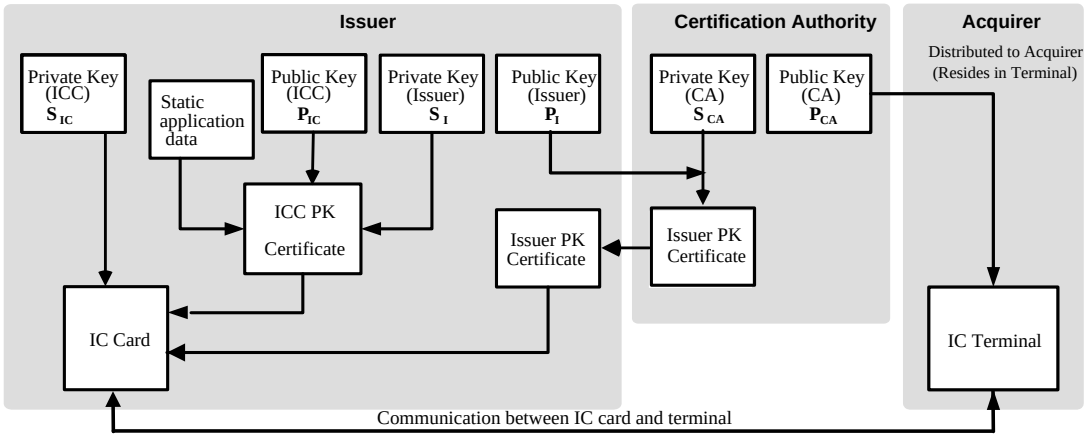
Section 12 is added after Section 11:

12 Offline Dynamic Data Authentication using ECC (XDA)

This section describes Extended Data Authentication (XDA), an offline dynamic data authentication mechanism that utilises ECC public key cryptography.

When performing XDA the ICC generates a digital signature that is verified by the terminal. This digital signature authenticates the ICC and confirms the legitimacy of the Application Cryptogram (AC) and other critical data originating from the ICC and terminal. This precludes the counterfeiting of any such card and provides data integrity between the card and the terminal. XDA signatures are generated by the ICC in response to the GENERATE AC command.

XDA requires the existence of a certification authority, a highly secure cryptographic facility that 'signs' the Issuer's Public Keys. If a terminal application is configured to be XDA capable then the terminal shall contain the appropriate certification authority's public key(s) for that application. This specification permits multiple AIDs to share the same Certification Authority Public Keys. The relationship between the data and the cryptographic keys is shown in Figure 13b. See also section 11.2.



Card provides to Terminal:

- Issuer PK Certificate (P_I signed by the CA S_{CA})
- ICC PK Certificate (P_{IC} and static application data signed by Issuer S_I)
- Card and terminal dynamic data and digital signature (dynamic data signed by Card S_{IC})

Terminal:

- Uses P_{CA} to verify that the Issuer's P_I was signed by CA
- Uses P_I to verify that Card P_{IC} and static application data were signed by Issuer
- Uses P_{IC} to verify the card's signature on the dynamic data

Figure 13b: Diagram of offline dynamic data authentication

ICCs that support XDA contain an ICC Private Key that the ICC uses for generating the Signed Dynamic Application Data.

ICCs that support XDA shall generate the data element listed in Table 27b:

Data Element	Length	Description
Signed Dynamic Application Data	var.	Generated by the ICC using the private key that corresponds to the public key authenticated in the ICC Public Key Certificate. This data element is a digital signature covering critical ICC and terminal data, as described in section 12.5.

Table 27b: Data Element Generated for XDA

XDA uses the signature scheme specified in Annex A2.2 and Annex B2.2.
Section 12.1 contains an overview of the keys and certificates involved in the XDA process. Sections 12.2, 12.3 and 12.4 specify the initial steps in the process, namely:

- Retrieval of the Certification Authority Public Key by the terminal.
- Authentication and Recovery of the Issuer Public Key by the terminal.
- Authentication and Recovery of the ICC Public Key by the terminal.

If the first of these steps fails then the CA ECC key is considered to be missing and therefore XDA is considered to have failed. If either of the last two of these steps fail then ECC key recovery is considered to have failed and therefore XDA is considered to have failed. Terminal actions in case of ECC key recovery failures are addressed in section 6.3 of Book 4. Note that ECC key recovery failure is not reflected in the Terminal Verification Results before the GENERATE AC command is sent and so cannot influence Terminal Action Analysis until after the GENERATE AC command is sent.

The final step of the XDA process consists of generation by the ICC and verification by the terminal of the Signed Dynamic Application Data. Section 12.5 specifies the dynamic signature generation and verification processes for XDA.

If dynamic signature verification fails then XDA is considered to have failed. XDA shall only be considered successful if CA key retrieval is successful and ECC key recovery is successful and XDA signature verification is successful.

12.1 Keys and Certificates

To support offline dynamic data authentication an ICC shall have its own public key pair consisting of a private signature key and the corresponding public verification key. The ICC Public Key shall be stored in the ICC in a public key certificate tag '9F46'. To support offline data encipherment (PIN or biometrics) a separate public key certificate is required with an independent Algorithm Suite Indicator (see section 13). Two certificates are still required even if the same key pair is used for both XDA and ODE[40a].

More precisely, a three-layer public key certification scheme is used. Each ICC Public Key is certified by its issuer, and the Certification Authority certifies the Issuer Public Key. This implies that, for the verification of an ICC signature, the terminal first needs to verify two certificates in order to authenticate and recover the ICC Public Key, which is then employed to verify the ICC's dynamic signature (or to encipher offline data).

To generate the Issuer Public Key Certificate, the Certification Authority signs the Issuer Public Key data using the Certification Authority Private Key S_{CA} with the signature scheme specified in Annex A2.2.

To generate the ICC Public Key Certificate, the Issuer signs the ICC Public Key data using the Issuer Private Key S_I with the signature scheme specified in Annex A2.2.

Aside from the Certification Authority Public Key, all the information necessary for ICC Public Key authentication is specified in Table 27c and stored in the ICC. With the exception of the RID, which can be obtained from the AID, this information may be retrieved with the READ RECORD command.

All public keys are ECC keys (a hybrid approach using a combination of RSA and ECC keys is not supported).

[40a] This is unlike RSA certificates, where if a public key is used for both ODA and ODE then only a single public key certificate is needed.

Tag	Name	Description
—	Registered Application Provider Identifier (RID)	5 byte identifier obtained from the Application Identifier (AID)
'8F'	Certification Authority Public Key Index	1 byte which indicates which of the application's Certification Authority Public Keys that reside in the terminal is to be used with this ICC.
'90'	Issuer Public Key Certificate	Includes a digital signature on the Issuer Public Key and other data that enables the terminal to obtain an authenticated copy of the Issuer Public Key and other data as described in section 12.3
'9F46'	ICC Public Key Certificate	Includes a digital signature on the ICC Public Key and other data that enables the terminal to obtain an authenticated copy of the ICC Public Key and other data as described in section 12.4.
'9F2D'	ICC Public Key Certificate for ODE	Includes a digital signature on the ICC Public Key for ODE and other data that enables the terminal to obtain an authenticated copy of the ICC Public Key for ODE and other data as described in section 12.4. (only needed if ODE is supported)
—	Static data to be authenticated	The ICC-resident static data to be authenticated as specified in section 10.3 of Book 3.

Table 27c: ICC Data Required for Public Key Authentication for XDA and ECC ODE

12.1.1 Certification Revocation List

The terminal may support a Certification Revocation List (CRL) that lists the Issuer Public Key Certificates that payment systems have revoked. If during XDA a concatenation of the RID and Certification Authority Public Key Index from the card and the Certificate Serial Number recovered from the Issuer Public Key Certificate is on this list, ECC key recovery fails as described in section 12.3, Step 7.

The requirements for the CRL are listed in section 5.1.2.

12.2 Retrieval of Certification Authority Public Key

To support offline dynamic data authentication (and offline data encipherment), terminals store Certification Authority Public Keys for each RID along with associated key-related information (see section 11.2.2). The terminal is able to locate any such key (and key-related information) given the RID and Certification Authority Public Key Index as provided by the ICC.

Using the RID and Certification Authority Public Key Index read from the card, the terminal shall identify and retrieve the terminal-stored Certification Authority Public Key and associated key-related information, including the Certification Authority Public Key Algorithm Suite Indicator.

If the terminal does not have the public key associated with this index and RID or the index is missing (per section 7.5 of Book 3), then the terminal shall consider the CA ECC key missing and XDA (and/or ODE) processing is considered to have failed. Otherwise, the terminal shall continue processing with the steps in the following section.

12.3 Authentication and Recovery of Issuer Public Key

The Issuer Public Key is contained in the Issuer Public Key Certificate (tag '90') included in data read from the ICC. If the Issuer Public Key Certificate is missing (per section 7.5 of Book 3) then the terminal shall consider ECC key recovery as failed and XDA (and/or ODE) processing is considered to have failed.

Authentication and recovery of the Issuer Public Key consists of verification of the signature in the Issuer Public Key Certificate using the Certification Authority Public Key and the algorithms indicated by Certification Authority Public Key Algorithm Suite Indicator, verification of the format and values of the data elements found in the certificate, and recovery of the y-coordinate from the x-coordinate. This process is described below.

The result of these verifications is either a valid and authentic copy of the Issuer Public Key, or failure.

Table 27d shows the Issuer Public Key Certificate and is the data signed by the Certification Authority Private Key and the resulting signature. That is, the concatenation of the data elements 1 through 9 in this table forms the input to the signature function in Annex A2.2.2.

	Name	Description	Length (bytes)
1	Issuer Certificate Format	Always Hex value '12' for this version of the specifications.	1
2	Issuer Certificate Encoding	Always Hex value '00' for this version of the specifications.	1

	Name	Description	Length (bytes)
3	Issuer Identifier	Uniquely identifies the Issuer in the context of the RID (leftmost 3-10 digits from the PAN padded to the right with hex 'F's).	5
4	Issuer Public Key Algorithm Suite Indicator	Identifies the algorithm suite to be used with the Issuer Public Key when verifying issuer signatures. See section B2.4.	1
5	Issuer Certificate Expiration Date	Issuer Certificate Expiration Date (YYYYMMDD, UTC).	4
6	Issuer Certificate Serial Number	Number assigned to the issuer certificate that is unique amongst all certificates created by the payment system key that created the issuer certificate.	3
7	RID	Payment System identified in the AID (first 5 bytes).	5
8	Certification Authority Public Key Index	In conjunction with the RID, identifies which of the Kernel application's Certification Authority Public Keys (and associated algorithms) to use when verifying the issuer certificate.	1
9	Issuer Public Key	Representation of Issuer Public Key (x-coordinate of Issuer Public Key point) on the curve identified by the Issuer Public Key Algorithm Suite Indicator.	N_{FIELD}
10	Issuer Public Key Certificate Signature	A digital signature on data objects 1 through 9 of this table. Verified using the Certification Authority Public Key and algorithms identified by data object 8.	N_{SIG}

Table 27d: Issuer Public Key Certificate tag '90' (ECC)

Note: When using EC-SDSA, the length N_{SIG} of a digital signature is equal to the length N_{HASH} of the hash plus the length of a field element. Thus, if using P-256 and a 256-bit hash algorithm (such as SHA-256), then $N_{HASH} = N_{FIELD} = 32$ and $N_{SIG} = N_{HASH} + N_{FIELD} = 64$. Furthermore, for item 10 in Table 27d the lengths N_{SIG}, N_{HASH}, and N_{FIELD} are determined by the Certification Authority Public Key Algorithm Suite Indicator whereas for item 9 in Table 27d the length N_{FIELD} is determined by the Issuer Public Key Algorithm Suite Indicator.

To authenticate the Issuer Public Key the terminal shall perform the following steps:

1. Check that the length in bytes of the Issuer Public Key Certificate as defined in Table 27d is equal to $N_{FIELD} + N_{SIG} + 21$. If this fails, ECC key recovery has failed.
2. Check the Issuer Certificate Format. If it is not '12', ECC key recovery has failed.
3. Check the Issuer Certificate Encoding. If it is not '00', ECC key recovery has failed.

4. Verify that the Issuer Identifier matches the leftmost 3 - 10 digits of the PAN (tag '5A') (allowing for the possible padding of the Issuer Identifier with hexadecimal 'F's). If not, ECC key recovery has failed.
5. Verify that the date specified in the Issuer Certificate Expiration Date is equal to or later than today's date. If the Issuer Certificate Expiration Date is earlier than today's date, the certificate has expired, in which case ECC key recovery has failed.
6. Verify that the RID and the Certification Authority Public Key Index in the certificate matches the RID and the Certification Authority Public Key Index used to retrieve the Certification Authority Public Key. If not, ECC key recovery has failed.
7. Verify that the concatenation of RID, Certification Authority Public Key Index, and Issuer Certificate Serial Number is valid. If not, ECC key recovery has failed.[40b]
8. Verify that the Issuer Public Key Algorithm Suite Indicator is an approved identifier in accordance with Annex B2.4. If not, then ECC key recovery has failed.
9. Concatenate the first 9 data elements of the Issuer Public Key Certificate
10. Using the Certification Authority Public Key and the algorithms indicated by the associated Certification Authority Public Key Algorithm Suite Indicator, verify as specified in Annex A2.2.3 the Issuer Public Key Certificate Signature contained in the Issuer Public Key Certificate (tag '90') with the concatenated data from step 9 being the MSG as specified in Annex A2.2.3. If signature verification fails then ECC key recovery has failed.
11. Recover the y-coordinate of the Issuer Public Key point using the Point4x() function applied to the x-coordinate obtained from the Issuer Public Key field of the certificate (item 9 in Table 27d) as described in section B2.2.1(e).

If the above steps were all successful then authentication and recovery of the Issuer Public Key is successful and the terminal shall continue with the steps in the following section. If any of the above steps were not successful (i.e. ECC key recovery has failed) then XDA (and/or ODE) processing is considered to have failed.

12.4 Authentication and Recovery of ICC Public Key

The ICC Public Key is contained in the ICC Public Key Certificate (tag '9F46') included in data read from the ICC. If ODE is supported (see section 13) then the ICC Public Key for ODE is contained in the ICC Public Key Certificate for ODE (tag '9F2D') included in data read from the ICC. If the ICC Public Key Certificate is missing (per section 7.5 of Book 3) then the terminal shall consider ECC key recovery as failed and XDA (and/or ODE) processing is considered to have failed.

[40b] This step is optional and is to allow the revocation of the Issuer Public Key Certificate against a list that may be kept by the terminal.

Authentication and recovery of the ICC Public Key consists of verification of the signature in the ICC Public Key Certificate using the Issuer Public Key and the algorithms indicated by the Issuer Public Key Algorithm Suite Indicator, verification of the format and values of the data elements found in the certificate, and recovery of the y-coordinate from the x-coordinate. This process is described below (under ICC Public Key Authentication and Recovery).

As well as the ICC Public Key, the signature in the ICC Public Key Certificate is intended to protect the authenticity of certain other static card data. So verification of the signature additionally involves the verification of the format and values of other data elements protected by the certificate and this includes the Static Data to be Authenticated as specified in Book 3 section 10.3. This process is described below (under ICC Public Key Authentication and Recovery). The same certificate format and certificate verification process is used for ICC Public Key Certificates for ODE.

The result of these verifications is either a valid and authentic copy of the ICC Public Key and the Static Data to be Authenticated, or ECC key recovery failure.

	Name	Description	Length (bytes)
1	ICC Certificate Format	Always Hex value '14' for this version of the specifications.	1
2	ICC Certificate Encoding	Always Hex value '00' for this version of the specifications.	1
3	ICC Public Key Algorithm Suite Indicator	Identifies the algorithm suite to be used with the certified ICC Public Key (a Signature Algorithm Suite in the case of tag '9F46' and an ODE Algorithm Suite in the case of tag '9F2D'. See sections B2.4.1 and B2.4.2 respectively).	1
4	ICC Certificate Expiration Date	ICC Certificate Expiration Date (YYYYMMDD, UTC).	4
5	ICC Certificate Expiration Time	ICC Certificate Expiration Time (HHMM, UTC).	2
6	ICC Certificate Serial Number	Number assigned to the ICC certificate that is unique amongst all certificates created by the issuer key that created the ICC certificate	6
7	ICCD Hash Encoding	For this version of the specifications, always Hex value '00', indicating TLV encoding of the input is used when computing the ICCD Hash.	1
8	ICCD Hash Algorithm Indicator	Identifies the hash algorithm used to calculate the ICCD Hash. See section B2.3.	1

	Name	Description	Length (bytes)
9	ICCD Hash	Hash of the Issuer Certified Card Data (ICCD) using the hash function identified by the ICCD Hash Algorithm Indicator.	N_{HASH}
10	ICC Public Key	Representation of ICC Public Key (x-coordinate of ICC Public Key point) on the curve identified by the ICC Public Key Algorithm Suite Indicator.	N_{FIELD}
11	ICC Public Key Certificate Signature	A digital signature on data objects 1 through 10 of this table. Verified using the Issuer Public Key obtained from the certificate in Table 27d and the algorithms identified by the Issuer Public Key Algorithm Suite Indicator.	N_{SIG}

Table 27e: ICC Public Key Certificate tag '9F46' (XDA) and tag '9F2D' (ODE)

Note: When using EC-SDSA the length N_{SIG} of a digital signature is equal to the length of the hash plus the length of a field element. Thus, if using P-256 and a 256-bit hash algorithm (such as SHA-256) then $N_{HASH} = N_{FIELD} = 32$ and $N_{SIG} = N_{HASH} + N_{FIELD} = 64$. Furthermore, for item 11 in Table 27e the lengths N_{SIG}, N_{HASH}, and N_{FIELD} are determined by the Issuer Public Key Algorithm Suite Indicator whereas for item 10 in Table 27e the length N_{FIELD} is determined by the ICC Public Key Algorithm Suite Indicator.

ICCD Hash

As part of ICC Public Key authentication, the terminal shall check that the hash of the Issuer Certified Card Data (ICCD) is equal to the ICCD Hash included in the ICC Public Key Certificate (see Table 27e).

The ICCD is the Static Data to be Authenticated which is equal to the concatenation of records identified by the AFL as participating in Offline Data Authentication, the tag, length and value of Application Interchange Profile, the tag, length and value of Application Identifier (AID) – terminal, and the tag, length, and value of PDOL (if received from the card) as specified in section 10.3 of Book 3.

It is expected that authenticated records will include Application Primary Account Number (PAN), Application Primary Account Number (PAN) Sequence Number, and Application Expiration Date.

Note: For this version of the specifications, the encoding method used for the ICCD is restricted to TLV encoding and so the ICCD Hash will be computed over the data objects formatted in TLV representation as described in section 10.3 of Book 3.

ICC Public Key Authentication and Recovery

The terminal shall perform the following steps:

1. Check that the first byte of the certificate is '14' and that the length in bytes of the ICC Public Key Certificate as defined in Table 27e is equal to 17 + N_{HASH} + N_{FIELD} + N_{SIG}. If this fails, ECC key recovery has failed.
2. Check the ICC Certificate Encoding. If it is not '00', ECC key recovery has failed.
3. Check that the certificate expiration time and date recovered from the certificate is later than the current time and date. If this is not the case then the certificate has expired and ECC key recovery has failed.
4. Verify that the ICC Public Key Algorithm Suite Indicator is an approved identifier in accordance with Annex B2.4 (approved for XDA if tag '9F46' or approved for ODE if tag '9F2D'). If not, then ECC key recovery has failed.
5. Verify that the ICCD Hash Algorithm Indicator is an approved identifier in accordance with Annex B2.3). If not, then ECC key recovery has failed.
6. Check that the hash of the Issuer Certified Card Data (ICCD) using the hash algorithm identified by ICCD Hash Algorithm Indicator is equal to the ICCD Hash included in the ICC Public Key Certificate (see Table 27e). If this check fails then ECC key recovery has failed.
7. Concatenate the first 10 data elements of the ICC Public Key Certificate.
8. Using the Issuer Public Key and the algorithms indicated by the associated Issuer Public Key Algorithm Suite Indicator, verify as specified in Annex A2.2.3 the Digital Signature contained in the ICC Public Key Certificate with the concatenated data from step 7 being the MSG as specified in Annex A2.2.3. If signature verification fails, ECC key recovery has failed.
9. Recover the y-coordinate of the ICC Public Key point using the Point4x() function applied to the x-coordinate obtained from the ICC Public Key field of the certificate (item 10 in Table 27e) as described in section B2.2.1(e).

If the above steps were all successful then authentication and recovery of the ICC Public Key is successful and the terminal shall continue with the steps in the following section (in case of tag '9F46') or section 13.2 (in case of tag '9F2D'). If any of the above steps were not successful (i.e. ECC key recovery has failed) then XDA processing is considered to have failed (in case of tag '9F46') or ODE processing is considered to have failed (in case of tag '9F2D').

12.5 XDA Signature Generation and Verification

Before the terminal can verify the XDA signature it needs to authenticate and recover the relevant public keys as described in sections 12.3 and 12.4. The authentication and recovery of the public keys may happen at any time before processing the response to the first GENERATE AC command.

12.5.1 Requesting, generating and verifying an XDA signature

Conditions for requesting an XDA signature are addressed in section 9.3 of Book 3 and section 6.3.2.2 of Book 4.

The terminal always requests an XDA signature in the GENERATE AC command if XDA is selected. Thus, an XDA signature is requested even if an AAC is requested and even if CA ECC key retrieval has failed or ECC key recovery has failed or, in the case of a second GENERATE AC command, verification of the XDA signature requested in the first GENERATE AC command has failed.[40c]

The ICC generates an XDA signature according to section 12.5.2 if the terminal has requested an XDA signature in the GENERATE AC command.

The terminal verifies an XDA signature according to section 12.5.3 if the terminal has requested and received an XDA signature, the Application Cryptogram returned is a TC or ARQC and XDA has not already failed. The terminal does not attempt XDA signature verification if an AAC is returned or if XDA has already failed.

12.5.2 Dynamic Signature Generation

The generation of the XDA signature takes place in the following steps.

10. The terminal issues a GENERATE AC command with the 'XDA signature requested' bits in the GENERATE AC command set to '01' according to sections 6.5.5.2 and 9.3 of Book 3.
11. The ICC performs the following steps:
    a. The ICC generates the AC.
    b. The ICC generates a digital signature as described in Annex A2.2.2 on the data specified in Table 27f using its ICC Private Key S_{IC} in conjunction with the corresponding algorithms. The resulting digital signature is called the Signed Dynamic Application Data.

Creation of Signed Dynamic Application Data

The ICC shall sign the dynamic application data shown in Table 27f using the ICC Private Key and the signature function in Annex A2.2.2. The input (MSG) to the signature function is the concatenation of the data elements in Table 27f and the output is the Digital Signature in the Signed Dynamic Application Data.

Field Name	Length	Description	Format
Signed Data Format	1	Hex value '15'	b

[40c] The reason for always requesting an XDA signature, even if an AAC is requested, is to avoid complicated logic for circumstances that will rarely occur.

PDOL data (1st GEN AC only)	$L_{PDOL\ data}$	Values of the data elements specified by the PDOL and in the order they appear in the PDOL and sent by the terminal in the GET PROCESSING OPTIONS command (Not present if no PDOL was present in final SELECT)	b
CDOL1 data (1st GEN AC only)	$L_{CDOL1\ data}$	Values of the data elements specified by the CDOL1 in the order they were sent by the terminal in the first GENERATE AC command.	b
CDOL2 data (2nd GEN AC only)	$L_{CDOL2\ data}$	(Second GENERATE AC only) Values of the data elements specified by the CDOL2 in the order they were sent by the terminal in the second GENERATE AC command.	b
Response data	L_{RESP}	TLV-encoded data in the response to the GEN AC command not including the Signed Dynamic Application Data. See Table 27h.	b

Table 27f: Dynamic Application Data to be Signed (XDA)

The Response Data in Table 27f shall be exactly (including their order) as returned in the response to the GENERATE AC command, but not including the Signed Dynamic Application Data.

The Signed Dynamic Application Data consists of the concatenation of the Signed Data Format and the Digital Signature. This is illustrated in Table 27g.

Field Name	Length	Description	Format
Signed Data Format	1	Hex value '15'	b
Digital Signature	N_{SIG}	This is the ICC ECC digital signature (*r*, *s*).	b

Table 27g: Format of XDA Signed Dynamic Application Data

The ICC response to GENERATE AC commands with XDA signature is coded according to format 2 as specified in section 6.5.5.4 of Book 3 (constructed data object with tag '77') and contains at least the mandatory data objects (TLV coded in the response) specified in Table 27h, and optionally the Issuer Application Data and other data objects.

Tag	Length	Value	Presence

'9F27'	1	Cryptogram Information Data	M
'9F36'	2	Application Transaction Counter	M
'9F26'	8	Application Cryptogram	M
'9F10'	var. up to 32	Issuer Application Data	O
	var.	Other Data Objects	O
'9F4B'	N_{SIG} + 1	Signed Dynamic Application Data	M

Table 27h: Data Objects Included in Response to GENERATE AC with XDA signature

12.5.3 Dynamic Signature Verification

This section continues the terminal processing from Step 1 of section 12.5.2.

The terminal receives and parses the GENERATE AC response.

Note: The terminal can determine the type of Application Cryptogram by inspecting the CID in the response.

The terminal shall continue XDA processing and verify the XDA signature in the following steps:

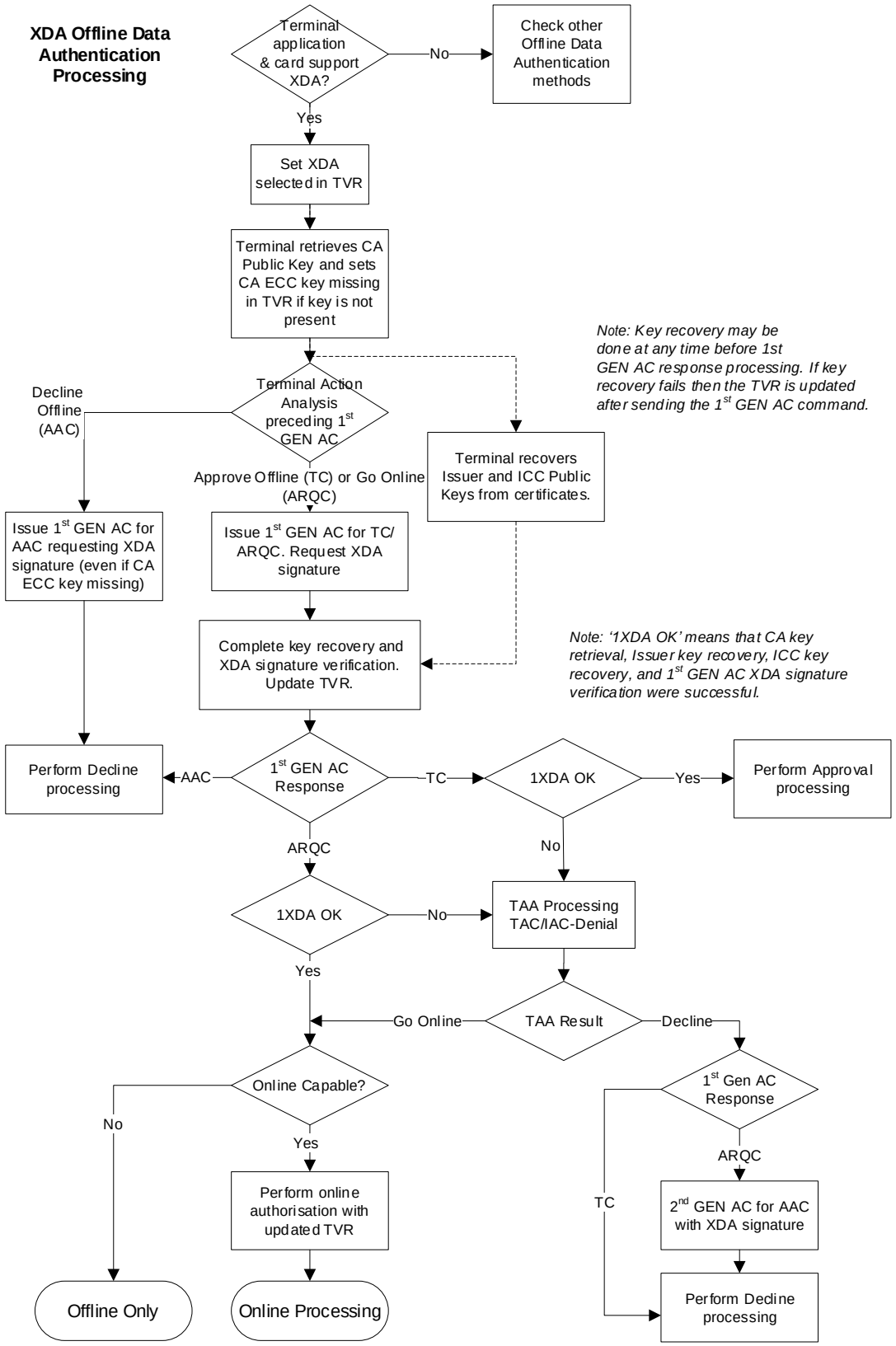
Verification of Signed Dynamic Application Data

1. Parse the Signed Dynamic Application Data according to Table 27g. If this fails, XDA signature verification has failed.
2. Check the Signed Data Format. If it is not '15', XDA signature verification has failed.
3. Verify the signature (r, s) from Table 27g on the data in Table 27f as specified in Annex A2.2.3, using the ICC Public Key and the associated ICC Public Key Algorithm Suite. If the verification fails, XDA signature verification has failed.

If all the above steps were executed successfully, XDA signature verification was successful. Otherwise XDA signature verification has failed.

12.5.4 Sample XDA Flow

The figure on the following page is an example of how a terminal might perform XDA. This sample flow provides a generalised illustration of the concepts of XDA. It does not necessarily contain all required steps and does not show parallel processing (for example, overlapping certificate recovery and signature generation). If any discrepancies are found between the text and flow, the text shall be followed. Book 4 sections 6.3.2.2.1 to 6.3.2.2.4 describe the processing for XDA-related failures.



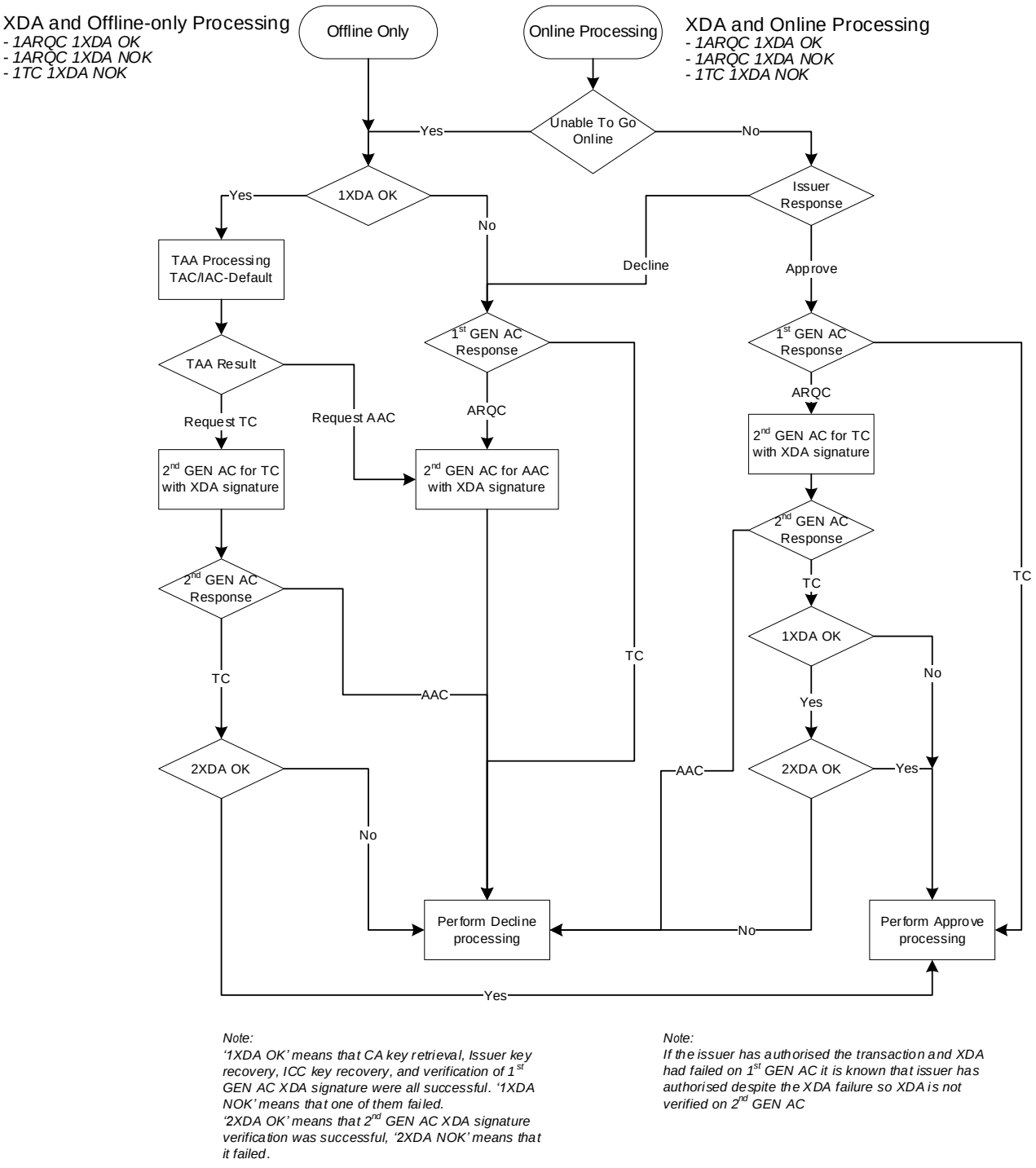


Figure 13c: XDA Sample Flow

Section 13 is added after Section 12:

13 Offline Data Encipherment using ECC

This section relates to the use of ECC for offline encipherment of data, in particular PINs and biometrics, to ensure the secure transfer of data from the CVM capture device to the ICC.

If supported, the ICC shall own a public/private key pair associated with offline data encipherment. The public key is then used by the CVM capture device or a secure component of the terminal (other than the CVM capture device) to derive symmetric keys using an approach based on ISO/IEC 11770-6. These are then used to encipher the input data in accordance with ISO/IEC 19772 Mechanism 5 (Encrypt-then-MAC).

If a PIN is to be enciphered then it is first formed into a PIN block as shown in section 6.5.12 of Book 3.

Note: It is not possible to mix RSA and ECC during a transaction as the card can only return one Certification Authority Public Key Index and one Issuer Public Key Certificate. Thus, it is not possible to perform ODE using ECC together with ODA using RSA, nor is it possible to perform ODE using RSA with ODA using the ECC based XDA.

13.1 Keys and Certificates

If offline data encipherment using ECC is supported by the ICC as indicated by the CVM List, then the ICC has a key pair consisting of a public encipherment key and the corresponding private decipherment key. This specification allows the ICC to use a dedicated ICC public key pair for ECC ODE, or the same ICC public key pair for both XDA and ECC ODE. However, in both cases the ICC will have a dedicated ICC Public Key Certificate for ODE tag '9F2D' containing the ICC Public Key and ICC Public Key Algorithm Suite Indicator (see Table 27e) which is authenticated and recovered using the same process as described in section 12.4.[40d]

13.2 PIN Encipherment

The terminal shall not encipher the PIN (step 3 below) until it has authenticated the ICC public key used for ECC ODE (sections 12.2, 12.3, and 12.4).

The exchange of an enciphered PIN between terminal and ICC shall take place in the following steps.

4. The PIN is entered in plaintext format on the PIN pad and a PIN block is constructed as defined in section 6.5.12 of Book 3.

[40d] ICC Public Key Certificates enforce key usage by including the appropriate Algorithm Suite Indicator.

5. The terminal issues a GET CHALLENGE command to the ICC to obtain an 8-byte unpredictable number UN_{ODE} from the ICC. When the response to the GET CHALLENGE command is anything other than an 8 byte data value with SW1 SW2 = '9000', then the terminal shall consider that the Offline Enciphered PIN (ECC) CVM has failed.

   Note: a fresh GET CHALLENGE command is issued irrespective of previous CVM processing that may also have used a GET CHALLENGE command.

Steps 1 and 2 may be performed in either order.

6. If Key Derivation has already been performed for the current transaction, then do not perform Key Derivation again and use the previously derived keys.

   If Key Derivation has not yet been performed for the current transaction, then:

   a) Generate an ephemeral key pair (d, R) as described in B2.2.4, for the curve identified by the certificate's ICC Public Key Algorithm Suite Indicator and set R_x to be the x-coordinate of R converted to a byte-string of length N_{FIELD} using I2OS() per B2.6.1.

   b) Perform Key Derivation as described in A2.3.2, with the UN_{ODE} generated in step 2 as the UN for key derivation UN_{KD}, the certificate's ICC Public Key Algorithm Suite Indicator for the Algorithm Suite Indicator, d from step 3a), and the ICC Public Key for ODE recovered from the ICC Public Key Certificate for ODE (tag '9F2D' as specified in section 13.1) as the point Q, to derive keys and initialise the counter N.

7. Apply the ECC encryption function (see A2.3.3) indicated by the certificate's ICC Public Key Algorithm Suite Indicator (see B2.4), with K_1[40e] from step 3b) as K_X, K_2 from step 3b) as K_Y, and A set to null, to the data specified in Table 27i as MSG in order to obtain the ciphertext C. *If encryption fails, then* the terminal shall consider that the Offline Enciphered PIN (ECC) CVM has failed.

8. Concatenate R_x and C to obtain the Enciphered Data

   Enciphered Data := R_x || C

Field Name	Length	Description	Format
Data Header	1	Hex Value '7F'	b
PIN Block	8	PIN in PIN Block	b
ICC Unpredictable Number (UN_{ODE})	8	Unpredictable number obtained from the ICC with the GET CHALLENGE command	b

Table 27i: Data to be Enciphered for PIN Encipherment (ECC)

9. The terminal issues a VERIFY command including the Enciphered Data obtained in the previous step.

[40e] The derivation function generates four AES keys, but this specification only uses K_1 and K_2.

13.3 PIN Decipherment and Verification

The decipherment and verification of an enciphered PIN by the ICC shall take place in the following steps.

10. If the length of the Enciphered Data received in the VERIFY command is less than N_{FIELD} of the curve plus the length of a MAC (both as determined by the Algorithm Suite used by the ICC), the ICC shall return SW1 SW2 = '6984' and Offline Enciphered PIN (ECC) CVM has failed.
11. The ICC extracts R_x from the Enciphered Data received in the VERIFY command and calculates an ECC public key Q using the Point4x() function as described in section B2.2.1(e).

    Note: The length of R_x is N_{FIELD} for the curve determined by the Algorithm Suite used by the ICC. For P-256 the length of R_x is 32 bytes. For P-521 the length of R_x is 66 bytes.

    The remainder of the Enciphered Data, after R_x has been extracted, is C.
12. The ICC derives two 128-bit AES keys K_1 and K_2 using the key derivation process described in A2.3.2, with the UN_{ODE} as the UN for key derivation UN_{KD}, the ICC Public Key Algorithm Suite Indicator, the ICC private key for ECC ODE, and Q generated above.
13. To recover the plaintext data specified in Table 27i, the ICC applies the ECC decryption function (see A2.3.4), with K_1 and K_2 derived in step 3 above as K_X and K_Y respectively, C extracted in step 2 above and A as null.
14. If this decryption function fails, the ICC shall return SW1 SW2 = '6984' to indicate that Offline Enciphered PIN (ECC) CVM has failed.
15. The ICC verifies whether the ICC Unpredictable Number recovered in step 4 is equal to the ICC Unpredictable Number (UN_{ODE}) generated by the ICC with the GET CHALLENGE command. If this is not the case, the ICC shall return SW1 SW2 = '6984' to indicate that Offline Enciphered PIN (ECC) CVM has failed.
16. The ICC verifies whether the Data Header recovered in step 4 is equal to '7F'. If this is not the case, the ICC shall return SW1 SW2 = '6984' to indicate that Offline Enciphered PIN (ECC) CVM has failed.
17. The ICC verifies whether the PIN included in the PIN Block recovered in step 4 corresponds with the PIN stored in the ICC. If this is not the case, PIN verification has failed.

The order of Steps 6, 7 and 8 is mandatory.

If all the above steps were executed successfully, enciphered PIN verification was successful.

The terminal behaviour on receipt of the VERIFY response is described in Book 3 section 10.5.1.

13.4 Biometric Data Encipherment

The encipherment and exchange of biometric data between terminal and ICC takes place in the following steps, as illustrated in Book 3 Figure 12a.

1. The biometric template is captured on a biometric capture device and the Biometric Data Block (BDB) is constructed by the Biometric Processing Application.
2. The terminal issues a GET CHALLENGE command to the ICC to obtain an 8-byte unpredictable number UN_{ODE} from the ICC. When the response to the GET CHALLENGE command is anything other than an 8 byte data value with SW1 SW2 = '9000', then the terminal considers that the Offline Biometric CVM has failed.

   Note: a fresh GET CHALLENGE command is issued irrespective of previous CVM processing that may also have used a GET CHALLENGE command.

Steps 1 and 2 may be performed in any order.

3. If Key Derivation has already been performed for the current transaction, then do not perform Key Derivation again and use the previously derived keys.

   If Key Derivation has not yet been performed for the current transaction, then:

   a) Generate an ephemeral key pair (d, R) as described in B2.2.4, for the curve identified by the certificate's ICC Public Key Algorithm Suite Indicator and set R_x to be the x-coordinate of R converted to a byte-string of length N_{FIELD} using I2OS() per B2.6.1.

   b) Perform Key Derivation as described in A2.3.2, with the UN_{ODE} generated in step 2 as the UN for key derivation UN_{KD}, the certificate's ICC Public Key Algorithm Suite Indicator for the Algorithm Suite Indicator, d from step 3a), and the ICC Public Key for ODE recovered from the ICC Public Key Certificate for ODE (tag '9F2D' as specified in section 13.1) as the point Q, to derive keys and initialise the counter N.
4. Apply the ECC encryption function (see A2.3.3) indicated by the certificate's ICC Public Key Algorithm Suite Indicator (see B2.4), with K_1[40f] from step 3b) as K_X, K_2 from step 3b) as K_Y, and A set to null, to the data specified in Table 27j as MSG in order to obtain the ciphertext C. If encryption fails, then the terminal considers that the Offline Biometric CVM has failed.
5. Concatenate R_x and C to obtain the Enciphered Data

   Enciphered Data := R_x || C

Field Name	Length	Description	Format
ICC Unpredictable Number (UN_{ODE})	8	Unpredictable number obtained from the ICC with the GET CHALLENGE command	b

[40f] The derivation function generates four AES keys, but this specification only uses K_1 and K_2.

Field Name	Length	Description	Format
Biometric Solution ID Length	1	Length of Biometric Solution ID	b
Biometric Solution ID	value of Biometric Solution ID Length	As defined in Book 3 Table 43a	b
Biometric Type Length	1	Length of Biometric Type data element	b
Biometric Type	value of Biometric Type Length	As defined in Book 3 Table 43b	b
Biometric Subtype	1	As defined in Book 3 Table 43c	b
Biometric Data Block (BDB) Length	2	Length of BDB data element	b
Biometric Data Block (BDB)	value of BDB Length	A block of data with a specific format that contains information captured from a biometric capture device	b

Table 27j: Data to be enciphered for Biometric Encipherment

Note: The length bytes in Table 27j are simple binary bytes, and are *not* coded as BER-TLV length fields.

6. The terminal shall construct the Biometric Verification Data Template, and order the data elements as indicated in Table 27k.
   a) The value of Biometric Type is set as the plaintext value referenced in step 4, Table 27j
   b) The value of Biometric Solution ID is set as the plaintext value referenced in step 4, Table 27j
   c) The value of Enciphered Biometric Data is set as the value of the Enciphered Data generated in step 5

Tag	Length	Field Name
'81'	var.	Biometric Type
'90'	var.	Biometric Solution ID
'DF51'	var.	Enciphered Biometric Data

Table 27k: Biometric Verification Data Template

Note: The data objects in Table 27k are BER-TLV coded. The length of all TLV coded data objects are coded on the minimum number of bytes (that is, on 1 byte if < 128, on 2 bytes if in the range 128 to 255, and so on). See Book 3 Annex B.2 for BER-TLV coding rules. There shall be no '00' padding bytes before, between, or after the BER-TLV coded data objects.

7. The terminal shall construct and issue VERIFY command(s) as following:
    a) If the length of the value field of the Biometric Verification Data Template is no more than 255 bytes, the terminal shall set the value field of the Biometric Verification Data Template as the data field of the single VERIFY command.
    b) If the length of the value field of the Biometric Verification Data Template is more than 255 bytes, the terminal shall divide the value field of the Biometric Verification Data Template into 255 byte data blocks. The last data block may be 1 to 255 bytes in length. Then the terminal shall set these data blocks as the data fields of the commands and chain the commands using command chaining defined in Book 3 section 6.5.13.
    c) The terminal shall issue the VERIFY command(s).

13.5 Biometric Data Decipherment and Recovery

The decipherment and recovery of biometric data by the ICC takes place in the following steps.

1. If the length of the Enciphered Data received in the VERIFY command(s) is less than N_{FIELD} of the curve plus the length of a MAC (both as determined by the Algorithm Suite used by the ICC), the ICC shall return SW1 SW2 '6984' and Offline Biometric CVM has failed.
2. The ICC extracts R_x from the Enciphered Data received in the Verify command, and calculates an ECC public key Q using Point4x() function as described in section B2.2.1(e).

    Note: The length of R_x is N_{FIELD} for the curve determined by the Algorithm Suite used by the ICC. For P-256 the length of R_x is 32 bytes. For P-521 the length of R_x is 66 bytes.

    The remainder of the Enciphered Data, after Rx has been extracted, is C.
3. The ICC derives two 128-bit AES keys K_1 and K_2 using the key derivation process described in A2.3.2, with the UN_{ODE} as the UN for key derivation UN_{KD}, the ICC Public Key Algorithm Suite Indicator, the ICC private key for ECC ODE, and Q generated above.
4. To recover the plaintext data specified in Table 27j, the ICC applies the ECC decryption function (see A2.3.4), with K_1 and K_2 derived in step 3 above as K_X and K_Y respectively, C extracted in step 2 above and A as null.
5. If this decryption function fails, the ICC shall return SW1 SW2 = '6984' to indicate that Offline Biometric CVM has failed.
6. The ICC verifies whether the ICC Unpredictable Number recovered in step 4 is equal to the ICC Unpredictable Number (UN_{ODE}) generated by the ICC with the GET CHALLENGE command. If they are not the same, Offline Biometric CVM has failed.

7. The ICC verifies whether the Biometric Solution ID recovered in step 4 is equal to the Biometric Solution ID (tag '90') included in the Biometric Verification Data Template. If they are not the same, Offline Biometric CVM has failed.
8. The ICC verifies whether the Biometric Type recovered in step 4 is equal to the Biometric Type (tag '81') included in the Biometric Verification Data Template. If they are not the same, Offline Biometric CVM has failed.

If all the above steps were executed successfully, enciphered biometric recovery was successful.

In Section A.1.2.2, the Description 2. MAC Session Key is updated as:

In accordance with ISO/IEC 9797-1 Algorithm 5 (CMAC), two 128-bit sub-keys K_1 and K_2 are also derived.

Let Z be 16-bytes set to zero and let C be equal to Z except with its least significant bits set to '10000111b' (C is a CMAC-defined constant for a 16-byte block cipher).

$L := ALG(K_S)[Z]$

$K_1 = L << 1$. If $msb(L) = 1$ then $K_1 := K_1 \oplus C$

$K_2 = K_1 << 1$. If $msb(K_1) = 1$ then $K_2 := K_2 \oplus C$

where the following notation is used:

"<< 1" means left-shift 1 bit (i.e. the new 128-bit string comprises the 127 rightmost bits of the old string, appended with a zero bit)

Section A2.1 is updated as:

A2.1 Digital Signature Scheme ~~Giving Message Recovery~~ using RSA

This section describes the special case of the RSA digital signature scheme giving message recovery using a hash function according to ISO/IEC 9796-2, which is used in this specification for offline static and dynamic data authentication.

Section A2.2 is added after A2.1.3 as:

A.2.2 Digital Signature Scheme using ECC

This section describes the ECC version of the Schnorr digital signature scheme with appendix using a hash algorithm according to ISO/IEC 14888-3. It is the optimised version where only the x-coordinate is hashed rather than the x and y coordinates.

A2.2.1 Introduction

The signature generation and verification functions specified in A2.2 are for Signature Algorithm Suites '10' and '11' (see B2.4.1).

The Elliptic Curve Schnorr Digital Signature Algorithm (EC-SDSA) consists of a signature generation function and a signature verification function as described below. These functions make use of:

- The selected ECC curves specified in section B2.2.2.
- A hash algorithm Hash[ ] that maps a message of arbitrary length onto an N_{HASH}-byte hash code, where N_{HASH} is a fixed value for the given hash algorithm. For selected hash algorithms, see section B2.3.
- The combinations of ECC curves and hash algorithms are specified in section B2.4 and are determined by the Public Key Algorithm Suite Indicator associated with the key pair that generates the signature.

A2.2.2 Signature Generation

For EC-SDSA, the computation of a signature S on a message MSG of arbitrary length using a signer's private key d on an elliptic curve E over F_p with base point G of order n (note that for the curves defined in section B2.2.2, $n < p$) shall be as follows.

Input:

- Algorithm Suite Indicator identifying the curve E, group order n, and hash function Hash[ ] (output length N_{HASH}).
- d, the private signing key of the ICC.
- MSG (length $L \geq 0$), the message to be signed.

Output:

- $S := (r, s)$, the signature (a byte-string of length $N_{HASH} + N_{FIELD}$) on the message MSG.

Process:

a) Select a statistically unique and unpredictable integer k where $0 < k < n$. (This may be a random or a strong pseudo-random generation process and may use the string-to-integer techniques used in section B2.6.)

b) Compute $kG = (x_1, y_1)$ on the curve E. Convert the x_1 coordinate to a byte string B_1 of N_{FIELD} bytes according to the integer conversion function I2OS() in section B2.6.

c) Compute r = Hash[B_1 || MSG].

d) Compute $s' = (k + r'd) \bmod n$, where r' denotes r converted to an integer according to the integer conversion function OS2I() in section B2.6 then reduced mod n.

e) Convert the integer s' into a byte string s of N_{FIELD} bytes according to the integer conversion function I2OS() in section B2.6.

f) Output $S := (r, s)$.

That is, the signature S consists of a concatenated pair comprising the N_{HASH} byte string r and an integer formatted as the N_{FIELD} byte string s. The length in bytes of the Digital Signature is $N_{HASH} + N_{FIELD}$, where N_{HASH} is the output length of the hash algorithm in bytes and N_{FIELD} is the length of p (the field size) in bytes.

Note that the above Output is very slightly different to ISO/IEC 14888-3 since according to ISO/IEC 14888-3 the second element s of the signature S is simply an integer (and no encoding as a string of N_{FIELD} bytes is specified).

A2.2.3 Signature Verification

The function described in this section is used when verifying ECC signatures associated with Issuer Public Key Certificates, ICC Public Key Certificates and card-generated XDA signatures. The function uses curve information associated with the algorithm suite identified in the Certification Authority Public Key Related Data, the Issuer Public Key Certificate and the ICC Public Key Certificate, respectively.

The verification of an EC-SDSA signature S on a message MSG using the signer's public key point P on an elliptic curve E over F_p with base point G of order n (note that with the curves defined in section B2.2.2, $n < p$) and using a hash algorithm Hash[ ] shall be as follows.

Input:

- Algorithm Suite Indicator identifying the curve E, n, and Hash[ ] (output length N_{HASH}).
- S, a byte-string that is the (asserted) signature on the message MSG.
- P (or x-coordinate thereof), public key point of signer.
- MSG (length $L \geq 0$).

Output:

- Success or failure.

Process:

Preliminary processing: If only the x-coordinate of the signer's public key is available then the public key point P is first calculated using the Point4x() function as described in section B2.2.1(e).

Verification steps:

a) If the length in bytes of S is not $N_{HASH} + N_{FIELD}$, signature verification fails.

b) Parse S into the N_{HASH} byte string r and the N_{FIELD} byte string s. Thus $S = (r, s)$.

c) Convert s to an integer s' according to the integer conversion function OS2I() in section B2.6.

d) Convert r to a non-negative integer r' less than n by using OS2I() followed by reduction modulo n.

e) Check that $0 < s' < n$ and $0 < r'$. If either is not true then signature verification fails.

f) Compute $(x_2, y_2) = s'G - r'P$.

g) Convert the x-coordinate x_2 to a byte string B_2 of N_{FIELD} bytes according to the integer conversion function I2OS() in section B2.6.

h) Compute v = Hash[B_2 || MSG].

i) If $v = r$, then signature verification is successful; otherwise signature verification fails.

N_{HASH} is the output length of the hash algorithm in bytes and N_{FIELD} is the length of p (the field size) in bytes.

Section A2.3 is added after A2.2.3 as below:

A2.3 Encryption Scheme using ECC

A2.3.1 Introduction

The encryption and decryption functions specified in A2.3 are for ODE Algorithm Suites '00' and '01' (see B2.4.2).

Key derivation is only performed when encryption and decryption are first invoked during a transaction.

Key derivation initialises a global counter N, copies of which are maintained by the terminal and by the ICC.

Note: A single counter is used even if messages are exchanged in both directions because only a single message will be processed at a time and therefore synchronisation problems will not occur. If parallel processing were to be introduced in the future then two counters would be needed, one for each direction.

Note: Although the functions in A2.3 are specified in terms of the curves P-256 and P-521 and AES, the functions can be adapted to other curves and block ciphers.

A2.3.2 Key Derivation

Input:

- UN_{KD}, an 8-byte unpredictable number used for key derivation.
- Algorithm Suite Indicator identifying the curve, key agreement, and derivation method.
- d, a private ECC key.
- Q, a public key point on the curve.

Global variable:

- N, a 2-byte counter.

Output:

- K_1, a symmetric key used for encrypting/decrypting data sent from terminal to ICC.
- K_2, a symmetric key used for authenticating data sent from terminal to ICC.

- K_3, a symmetric key used for encrypting/decrypting data sent from ICC to terminal.
- K_4, a symmetric key used for authenticating data sent from ICC to terminal.

Process:

a) Set NumSKs : = 4.

b) Using the function PointMultiply() defined in B2.2.1(d), compute Z as the x-coordinate of the point dQ.

c) Convert Z from an integer to a bit string according to the integer conversion function I2BS() in section B2.6.
   - For P-256, Z is converted to a 256-bit string.
   - For P-521, Z is converted to a 640-bit string; the rightmost 521 bits of Z are a bit string representation of the x-coordinate of the point dQ and so the leftmost 119 bits of Z are zero 'padding bits'.

d) For P-256, write Z as $Z_0 \,||\, Z_1$ where Z_i is a 128-bit string for i = 0,1. This can be denoted as $Z_0 \,||\, Z_{t-1}$ with t = 2.

e) For P-521, write Z as $Z_0 \,||\, Z_1 \,||\, Z_2 \,||\, Z_3 \,||\, Z_4$ where Z_i is a 128-bit string for i = 0,1,2,3,4. This can be denoted as $Z_0 \,||\, Z_1 \,||\, Z_2 \,||\, Z_3 \,||\, Z_{t-1}$ with t = 5.

f) Apply CMAC (see section A1.2.2) to Z to produce an AES key derivation K_{DK} key as follows, using the 128-bit zero string (0^{128}) as the CMAC key:

$C_0 := AES(0^{128})[Z_0]$

$Z_{t-1} := Z_{t-1} \oplus$ 'CDD297A9DF1458771099F4B39468565C'

where the constant is according to CMAC and calculated as $AES(0^{128})[0^{128}] << 1$

for i from 1 to t-1, let $C_i := AES(0^{128})[C_{i-1} \oplus Z_i]$

$K_{DK} := C_{t-1}$

g) Derive K_i for i = 1 to NumSKs:

$K_i := AES(K_{DK})$['0i' || DerData]

where for this specification DerData is the 15-bytes SKD_VERSION || '00' || T || '02' || '00' where
   - SKD_VERSION is the 1-byte '01'
   - T is the 11 byte UN_{KD} || 'A5A5A5'
   - The final 2 bytes represent the length of the output (512 bits in the case of 128-bit session keys with NumSKs equal to 4).

h) Set the 2-byte variable N to the value '0000'. This is a global variable a copy of which is maintained by terminal and ICC. The terminal increments N after each successful encryption (see section A2.3.3) and the ICC increments N after each successful decryption (see section A2.3.4).

A2.3.3 Encryption and Authentication

This section describes both encryption of data and authentication of data. Authentication includes both the encrypted data, if present, and any additional unencrypted data. This allows the function to be used to authenticate data even if there is no data to encrypt.

Input:

- Algorithm Suite Indicator identifying the Authenticated Encryption method and Block Cipher.
- K_X, a symmetric key used for encrypting data.
- K_Y, a symmetric key used for authenticating data.
- MSG, a plaintext message.
- A, Additional Authentication Data.

Global variable:

- N, a 2-byte counter.

Output:

- C, the ciphertext (encrypted message) or an indication of failure.

Process:

The encryption process uses two derived keys because the encryption process is actually 'authenticated encryption' using a method called Encrypt-then-MAC (Mechanism 5 in ISO/IEC 19772).

The authenticated encryption mechanism described in this specification supports Additional Authenticated Data (AAD, data that is authenticated but not encrypted). Encryption of a plaintext MSG, denoted P, takes place in the following steps:

a) If $N = 65535$ then encryption has failed, so abort and report failure.

b) Generate the Starting Variable SV as the 16-byte all-zero string whose leftmost 2 bytes are set to the value of N.

c) If the message payload P is null then set C^* to be null and skip to step i).

d) If the bit-length of the message payload P is not a multiple of 8, then Encryption has failed, so abort and report failure.

e) If P is not a multiple of 16-bytes, then pad[43a] by appending r ($0 < r < 16$) zero-bytes to P.

f) Partition P into B 16-byte blocks:

$$P := P_1 \,||\, P_2 \,||\, \ldots \,||\, P_B$$

g) Encipher the message payload

$$C_i = P_i \oplus \text{AES}(K_X)[\, SV + ((i\text{-}1) \bmod 2^{112})] \text{ for } 1 <= i <= B$$

h) If padding had been applied then truncate the final rightmost r bytes from C_B thereby creating C_B', otherwise set $C_B' = C_B$. Set

$$C^* := C_1 \,||\, C_2 \,||\, \ldots \,||\, C_B'$$

[43a] This padding is only specified to simplify the specification in the case that the plaintext is not a multiple of 16-bytes and is not functionally necessary as any padding bytes added to the plaintext are truncated from the ciphertext before calculation of the MAC.

i) If the bit-length of A is not a multiple of 8, then encryption has failed, so abort and report failure.

j) Set:

$$D := \text{I2BS}(\text{len}(A)/8 , 64) \mid\mid A$$

k) Generate an 8-byte MAC using K_Y in accordance with section A1.2.2 over $D \mid\mid SV \mid\mid C^*$ and append this to C^* to create the ciphertext C

$$C := C^* \mid\mid \text{MAC}(D \mid\mid SV \mid\mid C^*).$$

l) Increment N.

m) Return the ciphertext C.

No result other than an indication of failure shall be returned in case of failure.

A2.3.4 Authentication and Decryption

This section describes both authentication of data and decryption of data. Authentication includes both encrypted data, if present, and any additional unencrypted data. This allows the function to be used to authenticate data even if there is no data to decrypt.

Input:

- Algorithm Suite Indicator identifying the Authenticated Encryption method and Block Cipher.
- K_X, a symmetric key used for decrypting data.
- K_Y, a symmetric key used for authenticating data.
- C, ciphertext of a message.
- A, Additional Authentication Data.

Global variable:

- N, a 2-byte counter.

Output:

- MSG, a plaintext message or an indication of failure.

Process:

The authenticated decryption mechanism described in this specification supports Additional Authenticated Data (data that is authenticated but not encrypted).

Decryption of the ciphertext C, takes place in the following steps:

a) If $N = 65535$ then decryption has failed, so abort and report failure.

b) If the length of the ciphertext C is less than the size of a MAC (8 bytes) then decryption has failed, so abort and report failure.

c) If the bit-length of A, is not a multiple of 8, then decryption has failed, so abort and report failure.

d) Set:

$$D := \text{I2BS}(\text{len}(A)/8\ ,\ 64)\ ||\ A$$

e) Partition the ciphertext C into C^* and an 8-byte candidate MAC value S'.

$$C := C^*\ ||\ S'$$

f) The bit-length of C^* should be a multiple of 8 and will be zero in the case that C^* is null (corresponding to a null plaintext). If the bit-length of C^* is not a multiple of 8, then decryption has failed, so abort and report failure.

g) Generate the Starting Variable SV as the 16-byte all-zero string whose leftmost 2 bytes are set to the value of N.

h) Calculate the 8-byte MAC S using K_Y in accordance with section A1.2.2 over $D\ ||\ SV\ ||\ C^*$.

i) If $S \neq S'$ then decryption has failed, so abort and report failure.

j) If C^* is null then the plaintext MSG is null so skip to step p).

k) If C^* is not a multiple of 16-bytes, then pad[43b] by appending r ($0 < r < 16$) zero-bytes to C^*.

l) Partition C^* into B 16-byte blocks:

$$C^* := C_1\ ||\ C_2\ ||\ \ldots\ ||\ C_B$$

m) Decrypt the ciphertext as

$$P_i = C_i \oplus \text{AES}(K_X)[\ SV + ((i\text{-}1) \bmod 2^{112})] \text{ for } 1 <= i <= B$$

n) If padding had been applied then truncate the final rightmost r bytes from P_B

o) The plaintext MSG is the byte-string

$$P := P_1\ ||\ P_2\ ||\ \ldots\ ||\ P_B$$

p) Increment N.

q) Return the plaintext message MSG.

No result other than an indication of failure shall be returned in case of failure.

[43b] This padding is only specified to simplify the specification in the case that the ciphertext is not a multiple of 16-bytes and is not functionally necessary as any padding bytes added to the ciphertext are truncated from the plaintext before output.

Sections B2.2 to B2.6 are added after B2.1.4 as below:

B2.2 Elliptic Curve Cryptography (ECC)

B2.2.1 Introduction

This annex describes elliptic curve cryptography (including algorithms and keys) to be used in conjunction with these specifications.

These specifications define two curves, P-256 and P-521 (see B2.2.2). P-521 is supported for contingency purposes only.

Elliptic curve cryptography is based on arithmetic on abstract mathematical objects called *elliptic curves*. An elliptic curve is built on (or *defined over*) a *field*. In the scope of this specification, this is always the finite field F_p for a prime p.

(a) The Finite Field Fp

The field F_p can be described as the set $\{0, 1, \ldots, p-1\}$ of non-negative integers smaller than p, with arithmetic *modulo* p.

Consider two field elements x and y in F_p. Their sum, difference, product, division, and inverse in F_p are defined in terms of regular integer arithmetic as follows.

Addition: The sum $x + y$ in F_p is:

$$(x + y) \bmod p$$

Subtraction: The difference $x - y$ in F_p is:

$$(x - y) \bmod p$$

Multiplication: The product $x \cdot y$ in F_p is:

$$(x \cdot y) \bmod p$$

Thus, addition, subtraction, and multiplication may be computed as usual and the result is then reduced modulo p.

Division: x / y in F_p (with $y \neq 0$) is defined in terms of an inverse:

$$(x \cdot y^{-1}) \bmod p$$

Inverse: The inverse of a non-zero y in F_p is the unique non-zero field element z (i.e. integer in $\{1, 2, \ldots, p-1\}$) that satisfies:

$$(y \cdot z) \bmod p = 1$$

The value of z is best computed using the Extended Euclidean Algorithm for computation of the greatest common divisor (gcd). When computing the gcd of two integers a and b, it returns the gcd of a and b as well as two integers u and v that satisfy:

$$a \cdot u + b \cdot v = \gcd(a, b)$$

Because p is prime, $\gcd(p, y) = 1$, so when computing this gcd, the Extended Euclidean Algorithm returns u and v that satisfy:

$$p \cdot u + y \cdot v = 1$$

Therefore, $y \cdot v \bmod p = 1$, and so $y^{-1} = v \bmod p$.

Note that the algorithms defined in this specification do not use division or inverse.

(b) Elliptic Curves and Point Representation

An elliptic curve over F_p is a collection of points – pairs (x, y) of field elements in the underlying field that satisfy a given equation. For the selected curves, this *curve equation* has the form $y^2 = x^3 + ax + b$ which is computed in F_p. For all selected curves, $a = -3$.

To fully define an elliptic curve E over F_p, one must specify:

The value of p.

The curve equation (specified by the parameters $a = -3$ and b).

A base point G (also called the *generator*) on the curve.

The *order* n of G. For all supported curves, n is prime, and equals the number of points on the curve. (This implies that the so-called *cofactor* equals 1.)

For more details, see ISO/IEC 15946-1. The standards IEEE P1363 and SEC 1 and their references provide alternative descriptions and implementation hints that may be useful.

For transmission and processing, elliptic curve points and specifically their integer coordinates will need to be interpreted as (converted to) byte strings. See section B2.6.

This specification represents points such as public keys simply as the x-coordinate of the point. Note that with this representation a value of x can actually represent two points:

$$P = (x, y) \text{ and } -P = (x, -y).$$

(c) Point Addition

Input for adding two points:

- Algorithm Suite Indicator identifying the curve.
- Points (x_1, y_1) and (x_2, y_2) on the curve.

Output:

- Point (x_3, y_3) on the curve equal to the sum $(x_1, y_1) + (x_2, y_2)$.

The points on an elliptic curve together with one special extra point **0**, called the point at infinity, form a group, where the group operation is traditionally called addition, or point addition. That is, two points on an elliptic curve P and Q can be "added", and the result, denoted as $P + Q$, is a point on the same curve.

Point addition for selected curves in this specification is defined as follows.

For any point P on the curve (including $P = \mathbf{0}$):

$$P + \mathbf{0} = \mathbf{0} + P = P$$

For any point (x, y) on the curve.[43c]

$$(x, y) + (x, -y) = \mathbf{0}$$

In other words:

$$-(x, y) = (x, -y) \text{ and } (x, 0) + (x, 0) = \mathbf{0}$$

For any two points (x_1, y_1) and (x_2, y_2) with different x-coordinates, the sum $(x_1, y_1) + (x_2, y_2) = (x_3, y_3)$ which has coordinates:

$$x_3 = \lambda^2 - x_1 - x_2 \bmod p$$

$$y_3 = \lambda(x_1 - x_3) - y_1 \bmod p$$

with:

$$\lambda = (y_2 - y_1) \cdot (x_2 - x_1)^{-1} \bmod p$$

For any point (x, y) with $y \neq 0$, the sum $(x, y) + (x, y) = (x_3, y_3)$ which has coordinates:

$$x_3 = \lambda^2 - 2x \bmod p$$

$$y_3 = \lambda(x - x_3) - y \bmod p$$

with:

$$\lambda = (3x^2 - 3) \cdot (2y)^{-1} \bmod p$$

(d) Point Multiplication – PointMultiply()

Input for multiplying a point by a positive integer:

- Algorithm Suite Indicator identifying the curve.
- Positive integer k.
- Point P on the curve.

Output:

- Point kP on the curve.

Point multiplication (also called scalar multiplication) is repeated addition of an elliptic curve point to itself: for a positive integer k, $kP = P + P + \cdots + P$ (with k copies of P). For $k = 0$, the result is the point at infinity: $0P = \mathbf{0}$.

As a result of the group structure of the set of points on the curve, there is an integer n such that $nP = \mathbf{0}$ for all P on the curve. The number n is called the order of the curve group. As a consequence, for all points P and all integers k, $kP = (k \bmod n)P$.

A point multiplication kP can be computed efficiently using techniques analogous to efficient modular exponentiation. Note that especially efficient algorithms are known if the curve point is known in advance, as this allows for the use of precomputed intermediate results. These data are often called "public key multiples". In general, any fixed-base method can be used.

[43c] By the curve equation, this implies that $(x, -y)$ is also on the curve.

(e) Point Finding – Point4x()

Input for finding a point on a prime curve with given x-coordinate:

- Algorithm Suite Indicator identifying the curve.
- Positive integer $x < p$, where p is the characteristic of the curve field.

Output:

- Point (x,y) on the curve with least value y.

In order to find a point P on the curve that has a given x-coordinate x, it is necessary to find an integer y such that $y^2 = x^3 + ax + b \bmod p$. Note that if there is a non-zero solution y to this equation, and thus a point (x, y) on the curve, then there will be exactly two non-zero solutions y and $-y$, and thus two points P and $-P$ on the curve with this x-coordinate x. The routine Point4x() assumes that $p = 3 \bmod 4$ (which is the case for all the curves defined in this specification) and returns a point (x, y) where y can be computed as follows:

$$y' = \text{modsqrt}(x^3 + ax + b, p) = (x^3 + ax + b)^{((p+1)/4)} \bmod p$$

$$y = \min(y', p-y')$$

Under some attack conditions this function may return a point that is not on the curve. Therefore, card implementations need to take appropriate steps to ensure that the point returned is valid. Such attacks are less relevant to the Diffie-Hellman and the certificate verification functions in the Kernel.

Note: When the recovered point is to be used for Diffie-Hellman key agreement rather than signature verification either of the two possible y values may be used and therefore the $y = \min(y', p-y')$ step may be omitted and y' can be used directly.

B2.2.2 Selected Elliptic Curves

This section defines the selected curves and their representation. The curves are selected from ISO/IEC 15946-5. Other curves may be used for domestic use (see Table 28e) but the details are outside the scope of this specification.

The selected curves are listed in the following Table 28a.

Curve	N_{FIELD} (bytes)	Public Key Length (bytes) (= N_{FIELD})	Curve Type
P-256	32	32	Pseudo-random over F_p
P-521	66	66	Pseudo-random over F_p

Table 28a: Selected Elliptic Curves

An elliptic curve Public Key is a point on the curve and in this specification a Public Key is represented by the x-coordinate of the point.

In the choice of curves one should take into account the lifetime of the keys compared to the expected progress in ECC cryptanalysis. The minimum and maximum strengths of curves mandated by each of the payment systems are specified in their corresponding proprietary specifications.

Below, each of the supported curves is specified in detail.

(a) Curve P-256

The curve P-256 is defined over F_p for a 256-bit prime p, with curve equation $y^2 = x^3 - 3x + b$ (i.e. $a = -3$). It has the following parameters. The prime p and (integer) order n are given in both decimal and hexadecimal form; field elements (the parameter b and the base point coordinates) are given in hexadecimal only.

Parameter	Value
$p =$	$2^{256} - 2^{224} + 2^{192} + 2^{96} - 1 =$ 115 79208 92103 56248 76269 74469 49407 57353 00861 43415 29031 41955 33631 30886 70978 53951 FFFFFFFF 00000001 00000000 00000000 00000000 FFFFFFFF FFFFFFFF FFFFFFFF
$n =$	115 79208 92103 56248 76269 74469 49407 57352 99969 55224 13576 03424 22259 06106 85120 44369 FFFFFFFF 00000000 FFFFFFFF FFFFFFFF BCE6FAAD A7179E84 F3B9CAC2 FC632551
$b =$	5AC635D8 AA3A93E7 B3EBBD55 769886BC 651D06B0 CC53B0F6 3BCE3C3E 27D2604B
$G_x =$	6B17D1F2 E12C4247 F8BCE6E5 63A440F2 77037D81 2DEB33A0 F4A13945 D898C296
$G_y =$	4FE342E2 FE1A7F9B 8EE7EB4A 7C0F9E16 2BCE3357 6B315ECE CBB64068 37BF51F5

Table 28b: Parameters of Curve P-256

(b) Curve P-521

The curve P-521 is defined over F_p where p is the 521-bit Mersenne prime, with curve equation $y^2 = x^3 - 3x + b$ (i.e. $a = -3$). It has the following parameters. The prime p and (integer) order n are given in both decimal and hexadecimal form; field elements (the parameter b and the base point coordinates) are given in hexadecimal only.

Parameter	Value
$p =$	$2^{521} - 1 =$ 68 64797 66013 06097 14981 90079 90813 93217 26943 53001 43305 40939 44634 59185 54318 33976 56052 12255 96406 61454 55497 72963 11391 48085 80371 21987 99971 66438 12574 02829 11150 57151 01FF FFFFFFFF FFFFFFFF FFFFFFFF FFFFFFFF FFFFFFFF FFFFFFFF FFFFFFFF FFFFFFFF FFFFFFFF FFFFFFFF FFFFFFFF FFFFFFFF FFFFFFFF FFFFFFFF FFFFFFFF FFFFFFFF
$n =$	68 64797 66013 06097 14981 90079 90813 93217 26943 53001 43305 40939 44634 59185 54318 33976 55394 24505 77463 33217 19753 29639 96371 36332 11138 64768 61244 03803 40372 80889 27070 05449 01FF FFFFFFFF FFFFFFFF FFFFFFFF FFFFFFFF FFFFFFFF FFFFFFFF FFFFFFFF FFFFFFFA 51868783 BF2F966B 7FCC0148 F709A5D0 3BB5C9B8 899C47AE BB6FB71E 91386409
$b =$	051 953EB961 8E1C9A1F 929A21A0 B68540EE A2DA725B 99B315F3 B8B48991 8EF109E1 56193951 EC7E937B 1652C0BD 3BB1BF07 3573DF88 3D2C34F1 EF451FD4 6B503F00
$G_x =$	C6 858E06B7 0404E9CD 9E3ECB66 2395B442 9C648139 053FB521 F828AF60 6B4D3DBA A14B5E77 EFE75928 FE1DC127 A2FFA8DE 3348B3C1 856A429B F97E7E31 C2E5BD66
$G_y =$	118 39296A78 9A3BC004 5C8A5FB4 2C7D1BD9 98F54449 579B4468 17AFBD17 273E662C 97EE7299 5EF42640 C550B901 3FAD0761 353C7086 A272C240 88BE9476 9FD16650

Table 28c: Parameters of Curve P-521

B2.2.3 Required Support for Elliptic Curves

EMV kernels supporting XDA or ECC ODE shall support ECC P-256 and P-521. Support of other curves is outside the scope of EMV although Algorithm Suite Indicators may be allocated (see section B2.4). P-521 is supported for contingency purposes only.

B2.2.4 Key Generation

Input for key generation:

- Algorithm Suite Indicator identifying the curve.

Output:

- An integer d and a point $P = dG$ on the curve.

This section describes the generation of ECC key pairs with special attention to long term Payment System, Issuer and ICC key pairs. The method of elliptic curve key pair generation follows ISO/IEC 15946-1.

The private key d for a given elliptic curve E with base point G of order n consists of a positive integer d, satisfying $1 < d < n-1$.

The corresponding public key point is $P = dG$ on the curve E.

Given an elliptic curve, all parameters including the base point are fixed. Therefore, key generation consists solely of random or pseudorandom generation of d and subsequent computation of the public key point $P = dG$ and thereby its x-coordinate representation.

This specification represents public keys using only their x-coordinate. For any non-zero point (x,y) on the curve, $(x,-y)$ will also be on the curve. The function in this specification that is sensitive to the value of the y-coordinate is signature verification and specifically Issuer Public Key Certificate verification using the Certification Authority Public Key, ICC Public Key Certificate verification using the Issuer Public Key, and ICC XDA signature verification using the ICC Public Key. For this reason the Payment System key, the Issuer key and the ICC key must be generated to ensure that the verifier can determine the correct value of the y-coordinate of the Public Key from knowledge of only the x-coordinate.

For this reason this specification requires that Payment System, Issuer and ICC keys are generated to ensure that the y-coordinate as an integer modulo p is less than $(p+1)/2$. The function Point4x() defined in B2.2.1(e) will then generate the correct y-coordinate given the correct x-coordinate.

Two examples of how to generate such Payment System, Issuer and ICC keys are provided below. The first approach 1) will take longer, but the second approach 2) may require special HSM functionality.

1. Repeatedly generate a random non-negative integer d satisfying $1 < d < n-1$ – for example, d := 2+RandomInteger(n-3) – until the y-coordinate of dG is less than $(p+1)/2$.
2. Generate a random non-negative integer d' satisfying $1 < d' < n-1$ for example d' := 2+RandomInteger(n-3), and if the y-coordinate of $d'G$ is less than $(p+1)/2$ then $d := d'$ else set $d := n - d'$.

The public key is then calculated to be the point $P = dG$ and its representation is the x-coordinate of P. Note that this construction of the Payment System key, Issuer key and ICC key reduces the key space of the private key by one bit.

ECC encryption requires that the terminal generate an ephemeral key pair. As this key is used for Diffie-Hellman key agreement rather than signature, the y-coordinate does not require the checks stipulated above. Thus, the terminal ephemeral private key d can be generated without the range checks on y:

> Generate a random non-negative integer d' satisfying $1 < d < n-1$ for example d := 2+RandomInteger(n-3).

For security reasons, a strong pseudo-random or random number generator is required (see section B2.5).

B2.3 Hash Algorithms (for ECC)

Table 28d below lists recognised hash algorithms and their corresponding algorithm indicators. This indicator is used in the ICC Public Key Certificate to identify the hash algorithm to be used when calculating the ICCD Hash.

Hash Algorithm Indicator	Hash Algorithm	N_{HASH}	Block Size
'01'	SHA-1 (not used with ECC, see B3)	n/a	n/a
'02'	SHA-256	32	64
'03'	SHA-512	64	128
'04' (RFU)	SHA-3 256	32	136
'05' (RFU)	SHA-3 512	64	172
'80'	SM3	32	64

Table 28d: Recognised Hash Algorithms

Note that Hash Algorithm Indicators specified as reserved for future use (RFU) are not currently used by this specification and therefore are not approved identifiers.

Note that the Chinese algorithm SM3 is included for domestic suites.

All these hash algorithms can hash an arbitrarily large input[43d] and implementations shall be designed to support this.

SHA-256, SHA-512, and SHA-3 are standardised in FIPS 202 and ISO/IEC 10118-3.

SM3 is standardised in ISO/IEC 10118-3.

SHA-512 and SHA-3 512 are supported for contingency purposes only.

B2.4 Cryptographic Algorithm Suites (for ECC)

This section lists the Public Key Algorithm Suites and associated Indicators used in this specification for ECC-based mechanisms.

B2.4.1 Cryptographic Algorithm Suites for ECC Signatures

Table 28e below applies to the algorithms used for verifying ECC signatures hence for verifying public key certificates and XDA signatures.

[43d] Actually SHA-256 and SM3 have a limit of 2^{64} bits and SHA-512 has a limit of 2^{128} bits.

Signature Algorithm Suite Indicator	Signature Mechanism	Hash Algorithm	ECC Curve	N_{FIELD} / N_{SIG}
'10'	EC-SDSA	SHA-256	P-256	32 / 64
'11'	EC-SDSA	SHA-512	P-521	66 / 130
'12' (RFU)	EC-SDSA	SHA-3 256	P-256	32 / 64
'13' (RFU)	EC-SDSA	SHA-3 512	P-521	66 / 130
'80'	SM2-DSA	SM3	SM2-P256	32 / 64

Table 28e: Cryptographic Algorithm Suite Indicators for ECC Signatures

The Signature Mechanism EC-SDSA is described in annex A2.2.

Note that Signature Algorithm Suite Indicators specified as reserved for future use (RFU) are not currently used by this specification and therefore are not approved identifiers.

Note that the Chinese algorithm SM2-DSA is included for domestic suites.

For details of the hash algorithms recognised for use with the Signature Algorithm Suites see section B2.3.

P-521 is supported for contingency purposes only.

B2.4.2 Cryptographic Algorithm Suites for ECC Encryption

Table 28f below applies to the algorithms used for ECC encryption.

ODE Algorithm Suite Indicator	ECC Curve	N_{FIELD}	Key Agreement and Derivation	Authenticated Encryption	Block Cipher
'00'	P-256	32	DH 16 byte keys	EtM	AES
'01'	P-521	66	DH 16 byte keys	EtM	AES
'88'	SM2-P256	32	DH 16 byte keys	EtM	SM4

Table 28f: Cryptographic Algorithm Suite Indicators for Offline Data Encryption (e.g. for PIN or biometrics)

The encryption methods indicated above are described in annex A2.3.

Note that the Chinese curve SM2-P256 and block cipher SM4 are included for domestic suites.

Annex A2.3 specifies

- Diffie-Hellman (DH) key agreement and key derivation
- Encrypt-then-MAC (EtM) authenticated encryption and uses Counter Mode encryption and CMAC with Additional Authenticated Data

AES and SM4 are standardised in ISO/IEC 18033-3.

P-521 is supported for contingency purposes only.

B2.5 Random Number Generation (for ECC)

This section defines functions and requirements for generating 'random numbers' especially for ECC key generation in section B2.2.4.

The function RandomInteger() takes as input a bounding integer m and returns a random non-negative integer less than m, i.e. in the range [0,…,m-1]. One way to achieve this is as follows:

1. M := len(m), the bit-length of m
2. d := BS2I(RandomBits(M))
3. If ($d > m - 1$), then go to step 2
4. Return d.

The function RandomBits() takes an integer M as parameter and returns a string $X = <X_{M-1}, \ldots, X_0>$ of M random bits. Techniques and requirements for generating random bit strings of a given length by both deterministic and non-deterministic methods are provided in ISO/IEC 18031.

ISO/IEC 18031 classifies random bit generators as either non-deterministic (where the primary entropy source can, over time, provide unlimited entropy) or deterministic (where the primary entropy source has limited entropy such as a seed value).

Note: An algorithm for generating the Terminal Unpredictable Number (tag 9F37) is included in section 11 in accordance with Specification Bulletin 144, June 2014.

B2.6 Integer Conversion Functions (for ECC)

There are mathematical functions used in this specification (e.g. signature generation) that involve the processing of integers which are represented as bits and bytes. This section therefore specifies how strings of bits and bytes are to be interpreted as integers.

Note that because the elliptic curves identified in this specification operate over finite fields with a prime number p of elements, each element of the field is naturally interpreted as a non-negative integer less than p. and points are then represented as pairs of such integers.

B2.6.1 Integers and Bits and Bytes

For the purposes of this section the conversion between integers and bytes is mediated by conversion to strings of bits. The conversion functions defined in this section are consistent with those used in the standards of ISO/IEC JTC1 SC27 (see for example ISO/IEC 14888-3 Annex B) and use the term Octet instead of Byte and thereby distinguish OS (Octet String) from BS (Bit String).

BS2I() and I2BS()

Input for BS2I():

- A bit string x of length $l > 0$.

Output:

- A non-negative integer value v.

The function BS2I(x) maps a bit string x of length l to a non-negative integer value v, as follows. If $x = <x_{l-1}, \ldots, x_0>$ where $x_0,\ldots, x_{l-1}$ are bits, then the value v is defined as

$$v = \sum_{k=0}^{l-1} x_k 2^k$$

For the empty string, v is equal to 0.

- Example: BS2I(000 0010 1010 1100 0001) = 10945

The function BS2I() is invoked from the function RandomInteger() (section B2.5) used by Key Generation (section B2.2.4).

Input for I2BS():

- Two non-negative integers v and l.

Output:

- A bit string x of length l.

The function I2BS(v,l) which takes as input two non-negative integers v and l, and outputs the unique bit string x of length l such that BS2I(x) = v, if such an x exists. If no such x exists then the function outputs an error message.

- Example 1: I2BS(10945, 19) = 000 0010 1010 1100 0001
- Example 2: I2BS(10945, 14) = 10 1010 1100 0001

The function I2BS() is invoked from Key Derivation (section A2.3.2), Encryption and Authentication (section A2.3.3), and Authentication and Decryption (section A2.3.4).

OS2BS() and BS2OS()

Input for OS2BS():

- An octet string x.

Output:

- A bit string y.

The function OS2BS(x) takes as input an octet string x, interprets it as a bit string y and outputs the bit string y.

- Example: OS2BS('00' '2A' 'C1') = 0000 0000 0010 1010 1100 0001

Input for BS2OS():

- A bit string y.

Output:

- An octet string x.

The function BS2OS(y) takes as input a bit string y, whose length is a multiple of 8, and outputs the unique octet string x such that y = OS2BS(x).

- Example: BS2OS(0000 0010 1010 1100 0001)
  = <0000 0000> <0010 1010> <1100 0001>
  = '00' '2A' 'C1'

OS2I() and I2OS()

Input for OS2I():

- An octet string x.

Output:

- An integer v.

The function OS2I(x) takes as input an octet string x and outputs the integer v = BS2I(OS2BS(x)).

- Example: OS2I(<0010 1010> <1100 0001>) = OS2I('2A' 'C1') = 10945

The function OS2I() is invoked from PIN Encipherment (section 13.2), Biometric Data Encipherment (section 13.4), Signature Generation (section A2.2.2), and Signature Verification (section A2.2.3).

Input for I2OS():

- two non-negative integers v and l.

Output:

- An octet string x of length l in octets.

The function I2OS(v,l) takes as input two non-negative integers v and l, and outputs the unique octet string x of length l in octets such that OS2I(x) = v, if such an x exists. Otherwise, the function outputs an error message.

- Example 1: I2OS(10945, 3) = <0000 0000> <0010 1010> <1100 0001>

  = '00' '2A' 'C1'

- Example 2: I2OS(10945, 2) = <0010 1010> <1100 0001>

  = '2A' 'C1'

The function I2OS() is invoked from Signature Generation (section A2.2.2), Signature Verification (section A2.2.3) and Encryption and Authentication (section A2.3.3).

Annex B3 is updated as below:

B3 Hashing Algorithms (for RSA)

This algorithm is standardised in ISO/IEC 10118-3 and is used by the RSA-based mechanism specified in Annex A2.1. SHA-1 takes as input messages of arbitrary length and produces a 20-byte hash value.

Annex D1 is updated as below:

D1 Issuer and ICC RSA Public Key Length Considerations

Annex D1 applies only to RSA-based mechanisms.

<Changes to Book 3>

In section 3, Definitions, delete the following terms:

~~**Accelerated Revocation**~~	~~A key revocation performed on a date sooner than the published key expiry date.~~
~~**Key Expiry Date**~~	~~The date after which a signature made with a particular key is no longer valid. Issuer certificates signed by the key must expire on or before this date. Keys may be removed from terminals after this date has passed.~~
~~**Key Life Cycle**~~	~~All phases of key management, from planning and generation, through revocation, destruction, and archiving.~~
~~**Key Replacement**~~	~~The simultaneous revocation of a key and introduction of a key to replace the revoked one.~~
~~**Key Revocation**~~	~~The key management process of withdrawing a key from service and dealing with the legacy of its use. Key revocation can be as scheduled or accelerated.~~
~~**Key Revocation Date**~~	~~The date after which no legitimate cards still in use should contain certificates signed by this key, and therefore the date after which this key can be deleted from terminals. For a planned revocation the Key Revocation Date is the same as the key expiry date.~~
~~**Planned Revocation**~~	~~A key revocation performed as scheduled by the published key expiry date.~~

In section 3, Definitions, these terms are incorporated in alphabetical order:

Elliptic Curve Cryptography	Public key cryptography based on the algebraic structure of elliptic curves over finite fields.
Extended Data Authentication	A form of offline dynamic data authentication.

In section 4.1, Abbreviations, the following Abbreviations are incorporated in alphabetical order:

ECC	Elliptic Curve Cryptography
N_{FIELD}	Length of a finite field element

N_{HASH}	Output length of a hash function
N_{SIG}	Length of an ECC Digital Signature
ODA	Offline Data Authentication
ODE	Offline Data Encipherment
SDAD	Signed Dynamic Application Data
TAA	Terminal Action Analysis
XDA	Extended Data Authentication

In Section 5.4, the first paragraph and the first bullet are hereby deleted and replaced with the following:

In several instances, the terminal is asked to build a flexible list of data elements to be passed to the card under the card's direction. To minimise processing within the ICC, such a list is not TLV encoded but is a single constructed field built by concatenating several data elements together. Since the elements of the constructed field are not TLV encoded, it is imperative that the ICC knows the format of this field when the data is received. This is achieved by including a Data Object List (DOL) in the ICC, specifying the format of the data to be included in the constructed field. DOLs currently used in this specification include:

- the Processing Options Data Object List (PDOL) used with the GET PROCESSING OPTIONS command
  Note: An ICC supporting XDA or ECC ODE may use PDOL to obtain Terminal Capabilities (tag '9F33') to determine whether the terminal supports XDA.

In Section 6.5.5.1, the second paragraph is hereby deleted and replaced with the following:

This command is also used when performing the Combined DDA/Application Cryptogram Generation (CDA) function as described in Book 2 section 6.6 and when performing the XDA offline data authentication function as described in Book 2 section 12.

In Section 6.5.5.2, replace Table 12 as below:

b8	b7	b6	b5	b4	b3	b2	b1	Meaning
0	0							AAC
0	1							TC
1	0							ARQC
1	1							RFU
		x						RFU
			0	0				No CDA/XDA signature ~~not~~ requested
			1	0				CDA signature requested
			0	1				XDA signature requested
			1	1				RFU
				~~x~~	x	x	x	RFU

Table 12: GENERATE AC Reference Control Parameter

In Section 6.5.5.4, the second and third paragraphs of Format 2 are hereby deleted in and replaced with the following:

Format 2 shall be used if the response is being returned with~~in~~ a signature as specified for the CDA or XDA functions described in sections 6.6 and 12 of Book 2. The ~~required~~ data elements for the response are shown below in Table 13a. These data elements may be in any order~~in the appropriate tables in that section~~.

Value	Presence
Cryptogram Information Data (CID)	M
Application Transaction Counter (ATC)	M
Application Cryptogram (AC)	C1
Issuer Application Data (IAD)	O
Other data objects	O
Signed Dynamic Application Data (SDAD)	C2

Table 13a: Format 2 GENERATE AC Response Message Data Field

Note 1: Application Cryptogram (AC) is not present when CDA signature is returned.

Note 2: When the card does not perform XDA or CDA, Signed Dynamic Application Data is not included in the response.

For both ~~formats~~ Format 1 and Format 2, the Cryptogram Information Data returned by the GENERATE AC response message is coded as shown in Table 14.

In Section 6.5.8.3, the following language is added:

Note: An ICC supporting XDA or ECC ODE may return PDOL indicating Terminal Capabilities (tag '9F33') in response to SELECT command.

In Section 6.5.12.1, the third paragraph is updated as follows:

The biometric data to be sent in the VERIFY command shall be enciphered for confidentiality, as described in Book 2 section 7.3 (for RSA) or section 13.4 (for ECC).

In Section 6.5.12.2, Table 23 is updated as:

b8	b7	b6	b5	b4	b3	b2	b1	Meaning
0	0	0	0	0	0	0	0	As defined in ISO/IEC 7816-4 [3]
1	0	0	0	0	0	0	0	Plaintext PIN, format as defined below
1	0	0	0	0	x	x	x	RFU for this specification
1	0	0	0	1	0	0	0	Enciphered PIN (RSA), format as defined in Book 2 section 7
1	0	0	0	1	0	0	1	Enciphered Biometric (RSA), format as defined in Book 2 section 7
1	0	0	0	1	0	1	0	Enciphered PIN (ECC), format as defined in Book 2 section 13
1	0	0	0	1	0	~~x~~1	~~x~~1	Enciphered Biometric (ECC), format as defined in Book 2 section 13~~RFU for this specification~~
1	0	0	0	1	1	x	x	RFU for the individual payment systems
1	0	0	1	x	x	x	x	RFU for the issuer

Table 23: VERIFY Command qualifier of reference data (P2)

In section 7.2, the paragraph after Table 26 is updated as follows:

Table 27 lists the data objects that must be present if the ICC supports ~~offline static data authentication (~~SDA~~)~~. Table 28 lists the data objects that must be present if the ICC supports ~~offline static data authentication (~~DDA and/or CDA~~)~~.[5] Table 28a lists the data objects that must be present if the ICC supports XDA. Offline data authentication is required to support offline transactions but is optional in cards that support only online transactions.

In Section 7.2, Table 28a is added after Table 28 as:

Tag	Value
'8F'	Certification Authority Public Key Index
'90'	Issuer Public Key Certificate
'9F46'	ICC Public Key Certificate

Table 28a: Data Required for XDA

In Section 7.5, insert the following note after the third paragraph:

Note: If data is missing during CVM processing, the CVM method fails. The 'ICC data missing' TVR bit is not set.

In Section 7.5, Table 31 is updated as:

Name	Tag	'ICC Data Missing' Shall Be Set If ...
Application Transaction Counter (ATC)	'9F36'	ATC is not returned by GET DATA command and both Lower and Upper Consecutive Offline Limit data objects are present
Cardholder Verification Method (CVM) List	'8E'	Not present and AIP indicates that cardholder verification is supported
Certification Authority Public Key Index	'8F'	Not present and ~~AIP indicates any form of offline data authentication is supported (~~SDA, DDA, CDA~~)~~ or XDA selected
Issuer Public Key Certificate	'90'	Not present and SDA, DDA, CDA or XDA selected~~Not present and AIP indicates any form of offline data authentication is supported (SDA, DDA or CDA)~~
Issuer Public Key Exponent	'9F32'	Not present and ~~AIP indicates any form of offline data authentication is supported (~~SDA, DDA or CDA~~)~~ selected.

Name	Tag	'ICC Data Missing' Shall Be Set If ...
Issuer Public Key Remainder	'92'	Not present and SDA, DDA or CDA selected and the 'length of the Issuer Public Key', as recovered from the Issuer Public Key Certificate, indicates that there was insufficient space for the entire Issuer Public Key in the certificate.~~Not present and AIP indicates any form of offline data authentication is supported (SDA, DDA or CDA) and the 'length of the Issuer Public Key', as recovered from the Issuer Public Key Certificate, indicates that there was insufficient space for the entire Issuer's Public Key in the certificate~~
Last Online Application Transaction Counter (ATC) Register	'9F13'	Last Online ATC Register is not returned by GET DATA command and both Lower and Upper Consecutive Offline Limits are present
Signed Static Application Data	'93'	Not present and ~~AIP indicates~~ SDA ~~supported~~ selected
ICC Public Key Certificate	'9F46'	Not present and DDA, CDA or XDA selected~~Not present and AIP indicates DDA or CDA supported~~
ICC Public Key Exponent	'9F47'	Not present and ~~AIP indicates~~ DDA or CDA ~~is supported~~ selected
ICC Public Key Remainder	'9F48'	Not present and ~~AIP indicates~~ DDA or CDA ~~is supported~~ selected and the 'length of the ICC Public Key', as recovered from the ICC Public Key Certificate, indicates that there was insufficient space for the entire ICC's~~s~~ Public Key in the certificate

Table 31: ICC Data Missing Indicator Setting

In section 9, the second paragraph is updated as:

The complete transaction flow is not shown in this chart, only the GENERATE AC commands, responses, and associated decisions. Furthermore, this chart applies to no ODA performed, SDA performed, and DDA performed. For CDA performed and XDA performed, please refer to Book 2 sections 6.6.3 and 12.5.4.

In section 9, the caption of Figure 7 is updated as:

Figure 7: Use of GENERATE AC Options (no ODA, SDA, DDA)

Section 9.3 is updated as:

Either one or two GENERATE AC commands are issued during the processing of a transaction according to this specification.

The ICC shall respond to the first GENERATE AC command with any of the following:

- TC
- ARQC
- AAC

The ICC shall respond to a second GENERATE AC command with either a TC or an AAC.

The possible responses listed above are in hierarchical order, with a TC being the highest and an AAC being the lowest. The terminal may request a TC, an ARQC, or an AAC. If the ICC responds with a cryptogram at a higher level or with a cryptogram of an undefined type, this indicates a logic error in the ICC. If this occurs after the first GENERATE AC command in a transaction, the transaction shall be terminated. If it occurs after the second GENERATE AC command, all processing for the transaction has been completed, and the cryptogram returned shall be treated as an AAC.

If the ICC response is an approval (TC) or online authorisation request (ARQC) and the reference control parameter of ~~the 'CDA signature requested' bit in~~ the GENERATE AC command is 'CDA signature requested' (b5-b4 = '10')~~1~~, the ICC shall return a public key signature in the GENERATE AC response ~~in a public key signature~~ as specified in Book 2 section 6.6 (CDA).

If the reference control parameter of the GENERATE AC command is 'XDA signature requested' (b5-b4 = '01'), the ICC shall return a public key signature in the GENERATE AC response as specified in Book 2 section 12 (XDA).

In section 9.3.1, a new footnote number 8 is added to the second paragraph as:

If the ICC responds with a TC[8] or an AAC, the terminal completes the transaction offline.

[8] If ECC key recovery failed or XDA signature verification failed and depending on the TAC-Denial and/or IAC-Denial, the terminal attempts to go online after the Terminal Action Analysis with the TC returned by the ICC.

In section 10.1 'Description', a note is added after the fourth paragraph as:

Note: An ICC supporting XDA or ECC ODE may request Terminal Capabilities (tag '9F33') using PDOL.

In section 10.3, ‘Conditions of Execution’, ‘Sequence of Execution’ and ‘Description’ are updated as:

Conditions of Execution:

Availability of data in the ICC to support offline data authentication is optional; its presence is indicated in the Application Interchange Profile (AIP). If the terminal and the ICC support a common method of offline data authentication, the terminal shall select this method and perform the offline data authentication. ~~Depending on the capabilities of the card and the terminal, SDA or DDA or CDA is performed.~~ The terminal attempts to perform the highest priority offline data authentication method supported by both the card and terminal according to the following rules:

If both of the following are true, the terminal shall select the XDA method, set the TVR bit for 'XDA selected' to the value 1 and shall perform XDA as specified in this section, in Book 2 section 12, and in Book 4 section 6.3.2:

- The AIP indicates that the card supports XDA.
- Terminal Capabilities for the selected AID indicates that the terminal supports XDA.

Otherwise if both of the following are true then CDA is selected, the terminal shall select the CDA method and perform CDA as specified in this section, in Book 2 section 6, and in Book 4 section 6.3.2:

- The AIP indicates that the card supports CDA.
- The terminal supports CDA.

Otherwise if both of the following are true then DDA is selected, the terminal shall select the DDA method and perform DDA as specified in this section, in Book 2 section 6, and in Book 4 section 6.3.2:

- The AIP indicates that the card supports DDA.
- The terminal supports DDA.

Otherwise if both of the following are true, the terminal shall select the SDA method, set the TVR bit for 'SDA selected' to the value 1 and perform SDA as specified in this ~~specification and shall set the 'SDA selected' bit in the TVR to 1~~ section, in Book 2 section 5, and in Book 4 section 6.3.2:

- The AIP indicates that the card supports SDA.
- The terminal supports SDA.

Otherwise (i.e. if neither XDA nor CDA nor DDA nor SDA is to be performed), the terminal shall set the 'Offline data authentication was not performed' bit in the TVR to 1.

Note: An ICC supporting both RSA based methods (SDA, DDA, CDA or RSA ODE) and ECC based methods (XDA or ECC ODE) is expected to return a PDOL indicating Terminal Capabilities (tag '9F33') in response to SELECT command. Based on the Terminal Capabilities, the ICC selects either RSA based methods or ECC based methods. Records referenced in the AFL contain either data objects needed for RSA based methods or data objects needed for ECC based methods, as selected by the ICC, but not both. In particular, if RSA methods are selected by the ICC, the certificates (tags '90' and '9F46') contained in the records referenced in the AFL are expected to be formatted as described in Book 2 Table 13 and Table 14, while if ECC methods are selected by the ICC, they are expected to be formatted as described in Book 2 Table 27d and Table 27e.

Sequence of Execution:

For SDA and DDA the terminal shall perform offline data authentication in any order after Read Application Data but before completion of the terminal action analysis.

For CDA and XDA the terminal shall start offline data authentication at any time after Read Application Data, but CDA and XDA cannot be successfully completed until after the response to the GENERATE AC command. The key recovery/authentication process may be conducted at any time during this period, but the terminal shall check the presence of the payment system CA Public Key prior to Terminal Action Analysis.

For XDA the terminal shall complete the ECC key recovery and XDA signature verification processes described in Book 2 section 12 before any online authorisation request and before the final Terminal Action Analysis prior to any second GENERATE AC.

~~**Note:** Although the terminal shall commence performing CDA before completion of Terminal Action Analysis, the terminal will not normally finish performing CDA until after it has received the response to the GENERATE AC command. (This is a necessary consequence of the design of CDA.)~~

Description:

SDA authenticates static data put into the card by the issuer. DDA, CDA and XDA authenticate ICC-resident data, data from the terminal, and the card itself. CDA and XDA also authenticate that certain data passed between the terminal and card has not been altered by an intermediate device.

For SDA, DDA, and CDA, input to the authentication process is formed from the records identified by the AFL, followed by the data elements identified by the optional Static Data Authentication Tag List (tag '9F4A').

For XDA, input to the authentication process is formed from the records identified by the AFL, followed by the tag, length and value of the AIP, the tag, length and value of Application Identifier (AID) – terminal, and, if a PDOL was received from the card, the tag, length and value of the PDOL. This forms the Issuer Certified Card Data (which is the Static Data to be Authenticated for XDA).

Only those records identified in the AFL as participating in offline data authentication are to be processed. Records are processed in the same sequence in which they appear within AFL entries. The records identified by a single AFL entry are to be processed in record number sequence. The first record begins the input for the authentication process, and each succeeding record is concatenated at the end of the previous record.

The records read for offline data authentication shall be TLV-coded with tag equal to '70'.

The data from each record to be included in the offline data authentication input depends upon the SFI of the file from which the record was read.

- For files with SFI in the range 1 to 10, the record tag ('70') and the record length are excluded from the offline data authentication process. All other data in the data field of the response to the READ RECORD command (excluding SW1 SW2) is included.
- For files with SFI in the range 11 to 30, the record tag ('70') and the record length are not excluded from the offline data authentication process. Thus all data in the data field of the response to the READ RECORD command (excluding SW1 SW2) is included.

If the records read for offline data authentication are not TLV-coded with tag equal to '70' then offline data authentication shall be considered to have been performed and to have failed; that is, the terminal shall set the 'Offline data authentication was performed' bit in the TSI to 1, and shall set the appropriate 'SDA failed' or 'DDA failed' or 'CDA failed' or 'ECC key recovery failed'[11] bit in the TVR.

The bytes of the record are included in the concatenation in the order in which they appear in the command response.

After all records identified by the AFL have been processed, the terminal processes the following:

- For SDA, DDA, and CDA, the Static Data Authentication Tag List is processed, if it exists. If the Static Data Authentication Tag List exists, it shall contain only the tag for the Application Interchange Profile. The tag must represent the AIP available in the current application. The value field of the AIP is to be concatenated to the current end of the input string. The tag and length of the AIP are not included in the concatenation.
- For XDA, the tag, length and value field of the AIP as received from the card are concatenated to the end of the concatenated records (even if the SDA Tag List exists or not), followed by the tag, length and value of Application Identifier (AID) – terminal. If a PDOL was received, the tag, length and the value of the PDOL as received from the card are appended to the end of the string. If no PDOL was received then nothing is appended.

Building of the input list for offline data authentication is considered the first step in the offline data authentication process. If the input cannot be built because of a violation of one of the above rules but offline data authentication should be performed according to the 'Conditions of Execution' above, offline data authentication shall be considered to have been performed and to have failed; that is, the terminal shall set the 'Offline data authentication was performed' bit in the TSI to 1 and shall set the appropriate 'SDA failed' or 'DDA failed' or 'CDA failed' or 'ECC key recovery failed'[12] bit in the TVR.

See Book 2 for additional steps to be performed for offline data authentication.

[11] This bit is not set until after the first GENERATE AC command and is not set if CA ECC key is missing. See section 6.3.2.2.2 of Book 4.
[12] This bit is not set until after the first GENERATE AC command and is not set if CA ECC key is missing. See section 6.3.2.2.2 of Book 4.

If the CA ECC Public Key referred by the CA Public Key Index is not present in the terminal or the index is missing, the bit 'CA ECC key missing' in the TVR must be set before the final TAA prior to the first GENERATE AC command.

If XDA is selected but the terminal fails to authenticate and recover the Issuer Public Key or the ICC Public Key (including verification of the Issuer Certified Card Data) and the CA ECC key is not missing, the bit 'ECC key recovery failed' in the TVR shall be set but only after issuing the first GENERATE AC command and before the TAA that is performed after processing the first GENERATE AC response. (Refer to the Note in Book 4 section 6.3.2.2.2 for further information regarding this).

If SDA is performed but is unsuccessful, the 'SDA failed' bit in the TVR shall be set to 1; otherwise it shall be set to 0.

If DDA is performed but is unsuccessful, the 'DDA failed' bit in the TVR shall be set to 1; otherwise it shall be set to 0.

If CDA is performed but is unsuccessful, the 'CDA failed' bit in the TVR shall be set to 1; otherwise it shall be set to 0.

If XDA is selected, the 'XDA selected' bit in the TVR shall be set to 1 before the final TAA prior to the first GENERATE AC command.

If XDA is selected but is unsuccessful, then one (and only one) of the 'CA ECC key missing', 'ECC key recovery failed' or 'XDA signature verification failed' bits in the TVR shall be set to 1; otherwise they shall be set to 0.

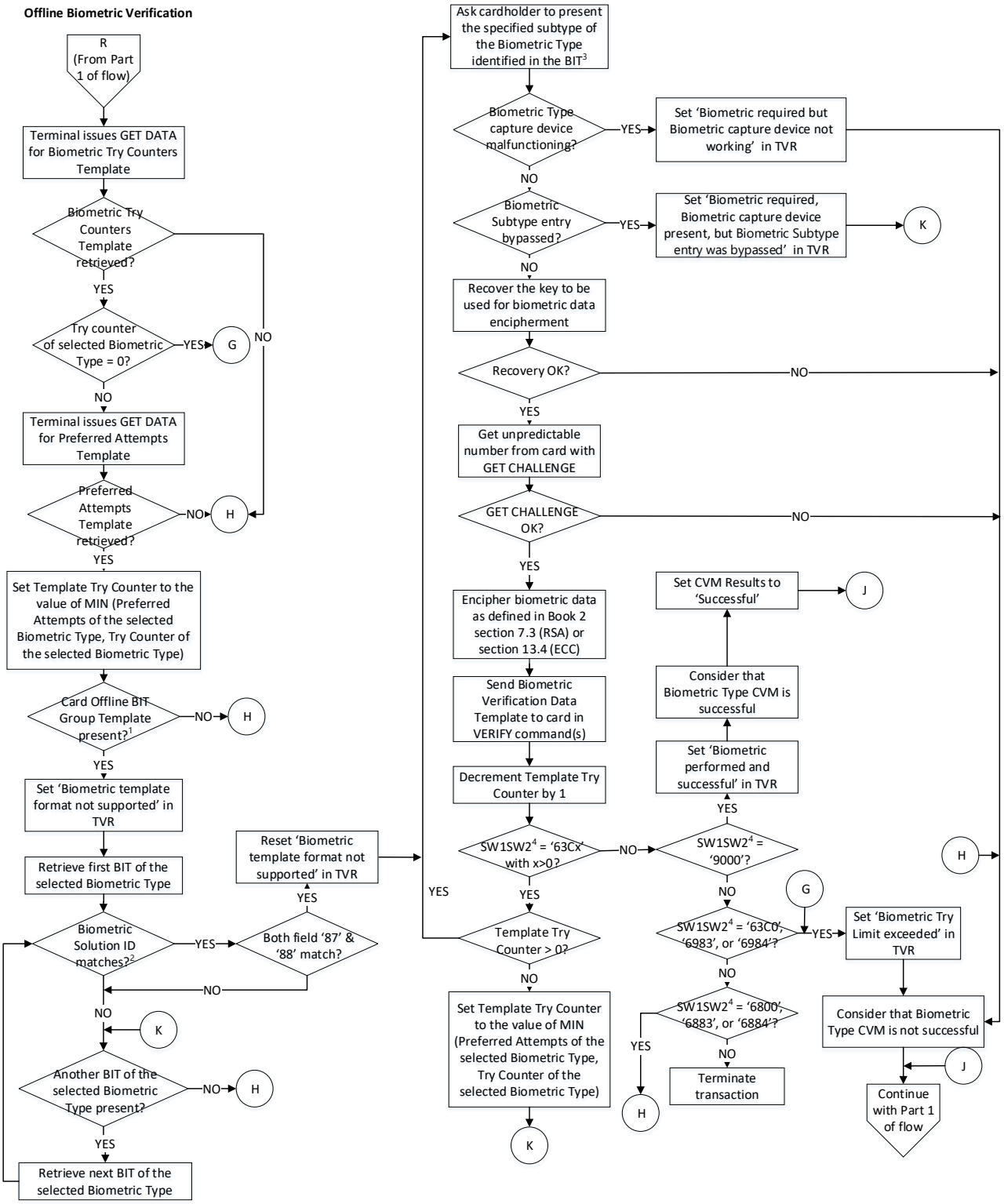
If XDA is selected then ECC key recovery shall have ended and XDA signature processing shall have ended before the TAA that is performed after processing the first GENERATE AC response. Note that ECC key recovery may have ended before or after the first GENERATE AC either by failing or succeeding, or by aborting due to CA ECC key missing. The bit 'XDA signature verification failed' in the TVR shall be set only if CA ECC key retrieval and ECC key recovery were successful but XDA signature verification failed.

Upon completion of the offline data authentication function, the terminal shall set the 'Offline data authentication was performed' bit in the TSI to 1.

In section 10.5, following note is added at the end of this section:

Note: If an ICC supports a CVM Code using either RSA ODE or ECC ODE, but does not support both the RSA and the ECC mode for this CVM Code, it is expected to return a PDOL indicating Terminal Capabilities (tag '9F33') in response to SELECT command. Based on the Terminal Capabilities, the ICC selects either RSA based methods or ECC based methods. Records referenced in the AFL contain a CVM List that reflects the ICC capabilities for the transaction. For instance, if the ICC supports Enciphered PIN verification performed by ICC with ECC but not with RSA, the CVM List in the records is expected to contain the Enciphered PIN verification performed by ICC CVM Code only if the ICC has selected ECC based methods for the transaction.

In section 10.5.5, replace Figure 12a – CVM Processing (Part 6 of 7) with the following:



In section 10.7 ‘Description’, the sixth paragraph is updated as:

The existence of each of the Terminal Action Codes is optional. In the absence of any Terminal Action Code, a default value consisting of all bits set to 0 is to be used in its place. However, it is strongly recommended that as a minimum, the Terminal Action Code - Online and Terminal Action Code - Default should be included with the bits corresponding to 'Offline data authentication was not performed', ~~and either~~ 'SDA failed', ~~or~~ 'DDA failed', ~~or~~ 'CDA failed', 'XDA signature verification failed', 'ECC key recovery failed' and 'CA ECC key missing' set to 1.[16]

In section 10.7 ‘Description’, the last two paragraphs are updated as:

If CDA is to be performed (as described in section 10.3 of this book and section 6.6 of Book 2), the terminal shall set the b~~it~~5-b4 for 'CDA signature requested' in the GENERATE AC command to '10'.

If XDA is to be performed (as described in section 10.3 of this book and section 12 of Book 2), the terminal shall set the b5-b4 for 'XDA signature requested' in the GENERATE AC command to '01'.

In section 10.9, ‘Conditions of Execution’ and ‘Sequence of Execution’ are updated as:

Conditions of Execution:

Online processing shall be performed if the ICC returns an ARQC (or a TC[20] when XDA has failed) in response to the first GENERATE AC command for the transaction.

Sequence of Execution:

The online processing function is performed when the terminal receives an ARQC in response to the first GENERATE AC command or when Terminal Action Analysis results in the transaction to be completed online after the first GENERATE AC command.

[20] If ECC key recovery or XDA signature verification failed and depending on the TAC-Denial and/or IAC-Denial, the terminal attempts to go online after the Terminal Action Analysis with the TC returned by the ICC.

In section 10.11, ‘Description’ is updated as:

If the terminal is to perform CDA (as described in section 10.3), the terminal shall set the 'CDA signature requested' bits in the GENERATE AC command to '10'.

If the terminal is to perform XDA (as described in section 10.3), the terminal shall set the 'XDA signature requested' bits in the GENERATE AC command to '01'.

In Annex A Table 33, following data elements are updated as:

Integrated Circuit Card (ICC) PIN Encipherment Public Key Certificate (RSA) or Integrated Circuit Card (ICC) Public Key Certificate for ODE (ECC)	ICC ~~PIN Encipherment~~ Public Key certified by the issuer for PIN or biometric encipherment (ODE). Format and length are different depending on RSA or ECC.	ICC	b	'70' or '77'	'9F2D'	N_I or $N_{FIELD} + N_{SIG} + N_{HASH} + 17$
Integrated Circuit Card (ICC) Public Key Certificate	ICC Public Key certified by the issuer. Format and length are different depending on RSA or ECC.	ICC	b	'70' or '77'	'9F46'	N_I or $N_{FIELD} + N_{SIG} + N_{HASH} + 17$
Issuer Public Key Certificate	Issuer public key certified by a certification authority. Format and length are different depending on RSA or ECC.	ICC	b	'70' or '77'	'90'	N_{CA} or $N_{FIELD} + N_{SIG} + 21$
Terminal Capabilities	Indicates the card data input, CVM, and security capabilities of the terminal for the selected AID	Terminal	b	—	'9F33'	3

In Annex A2 Table 34, following data element name is updated as:

Name	Template	Tag
ICC PIN Encipherment Public Key Certificate (RSA) or Integrated Circuit Card (ICC) Public Key Certificate for ODE (ECC)	'70' or '77'	'9F2D'

Annex C1 Table 37, AIP Byte 1 is updated as:

AIP Byte 1 (Leftmost)

b8	b7	b6	b5	b4	b3	b2	b1	Meaning
~~0~~1	x	x	x	x	x	x	x	~~RFU~~XDA supported

x	1	x	x	x	x	x	x	SDA supported
x	x	1	x	x	x	x	x	DDA supported
x	x	x	1	x	x	x	x	Cardholder verification is supported
x	x	x	x	1	x	x	x	Terminal risk management is to be performed
x	x	x	x	x	1	x	x	Issuer authentication is supported[1]
x	x	x	x	x	x	0	x	Reserved for use by the EMV Contactless Specifications
x	x	x	x	x	x	x	1	CDA supported

Annex C5 Table 42, TVR Bytes 1, 4 and 5 are updated as:

TVR Byte 1: (Leftmost)

b8	b7	b6	b5	b4	b3	b2	b1	Meaning
1	x	x	x	x	x	x	x	Offline data authentication was not performed
x	1	x	x	x	x	x	x	SDA failed
x	x	1	x	x	x	x	x	ICC data missing
x	x	x	1	x	x	x	x	Card appears on terminal exception file [25]
x	x	x	x	1	x	x	x	DDA failed
x	x	x	x	x	1	x	x	CDA failed
x	x	x	x	x	x	1	x	SDA selected
x	x	x	x	x	x	x	~~0~~1	~~RFU~~XDA selected

TVR Byte 4:

b8	b7	b6	b5	b4	b3	b2	b1	Meaning
1	x	x	x	x	x	x	x	Transaction exceeds floor limit
x	1	x	x	x	x	x	x	Lower consecutive offline limit exceeded
x	x	1	x	x	x	x	x	Upper consecutive offline limit exceeded
x	x	x	1	x	x	x	x	Transaction selected randomly for online processing
x	x	x	x	1	x	x	x	Merchant forced transaction online
x	x	x	x	x	1	x	x	Biometric Try Limit exceeded

x	x	x	x	x	x	1	x	A selected Biometric Type not supported
x	x	x	x	x	x	x	~~0~~1	~~RFU~~XDA signature verification failed

TVR Byte 5 (Rightmost):

b8	b7	b6	b5	b4	b3	b2	b1	Meaning
1	x	x	x	x	x	x	x	Default TDOL used
x	1	x	x	x	x	x	x	Issuer authentication failed
x	x	1	x	x	x	x	x	Script processing failed before final GENERATE AC
x	x	x	1	x	x	x	x	Script processing failed after final GENERATE AC
x	x	x	x	0	x	x	x	Reserved for use by the EMV Contactless Specifications
x	x	x	x	x	~~0~~1	x	x	CA ECC key missing~~Reserved for use by the EMV Contactless Specifications~~
x	x	x	x	x	x	~~0~~1	x	ECC key recovery failed~~Reserved for use by the EMV Contactless Specifications~~
x	x	x	x	x	x	x	0	Reserved for use by the EMV Contactless Specifications

Table 42: Terminal Verification Results

<Changes to Book 4>

In section 3, Definitions, delete the following terms:

~~**Accelerated Revocation**~~	~~A key revocation performed on a date sooner than the published key expiry date.~~
~~**Key Expiry Date**~~	~~The date after which a signature made with a particular key is no longer valid. Issuer certificates signed by the key must expire on or before this date. Keys may be removed from terminals after this date has passed.~~
~~**Key Life Cycle**~~	~~All phases of key management, from planning and generation, through revocation, destruction, and archiving.~~

~~**Key Replacement**~~	~~The simultaneous revocation of a key and introduction of a key to replace the revoked one.~~
~~**Key Revocation**~~	~~The key management process of withdrawing a key from service and dealing with the legacy of its use. Key revocation can be as scheduled or accelerated.~~
~~**Key Revocation Date**~~	~~The date after which no legitimate cards still in use should contain certificates signed by this key, and therefore the date after which this key can be deleted from terminals. For a planned revocation the Key Revocation Date is the same as the key expiry date.~~
~~**Planned Revocation**~~	~~A key revocation performed as scheduled by the published key expiry date.~~

In section 3, Definitions, add these terms in alphabetical order:

Elliptic Curve Cryptography	Public key cryptography based on the algebraic structure of elliptic curves over finite fields.
Extended Data Authentication	A form of offline dynamic data authentication.
Offline Data Encipherment	Offline encipherment of data, in particular for cardholder PIN and biometric data. See Book 2.

In section 4.1, Abbreviations, add these in alphabetical order:

ECC	Elliptic Curve Cryptography
ODE	Offline Data Encipherment
TAA	Terminal Action Analysis
XDA	Extended Data Authentication

In Section 6.3.2, the second paragraph is updated as:

All other terminals shall ~~support both offline static data authentication (SDA) and offline dynamic data authentication (DDA and optionally CDA)~~ be capable of performing SDA, DDA, optionally CDA, and optionally XDA, as described in Books 2 and 3.

Section 6.3.2.1 is updated as:

6.3.2.1 CDA

The following section applies when the selected form of offline data authentication is CDA.

When ~~the selected form of offline data authentication is CDA and~~ CDA fails prior to the final Terminal Action Analysis (for example, Issuer Public Key recovery fails ~~prior to Terminal Action Analysis~~) preceding the issuance of a first GENERATE AC command~~,~~ or second GENERATE AC command in the case 'unable to go online', the terminal shall set the TVR bit for 'CDA failed' to 1 and request the cryptogram type determined by Terminal Action Analysis. In this case, the GENERATE AC command shall not request a CDA signature and no further CDA processing is performed.

When ~~the selected form of offline data authentication is CDA and~~ a CDA failure is detected after the final Terminal Action Analysis preceding the issuance of a first or second GENERATE AC command, the terminal shall set the 'CDA failed' bit in the TVR to 1 and the following rules apply:

If CDA fails in conjunction with the first GENERATE AC:

- If the Cryptogram Information Data (CID) ~~bit~~ indicates that the card has returned a TC, the terminal shall decline the transaction and not perform a second GENERATE AC command.
- If the CID ~~bit~~ indicates that the card has returned an ARQC, the terminal shall complete the transaction processing by performing an immediate second GENERATE AC command requesting an AAC.

If CDA fails in conjunction with the second GENERATE AC, the terminal shall decline the transaction

If as part of dynamic signature verification the CID was retrieved from the ICC Dynamic Data (as recovered from the Signed Dynamic Application Data), then it is this value that shall be used to determine the cryptogram type. Otherwise the cleartext CID in the GENERATE AC response shall be used.

Section 6.3.2.2 is added:

6.3.2.2 XDA

When the selected form of offline data authentication is XDA, the terminal shall perform the processing defined in the following sections when XDA fails. The first three sections address the three different ways in which XDA can fail and the fourth section describes terminal processing after first GENERATE AC XDA failure.

Note: In the following sections, 'final TAA' means the Terminal Action Analysis that occurs immediately before issuance of a GENERATE AC command.

6.3.2.2.1 CA ECC Public Key Retrieval Failure

If the terminal experiences a CA ECC Public Key retrieval error, the terminal shall set the 'CA ECC key missing' bit in the TVR to 1 *before* the final TAA preceding the first GENERATE AC command. In this case the terminal omits ECC key recovery and XDA signature verification and continues with the processing defined in section 6.3.2.2.4.

6.3.2.2.2 ECC Key Recovery Failure

If the terminal experiences an ECC Issuer Public Key or ICC Public Key recovery error, the terminal shall:

- Set the 'ECC key recovery failed' bit in the TVR *after* issuance of the first GENERATE AC command but *before* the TAA that occurs following the first GENERATE AC command. However, the 'ECC key recovery failed' bit is not set if the 'CA ECC key missing' bit is set.
- Perform the processing defined in section 6.3.2.2.4.

Note: The terminal may perform ECC key recovery before issuance of the first GENERATE AC command, but setting of the 'ECC key recovery failed' bit in the TVR must occur after issuance of the first GENERATE AC command to ensure that the result of final TAA preceding the first GENERATE AC command does not vary based upon when ECC key recovery is performed by the terminal. If the TVR is included as input to generation of the Application Cryptogram, the issuer should set the 'ECC key recovery failed' bit in the TVR to 0 when performing Application Cryptogram verification, as the value of that bit may have changed after the Application Cryptogram was generated.

6.3.2.2.3 XDA Signature Verification Failure

After a GENERATE AC command, if either of the following is true:

- the Signed Dynamic Application Data (SDAD) was not received in the GENERATE AC response
- verification of the Signed Dynamic Application Data (SDAD) failed

then when ECC key recovery has ended the terminal shall:

- Set the 'XDA signature verification failed' bit in the TVR to 1 if CA ECC key retrieval and ECC key recovery were successful.
- For the first GENERATE AC, perform the processing defined in section 6.3.2.2.4.
- For the second GENERATE AC, decline the transaction. **Note:** This will only apply if XDA was successful on the first GENERATE AC as otherwise section 6.3.2.2.4 applies which specifies that the XDA signature on the second GENERATE AC is not verified.

6.3.2.2.4 Terminal Processing After First GENERATE AC XDA Failure

The terminal performs the processing described in this section when it experiences an XDA failure, specifically a CA ECC public key retrieval failure (as defined in section 6.3.2.2.1), an ECC key recovery failure (as defined in section 6.3.2.2.2) or an XDA signature verification failure for first GENERATE AC (as defined in section 6.3.2.2.3).

If the CID indicated that the card returned a TC or ARQC, the terminal shall:

- Perform TAA using the TAC-Denial and IAC-Denial. If the IAC-Denial does not exist, a default value with all bits set to 0 is used.
- If the result of this TAA is to decline the transaction (that is, a TVR bit is 1 and the corresponding TAC-Denial or IAC-Denial bit is also 1), the terminal shall:

- If the CID indicated that the card returned a TC, no second GENERATE AC command shall be issued to complete the transaction.
- If the CID indicated that the card returned an ARQC, a second GENERATE AC command requesting an AAC shall be issued to complete the transaction.

Otherwise (the result of this TAA is not to decline the transaction), the terminal shall attempt to send the transaction online for authorisation[1] and:

- The TVR value included in the online authorisation shall include the updated values of the 'ECC key recovery failed' and the 'XDA signature verification failed' bits.

  Note: If the TVR is included as input to generation of the Application Cryptogram, the issuer should set the 'ECC key recovery failed' and the 'XDA signature verification failed' bits in the TVR to 0 when performing Application Cryptogram verification, as the value of those bits may have changed after the Application Cryptogram was generated.

- If the terminal is able to process the transaction online, then:
  - If the CID indicated that the card returned a TC, the terminal shall not issue a second GENERATE AC command to complete the transaction. Depending on the Authorisation Response Code returned in the response message, the terminal shall determine whether to accept or decline the transaction.
  - If the CID indicated that the card returned an ARQC, the terminal shall issue a second GENERATE AC command to complete the transaction as described in section 6.3.8. Although an XDA signature will be requested it shall not be verified by the terminal.[2]
- If the terminal is for any reason unable to process the transaction online, then:
  - The terminal shall, decline the transaction
  - If the CID indicated that the card returned a TC, the terminal shall not issue a second GENERATE AC command to complete the transaction.
  - If the CID indicated that the card returned an ARQC, the terminal shall issue a second GENERATE AC command requesting an AAC to complete the transaction. Although an XDA signature will be requested it shall not be verified by the terminal.

If the CID indicated that the card returned an AAC, the terminal shall decline the transaction without issuing a second GENERATE AC command.

---

[1] In this scenario, a cryptogram type of TC may be sent in the online authorisation.

[2] an XDA signature is requested only for the sake of consistency - the terminal ignores the XDA signature provided in the response to the second GENERATE AC because it is known that if the issuer has authorised the transaction then it was despite XDA failure on the first GENERATE AC.

In section 6.3.8, the first paragraph is updated as:

Depending on the Authorisation Response Code returned in the response message, the terminal shall determine whether to accept or decline the transaction. It shall issue the second GENERATE AC command to the ICC indicating its decision if the card returned an ARQC in the first GENERATE AC response.

In section 6.4, the first three paragraphs are updated as:

A terminal supporting offline PIN verification or offline biometric verification shall support the VERIFY command. A terminal supporting offline PIN encipherment or offline biometric encipherment shall also support the GET CHALLENGE command. A terminal not supporting offline PIN verification nor offline biometric verification need not support the VERIFY command.

An offline-only terminal and an offline terminal with online capability shall support both SDA ~~and~~, DDA, ~~and~~ may optionally support CDA, and may optionally support XDA.

An online-only terminal need not support SDA or DDA or CDA or XDA. Individual payment systems will define rules for this case.

In section 6.5.2.2, the second paragraph is updated as:

The terminal shall not modify the Authorisation Response Code. ~~The~~ If the card returned an ARQC in the first GENERATE AC response, the terminal shall issue the second GENERATE AC command requesting either a TC for an approval or an AAC for a decline. If the Issuer Authentication Data is present in the authorisation response message, the terminal may issue the EXTERNAL AUTHENTICATE command either before or after the referral data is manually entered.

In section 7.4, the second paragraph is updated as:

The date is used for checking certificate expiration dates for data authentication and/or ~~offline PIN encipherment~~ ODE as well as application expiration/effective dates for processing restrictions. The time may be used for assuring transaction identification uniqueness as well as for input to the application cryptogram algorithm.

In section 10.2, Table 7, the two paragraphs after the table and the note [6] are updated as:

Data Elements	Notes
Acquirer Identifier	
Application Identifier (AID)	
Application Version Number	

Data Elements	Notes
Certification Authority Public Key (RSA) • Certification Authority Public Key Exponent • Certification Authority Public Key Modulus	Required if terminal supports offline data authentication and/or ~~offline PIN encipherment~~ ODE using RSA. See Book 2.
Certification Authority Public Key (ECC)	Required if terminal supports offline data authentication and/or ODE using ECC. See Book 2.
Certification Authority Public Key Index	Required if terminal supports offline data authentication and/or ~~offline PIN encipherment~~ ODE: The key index in conjunction with the Registered Application Provider Identifier (RID) of the payment system AID identifies the key and the algorithm for offline data authentication and/or ~~PIN encipherment~~ ODE. See Book 2.
Default Dynamic Data Authentication Data Object List (DDOL)	Required if terminal supports DDA or CDA.
Default Transaction Certificate Data Object List (TDOL)	If not present, a default TDOL with no data objects in the list shall be assumed.
Maximum Target Percentage to be used for Biased Random Selection	Required if offline terminal with online capability.
Merchant Category Code	
Merchant Identifier	
Merchant Name and Location	
Target Percentage to be used for Random Selection	Required if offline terminal with online capability.

Data Elements	Notes
Terminal Action Code - Default Terminal Action Code - Denial Terminal Action Code - Online	Required if non-zero values to be used[6]
Terminal Capabilities	If the terminal supports XDA or ECC ODE for any application, the terminal shall support an independently configurable Terminal Capabilities value for each application.
Terminal Floor Limit	Required if offline terminal or offline terminal with online capability.
Terminal Identification	
Terminal Risk Management Data	If required by individual payment system rules.
Threshold Value for Biased Random Selection	Required if offline terminal with online capability.
Transaction Currency Code	
Transaction Currency Exponent	
Transaction Reference Currency Code	
Transaction Reference Currency Conversion	
Transaction Reference Currency Exponent	

Table 7: Application Dependent Data Elements

The terminal shall provide the necessary logical key slots to handle the active and future replacement Certification Authority Public Keys necessary for data authentication and/or ~~offline PIN encipherment~~ ODE. Each logical key slot shall contain the following data: RID, Certification Authority Public Key Index, and Certification Authority Public Key.

When the ~~Certification Authority P~~public ~~K~~key is ~~loaded~~ installed into the terminal, the terminal shall verify the integrity of the public key ~~Certification Authority Public Key Check Sum~~ to detect a key entry or transmission error. This may be done by verifying a Certification Authority Public Key Check Sum (for RSA) or self-signed Certification Authority Public Key Certificate (for ECC). See Book 2 section 11.2. ~~This checksum is calculated using the terminal-supported Secure Hash Algorithm.~~ If the verification process fails, the terminal shall not accept the ~~Certification Authority P~~public ~~K~~key and, if operator action is needed, the terminal shall display an error message. ~~After the Certification Authority Public Key is successfully loaded, the terminal should store the Certification Authority Public Key Check Sum.~~

[6] According to Book 3, the default value consists of all bits set to 0, although the 'Offline data authentication was not performed', 'SDA failed', 'DDA failed', ~~and~~ 'CDA failed', 'CA ECC key missing', 'ECC key recovery failed', and 'XDA signature verification failed' bits are strongly recommended to be set to 1 in the Terminal Action Code - Default and Terminal Action Code - Online.

In section 12.1.1, Table 9 is updated as:

Data Element	Condition
Application Interchange Profile * [11]	
Application Transaction Counter *	
~~ARQC~~ Application Cryptogram *	
CID	The CID does not need to be forwarded to the issuer; the presence of this data element is defined in the respective payment system network interface specifications.
CVM Results	
IFD Serial Number	Present if Terminal Identifier does not implicitly refer to IFD Serial Number
Issuer Application Data *	Present if provided by ICC in GENERATE AC command response
Payment Account Reference (PAR)	Present if provided by ICC, at the discretion of the acquirer, subject to payment system requirements.
Terminal Capabilities	
Terminal Type	
TVR *	
Unpredictable Number*	Present if input to application cryptogram calculation

Table 9: ICC-specific Authorisation Request Data Elements

In section 12.1.2, Table 11 is updated as:

Data Element	Condition
Application Interchange Profile * [13]	
Application Transaction Counter *	
Application Usage Control	Present if requested by acquirer
~~ARQC~~ Application Cryptogram *	
CID	The CID does not need to be forwarded to the issuer; the presence of this data element is defined in the respective payment system network interface specifications.
CVM List	Present if requested by acquirer
CVM Results	
IFD Serial Number	Present if Terminal Identifier does not implicitly refer to IFD Serial Number
Issuer Action Code - Default	Present if requested by acquirer
Issuer Action Code – Denial	Present if requested by acquirer
Issuer Action Code – Online	Present if requested by acquirer
Issuer Application Data *	Present if provided by ICC in GENERATE AC command response
Payment Account Reference (PAR)	Present if provided by ICC, at the discretion of the acquirer, subject to payment system requirements.
Terminal Capabilities	
Terminal Type	
TVR *	
Unpredictable Number *	Present if input to application cryptogram calculation

Table 11: ICC-specific Financial Transaction Request Data Elements

In section 12.1.5, Table 17 is updated as:

Data Element	Condition
Application Cryptogram * (i.e., TC, ARQC, or AAC)	ARQC may be used as TC substitute
Application Interchange Profile * [16]	
Application Transaction Counter *	
Application Usage Control	Present if requested by acquirer
CID	The CID does not need to be forwarded to the issuer; the presence of this data element is defined in the respective payment system network interface specifications.
CVM List	Present if requested by acquirer
CVM Results	
IFD Serial Number	Present if Terminal Identifier does not implicitly refer to IFD Serial Number
Issuer Action Code - Default	Present if requested by acquirer
Issuer Action Code – Denial	Present if requested by acquirer
Issuer Action Code – Online	Present if requested by acquirer
Issuer Application Data *	Present if provided by ICC in GENERATE AC command response
Issuer Script Results	Present if script commands to ICC are delivered by terminal
Payment Account Reference (PAR)	Present if provided by ICC, at the discretion of the acquirer, subject to payment system requirements.
Terminal Capabilities	
Terminal Type	
TVR *	
~~TC/ARQC or AAC *~~	~~ARQC may be used as TC substitute~~
Unpredictable Number *	Present if input to application cryptogram calculation

Table 17: ICC-specific Batch Data Capture Data Elements

In section 12.1.7, Table 20 is updated as:

Data Element	Condition
Application Cryptogram (TC or AAC)	
Application Interchange Profile	
Application Transaction Counter	
CID	
CVM Results	
IFD Serial Number	Present if Terminal Identifier does not implicitly refer to IFD Serial Number
Issuer Application Data	Present if provided by ICC in GENERATE AC command response
Issuer Script Results	Present if script commands to ICC are delivered by terminal
Payment Account Reference (PAR)	Present if provided by ICC, at the discretion of the acquirer, subject to payment system requirements.
Terminal Capabilities	
Terminal Type	
TVR	
~~TC or AAC~~	
Unpredictable Number	Present if input to application cryptogram calculation

Table 20: ICC-specific Online Advice Data Elements

Section 12.2.1 is updated as:

During transaction processing, the terminal may send an authorisation request to the acquirer due to at least one of the following conditions:

- Online-only terminal type
- Attendant action (for example, merchant suspicious of cardholder)
- Terminal risk management parameters set by the acquirer
- Terminal action analysis in comparing TVR with Issuer Action Code - Online and Terminal Action Code - Online (see Book 3 section 10.7)

- Card action analysis via its response to the first GENERATE AC command: CID indicates ARQC returned (see Book 3)
- Terminal action analysis after first GENERATE AC with an XDA failure (see section 6.3.2.2.4)

If the terminal is unable to process the transaction online, as described in Book 3, the terminal shall compare the TVR with both Terminal Action Code - Default and Issuer Action Code - Default to determine whether to accept or decline the transaction offline and, if the card returned an ARQC in the first GENERATE AC response, the terminal shall issue the second GENERATE AC command to the ICC indicating its decision:

- If the terminal accepts the transaction, it shall set the Authorisation Response Code to 'Unable to go online, offline accepted'.
- If the terminal declines the transaction, it shall set the Authorisation Response Code to 'Unable to go online, offline declined'.

The result of card risk management performed by the ICC is made known to the terminal through the return of the CID indicating either a TC for an approval or an AAC for a decline.

In section 12.2.3, the second paragraph is updated as:

After repeat(s)[18], if any, of the authorisation request, the terminal shall process the transaction as being unable to go online. As described in Book 3, the terminal shall compare the TVR with both Terminal Action Code - Default and Issuer Action Code - Default to determine whether to accept or decline the transaction offline and, if the card returned an ARQC in the first GENERATE AC response, the terminal shall issue the second GENERATE AC command to the ICC indicating its decision:

Annex A2 Table 26 is updated as:

b8	b7	b6	b5	b4	b3	b2	b1	Meaning
1	x	x	x	x	x	x	x	Plaintext PIN for ICC verification
x	1	x	x	x	x	x	x	Enciphered PIN for online verification
x	x	1	x	x	x	x	x	Signature
x	x	x	1	x	x	x	x	Enciphered PIN for offline verification (RSA ODE)
x	x	x	x	1	x	x	x	No CVM Required
x	x	x	x	x	1	x	x	Online Biometric
x	x	x	x	x	x	1	x	Offline Biometric
x	x	x	x	x	x	x	~~0~~1	Enciphered PIN for offline verification (ECC ODE)~~RFU~~

Table 26: Terminal Capabilities Byte 2 - CVM Capability

Annex A2 Table 27 is updated as:

b8	b7	b6	b5	b4	b3	b2	b1	Meaning
1	x	x	x	x	x	x	x	SDA
x	1	x	x	x	x	x	x	DDA
x	x	1	x	x	x	x	x	Card capture
x	x	x	0	x	x	x	x	RFU
x	x	x	x	1	x	x	x	CDA
x	x	x	x	x	~~0~~1	x	x	~~RFU~~XDA
x	x	x	x	x	x	0	x	RFU
x	x	x	x	x	x	x	0	RFU

Table 27: Terminal Capabilities Byte 3 - Security Capability

Legal Notice

The EMV® Specifications are provided “AS IS” without warranties of any kind, and EMVCo neither assumes nor accepts any liability for any errors or omissions contained in these Specifications. EMVCO DISCLAIMS ALL REPRESENTATIONS AND WARRANTIES, EXPRESS OR IMPLIED, INCLUDING WITHOUT LIMITATION IMPLIED WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE, TITLE AND NON-INFRINGEMENT, AS TO THESE SPECIFICATIONS.

EMVCo makes no representations or warranties with respect to intellectual property rights of any third parties in or in relation to the Specifications. EMVCo undertakes no responsibility to determine whether any implementation of the EMV® Specifications may violate, infringe, or otherwise exercise the patent, copyright, trademark, trade secret, know-how, or other intellectual property rights of third parties, and thus any person who implements any part of the EMV® Specifications should consult an intellectual property attorney before any such implementation.

Without limiting the foregoing, the Specifications may provide for the use of public key encryption and other technology, which may be the subject matter of patents in several countries. Any party seeking to implement these Specifications is solely responsible for determining whether its activities require a license to any such technology, including for patents on public key encryption technology. EMVCo shall not be liable under any theory for any party’s infringement of any intellectual property rights in connection with the EMV® Specifications