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 2

Security and Key Management

Version 4.3
November 2011

* EMV is a registered trademark in the U.S. and other countries and an unregistered trademark elsewhere. The EMV trademark is owned by EMVCo.

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 these 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 these 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 these Specifications.

Revision Log - Version 4.3

The following changes have been made to Book 2 since the publication of Version 4.2. Numbering and cross references in this version have been updated to reflect changes introduced by the published bulletins.

Updated in support of the following Application Notes:

Application Note no. 41 Second Edition: Recommendations for CDA Terminals (revised)

Incorporated changes described in the following Specification Bulletins:

Specification Bulletin no. 74 Second Edition: AES option in EMV

Specification Bulletin no. 78: Removal of DDF Entries from PSE Records

Specification Bulletin no. 88: Application Selection Updates

Specification Bulletin no. 91: AES Support in Common Core Definitions (CCD)

Specification Bulletin no. 92: Various Changes to Book 2

Contents

Part I - General

Tables

Figures

Part I

General

1 Scope

This document, the *Integrated Circuit Card (ICC) Specifications for Payment Systems - Book 2, Security and Key Management,* describes the minimum security functionality required of integrated circuit cards (ICCs) and terminals to ensure correct operation and interoperability. Additional requirements and recommendations are provided with respect to the on-line communication between ICC and issuer and the management of cryptographic keys at terminal, issuer, and payment system level.

The *Integrated Circuit Card Specifications for Payment Systems* includes the following additional documents, all available on http://www.emvco.com:

- Book 1 - Application Independent ICC to Terminal Interface Requirements
- Book 3 - Application Specification
- Book 4 - Cardholder, Attendant, and Acquirer Interface Requirements

EMVCo also publishes security guidelines (see informative references 3 and 5).

1.1 Changes in Version 4.3

This release incorporates all relevant Specification Update Bulletins, Application Notes, amendments, etc. published up to the date of this release.

The Revision Log at the beginning of the Book provides additional detail about changes to this specification.

1.2 Structure

Book 2 consists of the following parts:

Part I	-	**General**
Part II	-	**Security and Key Management Techniques**
Part III	-	**Annexes**
Part IV	-	**Common Core Definitions**

Part I includes this introduction, as well as information applicable to all Books: normative references, definitions, abbreviations, notations, data element format convention, and terminology.

Part II covers:

- Offline static data authentication (SDA)
- Offline dynamic data authentication (DDA and CDA)
- Offline PIN encipherment
- Application cryptogram generation and issuer authentication
- Secure messaging
- Public key management principles and policies
- Terminal security and key management requirements

Part III (Annexes A-D) specifies the security mechanisms and the approved cryptographic algorithms required to implement the security functions specified, provides a list of informative references, and discusses implementation considerations.

Part IV defines an optional extension to be used when implementing the Common Core Definitions (CCD).

The Book also includes a revision log and an index.

1.3 Underlying Standards

This specification is based on the ISO/IEC 7816 series of standards and should be read in conjunction with those standards. However, if any of the provisions or definitions in this specification differ from those standards, the provisions herein shall take precedence.

1.4 Audience

This specification is intended for use by manufacturers of ICCs and terminals, system designers in payment systems, and financial institution staff responsible for implementing financial applications in ICCs.

2 Normative References

The following standards contain provisions that are referenced in these specifications. The latest version shall apply unless a publication date is explicitly stated.

ISO 639-1	Codes for the representation of names of languages – Part 1: Alpha-2 Code **Note:** This standard is updated continuously by ISO. Additions/changes to ISO 639-1:1988: Codes for the Representation of Names of Languages are available on: http://www.loc.gov/standards/iso639-2/php/code_changes.php
ISO 3166	Codes for the representation of names of countries and their subdivisions
ISO 4217	Codes for the representation of currencies and funds
ISO/IEC 7811-1	Identification cards – Recording technique – Part 1: Embossing
ISO/IEC 7811-3	Identification cards – Recording technique – Part 3: Location of embossed characters on ID-1 cards
ISO/IEC 7813	Identification cards – Financial transaction cards
ISO/IEC 7816-1	Identification cards – Integrated circuit(s) cards with contacts – Part 1: Physical characteristics
ISO/IEC 7816-2	Information technology – Identification cards – Integrated circuit(s) cards with contacts – Part 2: Dimensions and location of contacts
ISO/IEC 7816-3	Identification cards — Integrated circuit cards — Part 3: Cards with contacts — Electrical interface and transmission protocols
ISO/IEC 7816-4	Identification cards — Integrated circuit cards — Part 4: Organization, security and commands for interchange

ISO/IEC 7816-5	Identification cards — Integrated circuit cards — Part 5: Registration of application providers
ISO/IEC 7816-6	Identification cards – Integrated circuit cards – Part 6: Interindustry data elements for interchange
ISO 8583:1987	Bank card originated messages – Interchange message specifications – Content for financial transactions
ISO 8583:1993	Financial transaction card originated messages – Interchange message specifications
ISO/IEC 8825-1	Information technology – ASN.1 encoding rules: Specification of Basic Encoding Rules (BER), Canonical Encoding Rules (CER) and Distinguished Encoding Rules (DER)
ISO/IEC 8859	Information processing – 8-bit single-byte coded graphic character sets
ISO 9362	Banking – Banking telecommunication messages – Bank identifier codes
ISO 9564-1:2011	Financial services – Personal Identification Number (PIN) management and security – Part 1: Basic principles and requirements for PINs in card-based systems
ISO/IEC 9796-2:2010	Information technology – Security techniques – Digital signature schemes giving message recovery – Part 2: Integer factorization based mechanisms
ISO/IEC 9797-1:2011	Information technology – Security techniques – Message Authentication Codes - Part 1: Mechanisms using a block cipher
ISO/IEC 10116	Information technology – Security techniques – Modes of operation for an n-bit block cipher
ISO/IEC 10118-3	Information technology – Security techniques – Hash-functions – Part 3: Dedicated hash-functions
ISO/IEC 10373	Identification cards – Test methods

ISO 13491-1	Banking – Secure cryptographic devices (retail) – Part 1: Concepts, requirements and evaluation methods
ISO 13616	Banking and related financial services – International bank account number (IBAN)
ISO 16609	Banking – Requirements for message authentication using symmetric techniques
ISO/IEC 18031	Information technology - Security techniques - Random bit generation
ISO/IEC 18033-3	Information technology – Security techniques – Encryption algorithms – Part 3: Block ciphers

3 Definitions

The following terms are used in one or more books of these specifications.

Accelerated Revocation
A key revocation performed on a date sooner than the published key expiry date.

Application
The application protocol between the card and the terminal and its related set of data.

Application Authentication Cryptogram
An Application Cryptogram generated by the card when declining a transaction

Application Cryptogram
A cryptogram generated by the card in response to a GENERATE AC command. See also:

- Application Authentication Cryptogram
- Authorisation Request Cryptogram
- Transaction Certificate

Authorisation Request Cryptogram
An Application Cryptogram generated by the card when requesting online authorisation

Authorisation Response Cryptogram
A cryptogram generated by the issuer in response to an Authorisation Request Cryptogram.

Asymmetric Cryptographic Technique
A cryptographic technique that uses two related transformations, a public transformation (defined by the public key) and a private transformation (defined by the private key). The two transformations have the property that, given the public transformation, it is computationally infeasible to derive the private transformation.

Authentication
The provision of assurance of the claimed identity of an entity or of data origin.

Block
A succession of characters comprising two or three fields defined as prologue field, information field, and epilogue field.

Byte
8 bits.

Card A payment card as defined by a payment system.

Certificate The public key and identity of an entity together with some other information, rendered unforgeable by signing with the private key of the certification authority which issued that certificate.

Certification Authority Trusted third party that establishes a proof that links a public key and other relevant information to its owner.

Ciphertext Enciphered information.

Cold Reset The reset of the ICC that occurs when the supply voltage (VCC) and other signals to the ICC are raised from the inactive state and the reset (RST) signal is applied.

Combined DDA/Application Cryptogram Generation A form of offline dynamic data authentication.

Command A message sent by the terminal to the ICC that initiates an action and solicits a response from the ICC.

Compromise The breaching of secrecy or security.

Concatenation Two elements are concatenated by appending the bytes from the second element to the end of the first. Bytes from each element are represented in the resulting string in the same sequence in which they were presented to the terminal by the ICC, that is, most significant byte first. Within each byte bits are ordered from most significant bit to least significant. A list of elements or objects may be concatenated by concatenating the first pair to form a new element, using that as the first element to concatenate with the next in the list, and so on.

Contact A conducting element ensuring galvanic continuity between integrated circuit(s) and external interfacing equipment.

Cryptogram Result of a cryptographic operation.

Cryptographic Algorithm

An algorithm that transforms data in order to hide or reveal its information content.

Data Integrity

The property that data has not been altered or destroyed in an unauthorised manner.

Deactivation Sequence

The deactivation sequence defined in section 6.1.5 of Book 1.

Decipherment

The reversal of a corresponding encipherment.

Digital Signature

An asymmetric cryptographic transformation of data that allows the recipient of the data to prove the origin and integrity of the data, and protect the sender and the recipient of the data against forgery by third parties, and the sender against forgery by the recipient.

Dynamic Data Authentication

A form of offline dynamic data authentication

Embossing

Characters raised in relief from the front surface of a card.

Encipherment

The reversible transformation of data by a cryptographic algorithm to produce ciphertext.

Epilogue Field

The final field of a block. It contains the error detection code (EDC) byte(s).

Exclusive-OR

Binary addition with no carry, giving the following values:

$0 + 0 = 0$
$0 + 1 = 1$
$1 + 0 = 1$
$1 + 1 = 0$

Financial Transaction

The act between a cardholder and a merchant or acquirer that results in the exchange of goods or services against payment.

Function

A process accomplished by one or more commands and resultant actions that are used to perform all or part of a transaction.

Guardtime

The minimum time between the trailing edge of the parity bit of a character and the leading edge of the start bit of the following character sent in the same direction.

Hash Function

A function that maps strings of bits to fixed-length strings of bits, satisfying the following two properties:

- It is computationally infeasible to find for a given output an input which maps to this output.
- It is computationally infeasible to find for a given input a second input that maps to the same output.

Additionally, if the hash function is required to be collision-resistant, it must also satisfy the following property:

- It is computationally infeasible to find any two distinct inputs that map to the same output.

Hash Result

The string of bits that is the output of a hash function.

Inactive

The supply voltage (VCC) and other signals to the ICC are in the inactive state when they are at a potential of 0.4 V or less with respect to ground (GND).

Integrated Circuit Module

The sub-assembly embedded into the ICC comprising the IC, the IC carrier, bonding wires, and contacts.

Integrated Circuit(s)

Electronic component(s) designed to perform processing and/or memory functions.

Integrated Circuit(s) Card

A card into which one or more integrated circuits are inserted to perform processing and memory functions.

Interface Device

That part of a terminal into which the ICC is inserted, including such mechanical and electrical devices as may be considered part of it.

Issuer Action Code

Any of the following, which reflect the issuer-selected action to be taken upon analysis of the TVR:

- Issuer Action Code - Default
- Issuer Action Code - Denial
- Issuer Action Code - Online

Kernel The set of functions required to be present on every terminal implementing a specific interpreter. The kernel contains device drivers, interface routines, security and control functions, and the software for translating from the virtual machine language to the language used by the real machine. In other words, the kernel is the implementation of the virtual machine on a specific real machine.

Key A sequence of symbols that controls the operation of a cryptographic transformation.

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 Introduction The process of generating, distributing, and beginning use of a key pair.

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.

Key Withdrawal The process of removing a key from service as part of its revocation.

Keypad Arrangement of numeric, command, and, where required, function and/or alphanumeric keys laid out in a specific manner.

Library
A set of high-level software functions with a published interface, providing general support for terminal programs and/or applications.

Logical Compromise
The compromise of a key through application of improved cryptanalytic techniques, increases in computing power, or combination of the two.

Magnetic Stripe
The stripe containing magnetically encoded information.

Message
A string of bytes sent by the terminal to the card or vice versa, excluding transmission-control characters.

Message Authentication Code
A symmetric cryptographic transformation of data that protects the sender and the recipient of the data against forgery by third parties.

Nibble
The four most significant or least significant bits of a byte.

Padding
Appending extra bits to either side of a data string.

Path
Concatenation of file identifiers without delimitation.

Payment System Environment
A logical construct within the ICC, the entry point to which is a Directory Definition File (DDF) named '1PAY.SYS.DDF01'. This DDF contains a Payment System Directory which in turn contains entries for one or more Application Definition Files (ADFs) which are formatted according to this specification.

Physical Compromise
The compromise of a key resulting from the fact that it has not been securely guarded, or a hardware security module has been stolen or accessed by unauthorised persons.

PIN Pad
Arrangement of numeric and command keys to be used for personal identification number (PIN) entry.

Plaintext
Unenciphered information.

Planned Revocation
A key revocation performed as scheduled by the published key expiry date.

Potential Compromise A condition where cryptanalytic techniques and/or computing power has advanced to the point that compromise of a key of a certain length is feasible or even likely.

Private Key That key of an entity’s asymmetric key pair that should only be used by that entity. In the case of a digital signature scheme, the private key defines the signature function.

Prologue Field The first field of a block. It contains subfields for node address (NAD), protocol control byte (PCB), and length (LEN).

Public Key That key of an entity’s asymmetric key pair that can be made public. In the case of a digital signature scheme, the public key defines the verification function.

Public Key Certificate The public key information of an entity signed by the certification authority and thereby rendered unforgeable.

Response A message returned by the ICC to the terminal after the processing of a command message received by the ICC.

Script A command or a string of commands transmitted by the issuer to the terminal for the purpose of being sent serially to the ICC as commands.

Secret Key A key used with symmetric cryptographic techniques and usable only by a set of specified entities.

Signal Amplitude The difference between the high and low voltages of a signal.

Signal Perturbations Abnormalities occurring on a signal during normal operation such as undershoot/overshoot, electrical noise, ripple, spikes, crosstalk, etc. Random perturbations introduced from external sources are beyond the scope of this specification.

Socket An execution vector defined at a particular point in an application and assigned a unique number for reference.

State H
Voltage high on a signal line. May indicate a logic one or logic zero depending on the logic convention used with the ICC.

State L
Voltage low on a signal line. May indicate a logic one or logic zero depending on the logic convention used with the ICC.

Static Data Authentication
Offline static data authentication

Symmetric Cryptographic Technique
A cryptographic technique that uses the same secret key for both the originator's and recipient's transformation. Without knowledge of the secret key, it is computationally infeasible to compute either the originator's or the recipient's transformation.

T=0
Character-oriented asynchronous half duplex transmission protocol.

T=1
Block-oriented asynchronous half duplex transmission protocol.

Template
Value field of a constructed data object, defined to give a logical grouping of data objects.

Terminal
The device used in conjunction with the ICC at the point of transaction to perform a financial transaction. The terminal incorporates the interface device and may also include other components and interfaces such as host communications.

Terminal Action Code
Any of the following, which reflect the acquirer-selected action to be taken upon analysis of the TVR:

- Terminal Action Code - Default
- Terminal Action Code - Denial
- Terminal Action Code - Online

Terminate Card Session
End the card session by deactivating the IFD contacts according to section 6.1.5 of Book 1 and displaying a message indicating that the ICC cannot be used to complete the transaction

Terminate Transaction
Stop the current application and deactivate the card.

Transaction An action taken by a terminal at the user's request. For a POS terminal, a transaction might be payment for goods, etc. A transaction selects among one or more applications as part of its processing flow.

Transaction Certificate An Application Cryptogram generated by the card when accepting a transaction

Virtual Machine A theoretical microprocessor architecture that forms the basis for writing application programs in a specific interpreter software implementation.

Warm Reset The reset that occurs when the reset (RST) signal is applied to the ICC while the clock (CLK) and supply voltage (VCC) lines are maintained in their active state.

4 Abbreviations, Notations, Conventions, and Terminology

4.1 Abbreviations

μA	Microampere
μm	Micrometre
μs	Microsecond
a	Alphabetic (see section 4.3, Data Element Format Conventions)
AAC	Application Authentication Cryptogram
AC	Application Cryptogram
ACK	Acknowledgment
ADF	Application Definition File
AEF	Application Elementary File
AFL	Application File Locator
AID	Application Identifier
AIP	Application Interchange Profile
an	Alphanumeric (see section 4.3)
ans	Alphanumeric Special (see section 4.3)
APDU	Application Protocol Data Unit
API	Application Program Interface
ARC	Authorisation Response Code
ARPC	Authorisation Response Cryptogram
ARQC	Authorisation Request Cryptogram
ASI	Application Selection Indicator
ASN	Abstract Syntax Notation
ATC	Application Transaction Counter

ATM	Automated Teller Machine
ATR	Answer to Reset
AUC	Application Usage Control
b	Binary (see section 4.3)
BCD	Binary Coded Decimal
BER	Basic Encoding Rules (defined in ISO/IEC 8825–1)
BIC	Bank Identifier Code
BGT	Block Guardtime
BWI	Block Waiting Time Integer
BWT	Block Waiting Time
C	Celsius or Centigrade
CAD	Card Accepting Device
C-APDU	Command APDU
CBC	Cipher Block Chaining
CCD	Common Core Definitions
CCI	Common Core Identifier
CDA	Combined DDA/Application Cryptogram Generation
CDOL	Card Risk Management Data Object List
CID	Cryptogram Information Data
C_{IN}	Input Capacitance
CLA	Class Byte of the Command Message
CLK	Clock
cn	Compressed Numeric (see section 4.3)
CPU	Central Processing Unit
CRL	Certificate Revocation List
CSU	Card Status Update
C-TPDU	Command TPDU
CV	Cryptogram Version

CVM	Cardholder Verification Method
CVR	Card Verification Results
CV Rule	Cardholder Verification Rule
CWI	Character Waiting Time Integer
CWT	Character Waiting Time
D	Bit Rate Adjustment Factor
DAD	Destination Node Address
DC	Direct Current
DDA	Dynamic Data Authentication
DDF	Directory Definition File
DDOL	Dynamic Data Authentication Data Object List
DES	Data Encryption Standard
DF	Dedicated File
DIR	Directory
DOL	Data Object List
ECB	Electronic Code Book
EDC	Error Detection Code
EF	Elementary File
EN	European Norm
etu	Elementary Time Unit
f	Frequency
FC	Format Code
FCI	File Control Information
GND	Ground
Hex	Hexadecimal
HHMMSS	Hours, Minutes, Seconds
I/O	Input/Output
IAC	Issuer Action Code (Denial, Default, Online)

IAD	Issuer Application Data
IBAN	International Bank Account Number
I-block	Information Block
IC	Integrated Circuit
ICC	Integrated Circuit(s) Card
I_{CC}	Current drawn from VCC
IEC	International Electrotechnical Commission
IFD	Interface Device
IFS	Information Field Size
IFSC	Information Field Size for the ICC
IFSD	Information Field Size for the Terminal
IFSI	Information Field Size Integer
IIN	Issuer Identification Number
INF	Information Field
INS	Instruction Byte of Command Message
I_{OH}	High Level Output Current
I_{OL}	Low Level Output Current
ISO	International Organization for Standardization
K_{M}	Master Key
K_{S}	Session Key
L	Length
l.s.	Least Significant
Lc	Exact Length of Data Sent by the TAL in a Case 3 or 4 Command
LCOL	Lower Consecutive Offline Limit
L_{DD}	Length of the ICC Dynamic Data
Le	Maximum Length of Data Expected by the TAL in Response to a Case 2 or 4 Command
LEN	Length

Licc	Exact Length of Data Available or Remaining in the ICC (as Determined by the ICC) to be Returned in Response to the Case 2 or 4 Command Received by the ICC
Lr	Length of Response Data Field
LRC	Longitudinal Redundancy Check
M	Mandatory
mΩ	Milliohm
MΩ	Megohm
m.s.	Most Significant
m/s	Meters per Second
mA	Milliampere
MAC	Message Authentication Code
max.	Maximum
MF	Master File
MHz	Megahertz
min.	Minimum
MK	ICC Master Key for session key generation
mm	Millimetre
MMDD	Month, Day
MMYY	Month, Year
N	Newton
n	Numeric (see section 4.3)
NAD	Node Address
NAK	Negative Acknowledgment
nAs	Nanoampere-second
N_{CA}	Length of the Certification Authority Public Key Modulus
NF	Norme Française
N_{I}	Length of the Issuer Public Key Modulus
N_{IC}	Length of the ICC Public Key Modulus

NIST	National Institute for Standards and Technology
N_{PE}	Length of the ICC PIN Encipherment Public Key Modulus
ns	Nanosecond
O	Optional
O/S	Operating System
P1	Parameter 1
P2	Parameter 2
P3	Parameter 3
PAN	Primary Account Number
PC	Personal Computer
P_{CA}	Certification Authority Public Key
PCB	Protocol Control Byte
PDOL	Processing Options Data Object List
pF	Picofarad
P_{I}	Issuer Public Key
P_{IC}	ICC Public Key
PIN	Personal Identification Number
PIX	Proprietary Application Identifier Extension
POS	Point of Service
pos.	Position
PSE	Payment System Environment
PTS	Protocol Type Selection
R-APDU	Response APDU
R-block	Receive Ready Block
RFU	Reserved for Future Use
RID	Registered Application Provider Identifier
RSA	Rivest, Shamir, Adleman Algorithm
RST	Reset

SAD	Source Node Address
S-block	Supervisory Block
S_{CA}	Certification Authority Private Key
SDA	Static Data Authentication
SFI	Short File Identifier
SHA-1	Secure Hash Algorithm 1
S_I	Issuer Private Key
S_{IC}	ICC Private Key
SK	Session Key
SW1	Status Byte One
SW2	Status Byte Two
TAC	Terminal Action Code(s) (Default, Denial, Online)
TAL	Terminal Application Layer
TC	Transaction Certificate
TCK	Check Character
TDOL	Transaction Certificate Data Object List
t_F	Fall Time Between 90% and 10% of Signal Amplitude
TLV	Tag Length Value
TPDU	Transport Protocol Data Unit
t_R	Rise Time Between 10% and 90% of Signal Amplitude
TS	Initial Character
TSI	Transaction Status Information
TTL	Terminal Transport Layer
TVR	Terminal Verification Results
UCOL	Upper Consecutive Offline Limit
UL	Underwriters Laboratories Incorporated
V	Volt
var.	Variable (see section 4.3)

V_{CC}	Voltage Measured on VCC Contact
VCC	Supply Voltage
V_{IH}	High Level Input Voltage
V_{IL}	Low Level Input Voltage
V_{OH}	High Level Output Voltage
V_{OL}	Low Level Output Voltage
VPP	Programming Voltage
V_{PP}	Voltage Measured on VPP contact
WI	Waiting Time Integer
WTX	Waiting Time Extension
WWT	Work Waiting Time
YYMM	Year, Month
YYMMDD	Year, Month, Day

4.2 Notations

'0' to '9' and 'A' to 'F'	16 hexadecimal characters
xx	Any value
$A := B$	A is assigned the value of B
$A = B$	Value of A is equal to the value of B
$A \equiv B \bmod n$	Integers A and B are congruent modulo the integer n, that is, there exists an integer d such that $(A - B) = dn$
A mod n	The reduction of the integer A modulo the integer n, that is, the unique integer r, $0 \leq r < n$, for which there exists an integer d such that $A = dn + r$
A / n	The integer division of A by n, that is, the unique integer d for which there exists an integer r, $0 \leq r < n$, such that $A = dn + r$
$Y := ALG(K)[X]$	Encipherment of a data block X with a block cipher as specified in Annex A1, using a secret key K
$X = ALG^{-1}(K)[Y]$	Decipherment of a data block Y with a block cipher as specified in Annex A1, using a secret key K
$Y := Sign\ (S_K)[X]$	The signing of a data block X with an asymmetric reversible algorithm as specified in Annex A2, using the private key S_K
$X = Recover(P_K)[Y]$	The recovery of the data block X with an asymmetric reversible algorithm as specified in Annex A2, using the public key P_K
$C := (A \mid\mid B)$	The concatenation of an n-bit number A and an m-bit number B, which is defined as $C = 2^m A + B$.
Leftmost	Applies to a sequence of bits, bytes, or digits and used interchangeably with the term “most significant”. If $C = (A \mid\mid B)$ as above, then A is the leftmost n bits of C.

Rightmost — Applies to a sequence of bits, bytes, or digits and used interchangeably with the term “least significant”. If C = (A | | B) as above, then B is the rightmost m bits of C.

H := Hash[MSG] — Hashing of a message MSG of arbitrary length using a 160-bit hash function

X ⊕ Y — The symbol '⊕' denotes bit-wise exclusive-OR and is defined as follows:

X ⊕ Y — The bit-wise exclusive-OR of the data blocks X and Y. If one data block is shorter than the other, then it is first padded to the left with sufficient binary zeros to make it the same length as the other.

4.3 Data Element Format Conventions

The EMV specifications use the following data element formats:

a Alphabetic data elements contain a single character per byte. The permitted characters are alphabetic only (a to z and A to Z, upper and lower case).

an Alphanumeric data elements contain a single character per byte. The permitted characters are alphabetic (a to z and A to Z, upper and lower case) and numeric (0 to 9).

ans Alphanumeric Special data elements contain a single character per byte. The permitted characters and their coding are shown in the Common Character Set table in Annex B of Book 4.

There is one exception: The permitted characters for Application Preferred Name are the non-control characters defined in the ISO/IEC 8859 part designated in the Issuer Code Table Index associated with the Application Preferred Name.

b These data elements consist of either unsigned binary numbers or bit combinations that are defined elsewhere in the specification.

Binary example: The Application Transaction Counter (ATC) is defined as “b” with a length of two bytes. An ATC value of 19 is stored as Hex '00 13'.

Bit combination example: Processing Options Data Object List (PDOL) is defined as “b” with the format shown in Book 3, section 5.4.

cn Compressed numeric data elements consist of two numeric digits (having values in the range Hex '0'–'9') per byte. These data elements are left justified and padded with trailing hexadecimal 'F's.

Example: The Application Primary Account Number (PAN) is defined as “cn” with a length of up to ten bytes. A value of 1234567890123 may be stored in the Application PAN as Hex '12 34 56 78 90 12 3F FF' with a length of 8.

n Numeric data elements consist of two numeric digits (having values in the range Hex '0' – '9') per byte. These digits are right justified and padded with leading hexadecimal zeroes. Other specifications sometimes refer to this data format as Binary Coded Decimal (“BCD”) or unsigned packed.

Example: Amount, Authorised (Numeric) is defined as “n 12” with a length of six bytes. A value of 12345 is stored in Amount, Authorised (Numeric) as Hex '00 00 00 01 23 45'.

var.	Variable data elements are variable length and may contain any bit combination. Additional information on the formats of specific variable data elements is available elsewhere.

4.4 Terminology

proprietary	Not defined in this specification and/or outside the scope of this specification
shall	Denotes a mandatory requirement
should	Denotes a recommendation

Part II

Security and Key Management Techniques

5 Static Data Authentication (SDA)

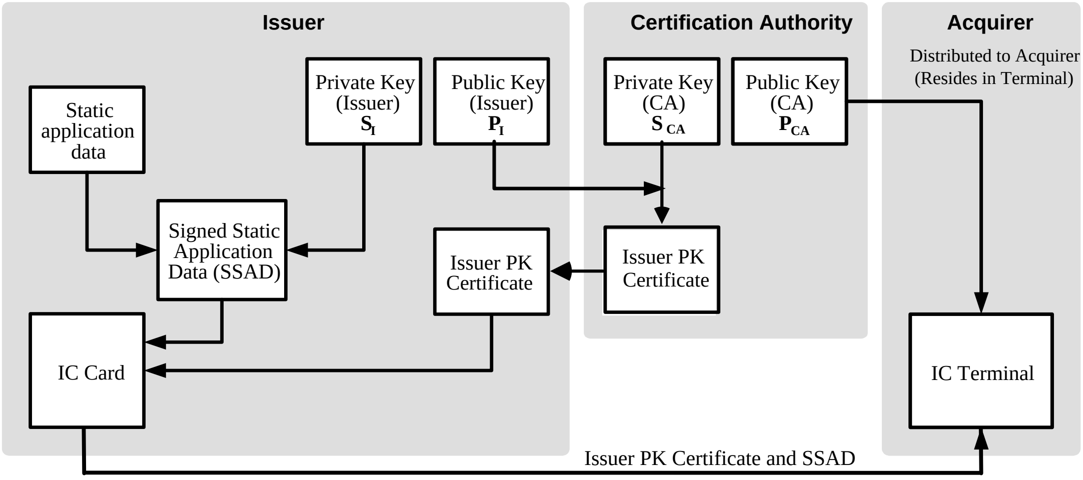
Offline static data authentication is performed by the terminal using a digital signature scheme based on public key techniques to confirm the legitimacy of critical ICC-resident static data. This detects unauthorised alteration of data after personalisation.

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.


Issuer
Certification Authority
Acquirer
Distributed to Acquirer (Resides in Terminal)
Static application data
Private Key (Issuer) S_I
Public Key (Issuer) P_I
Private Key (CA) S_{CA}
Public Key (CA) P_{CA}
Signed Static Application Data (SSAD)
Issuer PK Certificate
Issuer PK Certificate
IC Card
IC Terminal
Issuer PK Certificate and SSAD

Card provides to Terminal:

- Issuer PK Certificate (P_I signed by CA using S_{CA})
- Signed Static Application Data (SSAD) (signed by the Issuer using S_I)

Terminal:

- Uses P_{CA} to verify that the Issuer’s P_I was signed by the CA
- Uses P_I to verify that the Card’s SSAD was signed by the Issuer

Figure 1: Diagram of SDA

ICCs that support SDA shall contain the data elements listed in Table 1:

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.

SDA shall use a reversible algorithm as specified in Annex A2.1 and Annex B2. Section 5.1 contains an overview of the keys and certificates involved in the SDA process, and sections 5.2 to 5.4 specify the three main steps in the process, namely:

- Retrieval of the Certification Authority Public Key by the terminal
- Retrieval of the Issuer Public Key by the terminal
- Verification of the Signed Static Application Data by the terminal

If SDA fails then the terminal shall set the 'SDA failed' bit in the Terminal Verification Results (TVR) to 1.

5.1 Keys and Certificates

To support SDA, an ICC shall contain the Signed Static Application Data, which is signed with the Issuer Private Key. The Issuer Public Key shall be stored on the ICC with a public key certificate.

The bit length of all moduli shall be a multiple of 8, the leftmost bit of its leftmost byte being 1. All lengths are given in bytes.

The signature scheme specified in Annex A2.1 is applied to the data specified in Table 2 using the Certification Authority Private Key S_{CA} in order to obtain the Issuer Public Key Certificate.

The public key pair of the certification authority has a public key modulus of N_{CA} bytes, where $N_{CA} \leq 248$. The Certification Authority Public Key Exponent shall be equal to 3 or $2^{16} + 1$.

The signature scheme specified in Annex A2.1 is applied to the data specified in Table 3 using the Issuer Private Key S_I in order to obtain the Signed Static Application Data.

The public key pair of the issuer has an Issuer Public Key Modulus of N_I bytes, where $N_I \leq N_{CA} \leq 248$. If $N_I > (N_{CA} - 36)$, the Issuer Public Key Modulus is split into two parts, namely:

- the Leftmost Digits of the Issuer Public Key, consisting of the $N_{CA} - 36$ most significant bytes of the modulus, and
- the Issuer Public Key Remainder, consisting of the remaining $N_I - (N_{CA} - 36)$ least significant bytes of the modulus.

The Issuer Public Key Exponent shall be equal to 3 or $2^{16} + 1$.

All the information necessary for SDA is specified in Table 4 and stored in the ICC. With the exception of the RID, which can be obtained from the Application Identifier (AID; see Book 1, section 12.2.1), this information may be retrieved with the READ RECORD command. If any of this data is missing, SDA has failed.

Field Name	Length	Description	Format
Certificate Format	1	Hex value '02'	b
Issuer Identifier	4	Leftmost 3-8 digits from the Primary Account Number (PAN) (padded to the right with Hex 'F's)	cn 8
Certificate Expiration Date	2	MMYY after which this certificate is invalid	n 4
Certificate Serial Number	3	Binary number unique to this certificate assigned by the certification authority	b
Hash Algorithm Indicator	1	Identifies the hash algorithm used to produce the Hash Result in the digital signature scheme [1]	b
Issuer Public Key Algorithm Indicator	1	Identifies the digital signature algorithm to be used with the Issuer Public Key [1]	b
Issuer Public Key Length	1	Identifies the length of the Issuer Public Key Modulus in bytes	b
Issuer Public Key Exponent Length	1	Identifies the length of the Issuer Public Key Exponent in bytes	b
Issuer Public Key or Leftmost Digits of the Issuer Public Key	$N_{CA} - 36$	If $N_I \leq N_{CA} - 36$, consists of the full Issuer Public Key padded to the right with $N_{CA} - 36 - N_I$ bytes of value 'BB' If $N_I > N_{CA} - 36$, consists of the $N_{CA} - 36$ most significant bytes of the Issuer Public Key [2]	b
Issuer Public Key Remainder	0 or $N_I - N_{CA} + 36$	Present only if $N_I > N_{CA} - 36$ and consists of the $N_I - N_{CA} + 36$ least significant bytes of the Issuer Public Key.	b
Issuer Public Key Exponent	1 or 3	Issuer Public Key Exponent equal to 3 or $2^{16} + 1$	b

Table 2: Issuer Public Key Data to be Signed by Certification Authority (i.e., input to the hash algorithm)

[1] See Annex B for specific values assigned to approved algorithms.

[2] As can be seen in Annex A2.1, $N_{CA} - 22$ bytes of the data signed are retrieved from the signature. Since the length of the first through the eighth data elements in Table 2 is 14 bytes, there are $N_{CA} - 22 - 14 = N_{CA} - 36$ bytes remaining in the signature to store the Issuer Public Key Modulus.

Field Name	Length	Description	Format
Signed Data Format	1	Hex Value '03'	b
Hash Algorithm Indicator	1	Identifies the hash algorithm used to produce the Hash Result in the digital signature scheme [3]	b
Data Authentication Code	2	Issuer-assigned code	b
Pad Pattern	$N_I - 26$	Pad pattern consisting of $N_I - 26$ bytes of value 'BB' [4]	b
Static Data to be Authenticated	var.	Static data to be authenticated as specified in section 10.3 of Book 3 (see also section 5.1.1)	—

Table 3: Static Application Data to be Signed by Issuer (i.e., input to the hash algorithm)

[3] See Annex B for specific values assigned to approved algorithms.

[4] As can be seen in Annex A2.1, $N_I - 22$ bytes of the data signed are retrieved from the signature. Since the length of the first through the third data elements in Table 3 is 4 bytes, there are $N_I - 22 - 4 = N_I - 26$ bytes left for the data to be stored in the signature.

5.1.1 Static Data to be Authenticated

Input to the authentication process is formed from the records identified by the AFL, followed by the value of the Application Interchange Profile (AIP), if identified by the optional Static Data Authentication Tag List (tag '9F4A'). If present, the Static Data Authentication Tag List shall only contain the tag '82' identifying the AIP.

Tag	Length	Value	Format
—	5	Registered Application Provider Identifier (RID)	b
'8F'	1	Certification Authority Public Key Index	b
'90'	N_{CA}	Issuer Public Key Certificate	b
'92'	$N_I - N_{CA} + 36$	Issuer Public Key Remainder, if present	b
'9F32'	1 or 3	Issuer Public Key Exponent	b
'93'	N_I	Signed Static Application Data	b
—	Var.	Static data to be authenticated as specified in section 10.3 of Book 3 (see also section 5.1.1)	—

Table 4: Data Objects Required for SDA

5.1.2 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 SDA, 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, SDA fails as described in section 5.3 Step 10.

At a minimum each entry in the CRL shall contain the following data:

Name	Description	Format	Length
Registered Application Provider Identifier (RID)	Identifiers the application provider	b	5
Certification Authority Public Key Index	Identifies the public key in conjunction with the RID	b	1
Certificate Serial Number	Number unique to this certificate assigned by the certification authority	b	3
Additional Data	Optional terminal proprietary data, such as the date the certificate was added to the revocation list	b	var

Table 5: Minimum Data for Certificate Revocation List Entry

Additional data such as the date the certificate was added to the CRL may be included in the CRL entry.

The terminal shall be able to support at least thirty entries in the CRL for each RID for which the terminal has CA Public Keys.

The terminal shall be able to update the CRL as requested by the acquirer. The payment systems provide these updates to the acquirer. A reliable method of maintaining the CRL is defined by the terminal vendor and the acquirer and should meet the security requirements of the acquirer. It is the responsibility of the payment system to ensure that the number of revoked certificates does not exceed the maximum number of entries that terminals are required to support and the responsibility of the acquirer to ensure that appropriate entries are deleted in order to make way for new entries.

5.2 Retrieval of Certification Authority Public Key

The terminal reads the Certification Authority Public Key Index. Using this index and the RID, the terminal shall identify and retrieve the terminal-stored Certification Authority Public Key Modulus and Exponent and the associated key-related information, and the corresponding algorithm to be used. If the terminal does not have the key stored associated with this index and RID, SDA has failed.

5.3 Retrieval of Issuer Public Key

1. If the Issuer Public Key Certificate has a length different from the length of the Certification Authority Public Key Modulus obtained in the previous section, SDA has failed.
2. In order to obtain the recovered data specified in Table 6, apply the recovery function specified in Annex A2.1 to the Issuer Public Key Certificate using the Certification Authority Public Key in conjunction with the corresponding algorithm. If the Recovered Data Trailer is not equal to 'BC', SDA has failed.

Field Name	Length	Description	Format
Recovered Data Header	1	Hex Value '6A'	b
Certificate Format	1	Hex Value '02'	b
Issuer Identifier	4	Leftmost 3-8 digits from the PAN (padded to the right with Hex 'F's)	cn 8
Certificate Expiration Date	2	MMYY after which this certificate is invalid	n 4
Certificate Serial Number	3	Binary number unique to this certificate assigned by the certification authority	b
Hash Algorithm Indicator	1	Identifies the hash algorithm used to produce the Hash Result in the digital signature scheme [5]	b
Issuer Public Key Algorithm Indicator	1	Identifies the digital signature algorithm to be used with the Issuer Public Key [5]	b
Issuer Public Key Length	1	Identifies the length of the Issuer Public Key Modulus in bytes	b
Issuer Public Key Exponent Length	1	Identifies the length of the Issuer Public Key Exponent in bytes	b
Issuer Public Key or Leftmost Digits of the Issuer Public Key	N_{CA} –36	If $N_I \leq N_{CA} - 36$, consists of the full Issuer Public Key padded to the right with $N_{CA} - 36 - N_I$ bytes of value 'BB' If $N_I > N_{CA} - 36$, consists of the $N_{CA} - 36$ most significant bytes of the Issuer Public Key [6]	b
Hash Result	20	Hash of the Issuer Public Key and its related information	b
Recovered Data Trailer	1	Hex value 'BC'	b

Table 6: Format of Data Recovered from Issuer Public Key Certificate

3. Check the Recovered Data Header. If it is not '6A', SDA has failed.

[5] See Annex B for specific values assigned to approved algorithms.

[6] As can be seen in Annex A2.1, $N_{CA} - 22$ bytes of the data signed are retrieved from the signature. Since the length of the second through the ninth data elements in Table 6 is 14 bytes, there are $N_{CA} - 22 - 14 = N_{CA} - 36$ bytes left for the data to be stored in the signature.

4. Check the Certificate Format. If it is not '02', SDA has failed.
5. Concatenate from left to right the second to the tenth data elements in Table 6 (that is, Certificate Format through Issuer Public Key or Leftmost Digits of the Issuer Public Key), followed by the Issuer Public Key Remainder (if present), and finally the Issuer Public Key Exponent.
6. Apply the indicated hash algorithm (derived from the Hash Algorithm Indicator) to the result of the concatenation of the previous step to produce the hash result.
7. Compare the calculated hash result from the previous step with the recovered Hash Result. If they are not the same, SDA has failed.
8. Verify that the Issuer Identifier matches the leftmost 3-8 PAN digits (allowing for the possible padding of the Issuer Identifier with hexadecimal 'F's). If not, SDA has failed.
9. Verify that the last day of the month specified in the Certificate Expiration Date is equal to or later than today's date. If the Certificate Expiration Date is earlier than today's date, the certificate has expired, in which case SDA has failed.
10. Verify that the concatenation of RID, Certification Authority Public Key Index, and Certificate Serial Number is valid. If not, SDA has failed.[7]
11. If the Issuer Public Key Algorithm Indicator is not recognised, SDA has failed.
12. If all the checks above are correct, concatenate the Leftmost Digits of the Issuer Public Key and the Issuer Public Key Remainder (if present) to obtain the Issuer Public Key Modulus, and continue with the next steps for the verification of the Signed Static Application Data.

[7] This step is optional and is to allow the revocation of the Issuer Public Key Certificate against a Certification Revocation List that may be kept by the terminal (see section 5.1.2).

5.4 Verification of Signed Static Application Data

1. If the Signed Static Application Data has a length different from the length of the Issuer Public Key Modulus, SDA has failed.
2. In order to obtain the Recovered Data specified in Table 7, apply the recovery function specified in Annex A2.1 on the Signed Static Application Data using the Issuer Public Key in conjunction with the corresponding algorithm. If the Recovered Data Trailer is not equal to 'BC', SDA has failed.

Field Name	Length	Description	Format
Recovered Data Header	1	Hex value '6A'	b
Signed Data Format	1	Hex value '03'	b
Hash Algorithm Indicator	1	Identifies the hash algorithm used to produce the Hash Result in the digital signature scheme [8]	b
Data Authentication Code	2	Issuer-assigned code	b
Pad Pattern	$N_I - 26$	Pad pattern consisting of $N_I - 26$ bytes of value 'BB' [9]	b
Hash Result	20	Hash of the Static Application Data to be authenticated	b
Recovered Data Trailer	1	Hex Value 'BC'	b

Table 7: Format of Data Recovered from Signed Static Application Data

3. Check the Recovered Data Header. If it is not '6A', SDA has failed.
4. Check the Signed Data Format. If it is not '03', SDA has failed.
5. Concatenate from left to right the second to the fifth data elements in Table 7 (that is, Signed Data Format through Pad Pattern), followed by the static data to be authenticated as specified in section 10.3 of Book 3. If the Static Data Authentication Tag List is present and contains tags other than '82', then SDA has failed.

[8] See Annex B for specific values assigned to approved algorithms.

[9] As can be seen in Annex A2.1, $N_I - 22$ bytes of the data signed are retrieved from the signature. Since the length of the second through the fourth data elements in Table 7 is 4 bytes, there are $N_I - 22 - 4 = N_I - 26$ bytes left for the data to be stored in the signature.

6. Apply the indicated hash algorithm (derived from the Hash Algorithm Indicator) to the result of the concatenation of the previous step to produce the hash result.
7. Compare the calculated hash result from the previous step with the recovered Hash Result. If they are not the same, SDA has failed.

If all of the above steps were executed successfully, SDA was successful. The Data Authentication Code recovered in Table 7 shall be stored in tag '9F45'.

6 Offline Dynamic Data Authentication

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.

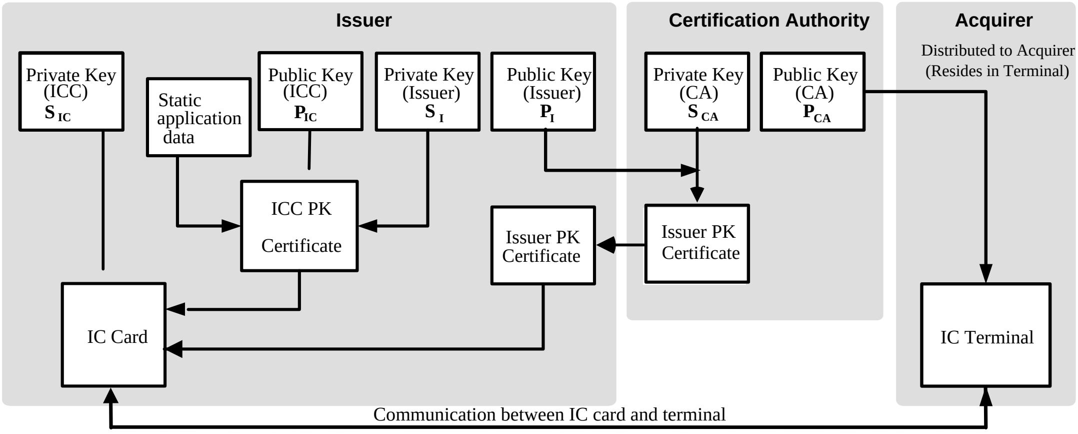
Two forms of offline dynamic data authentication exist:

- 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.

[10] In order to ensure that the ICC uses the correct value for the Unpredictable Number, the Issuer needs to ensure that both CDOL1 and CDOL2 contain tag '9F37'.


Issuer
Certification Authority
Acquirer
Distributed to Acquirer (Resides in Terminal)
Private Key (ICC) S_{IC}
Static application data
Public Key (ICC) P_{IC}
Private Key (Issuer) S_I
Public Key (Issuer) P_I
Private Key (CA) S_{CA}
Public Key (CA) P_{CA}
ICC PK Certificate
Issuer PK Certificate
Issuer PK Certificate
IC Card
IC Terminal
Communication between IC card and terminal

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 2: Diagram of offline dynamic data authentication

ICCs that support offline dynamic data authentication 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

ICCs that support offline dynamic data authentication 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.

Offline dynamic data authentication 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 offline dynamic data authentication 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.

6.1 Keys and Certificates

To support offline dynamic data authentication, an ICC shall own its own unique public key pair consisting of a private signature key and the corresponding public verification key. The ICC Public Key shall be stored on the ICC in a public key certificate.

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 retrieve and authenticate the ICC Public Key, which is then employed to verify the ICC's dynamic signature.

The bit length of all moduli shall be a multiple of 8, the leftmost bit of its leftmost byte being 1. All lengths are given in bytes.

The signature scheme as specified in Annex A2.1 is applied on the data in Table 10 and on the data in Table 11 using the Certification Authority Private Key S_{CA} and the Issuer Private Key S_I in order to obtain the Issuer Public Key Certificate and ICC Public Key Certificate, respectively.

The public key pair of the certification authority has a Certification Authority Public Key Modulus of N_{CA} bytes, where $N_{CA} \leq 248$. The Certification Authority Public Key Exponent shall be equal to 3 or $2^{16} + 1$.

The public key pair of the issuer has a Public Key Modulus of N_I bytes, where $N_I \leq N_{CA} \leq 248$. If $N_I > (N_{CA} – 36)$, the Issuer Public Key Modulus is divided into two parts, one part consisting of the $N_{CA} – 36$ most significant bytes of the modulus (the Leftmost Digits of the Issuer Public Key) and a second part consisting of the remaining $N_I – (N_{CA} – 36)$ least significant bytes of the modulus (the Issuer Public Key Remainder). Section D1.1 details additional restrictions on the length of the Issuer Public Key. The Issuer Public Key Exponent shall be equal to 3 or $2^{16} + 1$.

The public key pair of the ICC has an ICC Public Key Modulus of N_{IC} bytes, where $N_{IC} \leq N_I \leq N_{CA} \leq 248$. If $N_{IC} > (N_I – 42)$, the ICC Public Key Modulus is divided into two parts, one part consisting of the $N_I – 42$ most significant bytes of the modulus (the Leftmost Digits of the ICC Public Key) and a second part consisting of the remaining $N_{IC} – (N_I – 42)$ least significant bytes of the modulus (the ICC Public Key Remainder). Section D1.2 details additional restrictions on the length of the ICC Public Key. The ICC Public Key Exponent shall be equal to 3 or $2^{16} + 1$.

To execute offline dynamic data authentication, the terminal shall first retrieve and authenticate the ICC Public Key (this process is called ICC Public Key authentication). All the information necessary for ICC Public Key authentication is specified in Table 12 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. If any of this data is missing, offline dynamic data authentication has failed.

Field Name	Length	Description	Format
Certificate Format	1	Hex value '02'	b
Issuer Identifier	4	Leftmost 3-8 digits from the PAN (padded to the right with Hex 'F's)	cn 8
Certificate Expiration Date	2	MMYY after which this certificate is invalid	n 4
Certificate Serial Number	3	Binary number unique to this certificate assigned by the certification authority	b
Hash Algorithm Indicator	1	Identifies the hash algorithm used to produce the Hash Result in the digital signature scheme [11]	b
Issuer Public Key Algorithm Indicator	1	Identifies the digital signature algorithm to be used with the Issuer Public Key [11]	b
Issuer Public Key Length	1	Identifies the length of the Issuer Public Key Modulus in bytes	b
Issuer Public Key Exponent Length	1	Identifies the length of the Issuer Public Key Exponent in bytes	b
Issuer Public Key or Leftmost Digits of the Issuer Public Key	$N_{CA} - 36$	If $N_I \le N_{CA} - 36$, consists of the full Issuer Public Key padded to the right with $N_{CA} - 36 - N_I$ bytes of value 'BB' If $N_I > N_{CA} - 36$, consists of the $N_{CA} - 36$ most significant bytes of the Issuer Public Key [12]	b
Issuer Public Key Remainder	0 or $N_I - N_{CA} + 36$	Present only if $N_I > N_{CA} - 36$ and consists of the $N_I - N_{CA} + 36$ least significant bytes of the Issuer Public Key	b
Issuer Public Key Exponent	1 or 3	Issuer Public Key Exponent equal to 3 or $2^{16} + 1$	b

Table 10: Issuer Public Key Data to be Signed by Certification Authority (i.e., input to the hash algorithm)

[11] See Annex B for specific values assigned to approved algorithms.

[12] As can be seen in Annex A2.1, $N_{CA} - 22$ bytes of the data signed are retrieved from the signature. Since the length of the first through the eighth data elements in Table 10 is 14 bytes, there are $N_{CA} - 22 - 14 = N_{CA} - 36$ bytes left for the data to be stored in the signature.

Field Name	Length	Description	Format
Certificate Format	1	Hex value '04'	b
Application PAN	10	PAN (padded to the right with Hex 'F's)	cn 20
Certificate Expiration Date	2	MMYY after which this certificate is invalid	n 4
Certificate Serial Number	3	Binary number unique to this certificate assigned by the issuer	b
Hash Algorithm Indicator	1	Identifies the hash algorithm used to produce the Hash Result in the digital signature scheme [13]	b
ICC Public Key Algorithm Indicator	1	Identifies the digital signature algorithm to be used with the ICC Public Key [13]	b
ICC Public Key Length	1	Identifies the length of the ICC Public Key Modulus in bytes	b
ICC Public Key Exponent Length	1	Identifies the length of the ICC Public Key Exponent in bytes	b
ICC Public Key or Leftmost Digits of the ICC Public Key	$N_I - 42$	If $N_{IC} \leq N_I - 42$, consists of the full ICC Public Key padded to the right with $N_I - 42 - N_{IC}$ bytes of value 'BB' If $N_{IC} > N_I - 42$, consists of the $N_I - 42$ most significant bytes of the ICC Public Key [14]	b
ICC Public Key Remainder	0 or $N_{IC} - N_I + 42$	Present only if $N_{IC} > N_I - 42$ and consists of the $N_{IC} - N_I + 42$ least significant bytes of the ICC Public Key	b
ICC Public Key Exponent	1 or 3	ICC Public Key Exponent equal to 3 or $2^{16} + 1$	b
Static Data to be Authenticated	Var.	Static data to be authenticated as specified in section 10.3 of Book 3 (see also section 6.1.1)	b

Table 11: ICC Public Key Data to be Signed by Issuer (i.e., input to the hash algorithm)

[13] See Annex B for specific values assigned to approved algorithms.

[14] As can be seen in Annex A2.1, $N_I - 22$ bytes of the data signed are retrieved from the signature. Since the length of the first through the eighth data elements in Table 11 is 20 bytes, there are $N_I - 22 - 20 = N_I - 42$ bytes left for the data to be stored in the signature.

6.1.1 Static Data to be Authenticated

Input to the authentication process is formed from the records identified by the AFL, followed by the value of the AIP, if identified by the optional Static Data Authentication Tag List (tag '9F4A'). If present, the Static Data Authentication Tag List shall only contain the tag '82' identifying the AIP.

Tag	Length	Value	Format
—	5	Registered Application Provider Identifier (RID)	b
'8F'	1	Certification Authority Public Key Index	b
'90'	N_{CA}	Issuer Public Key Certificate	b
'92'	$N_I – N_{CA} + 36$	Issuer Public Key Remainder, if present	b
'9F32'	1 or 3	Issuer Public Key Exponent	b
'9F46'	N_I	ICC Public Key Certificate	b
'9F48'	$N_{IC} – N_I + 42$	ICC Public Key Remainder, if present	b
'9F47'	1 or 3	ICC Public Key Exponent	b
—	Var.	Static data to be authenticated as specified in section 10.3 of Book 3 (see also section 6.1.1)	—

Table 12: Data Objects Required for Public Key Authentication for offline dynamic data authentication

6.1.2 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 dynamic data authentication (DDA or CDA), 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, dynamic data authentication fails as described in Section 6.3 Step 10.

The requirements for the CRL are listed in Section 5.1.2.

6.2 Retrieval of Certification Authority Public Key

The terminal reads the Certification Authority Public Key Index. Using this index and the RID, the terminal can identify and retrieve the terminal-stored Certification Authority Public Key Modulus and Exponent and associated key-related information, and the corresponding algorithm to be used. If the terminal does not have the key stored associated with this index and RID, offline dynamic data authentication has failed.

6.3 Retrieval of Issuer Public Key

1. If the Issuer Public Key Certificate has a length different from the length of the Certification Authority Public Key Modulus obtained in the previous section, offline dynamic data authentication has failed.
2. In order to obtain the recovered data specified in Table 13, apply the recovery function as specified in Annex A2.1 on the Issuer Public Key Certificate using the Certification Authority Public Key in conjunction with the corresponding algorithm. If the Recovered Data Trailer is not equal to 'BC', offline dynamic data authentication has failed.

Field Name	Length	Description	Format
Recovered Data Header	1	Hex value '6A'	b
Certificate Format	1	Hex value '02'	b
Issuer Identifier	4	Leftmost 3-8 digits from the PAN (padded to the right with Hex 'F's)	cn 8
Certificate Expiration Date	2	MMYY after which this certificate is invalid	n 4
Certificate Serial Number	3	Binary number unique to this certificate assigned by the certification authority	b
Hash Algorithm Indicator	1	Identifies the hash algorithm used to produce the Hash Result in the digital signature scheme [15]	b
Issuer Public Key Algorithm Indicator	1	Identifies the digital signature algorithm to be used with the Issuer Public Key [15]	b
Issuer Public Key Length	1	Identifies the length of the Issuer Public Key Modulus in bytes	b
Issuer Public Key Exponent Length	1	Identifies the length of the Issuer Public Key Exponent in bytes	b
Issuer Public Key or Leftmost Digits of the Issuer Public Key	N_{CA} – 36	If $N_I \le N_{CA} - 36$, consists of the full Issuer Public Key padded to the right with $N_{CA} - 36 - N_I$ bytes of value 'BB' If $N_I > N_{CA} - 36$, consists of the $N_{CA} - 36$ most significant bytes of the Issuer Public Key [16]	b
Hash Result	20	Hash of the Issuer Public Key and its related information	b
Recovered Data Trailer	1	Hex value 'BC'	b

Table 13: Format of Data Recovered from Issuer Public Key Certificate

[15] See Annex B for specific values assigned to approved algorithms.

[16] As can be seen in Annex A2.1, $N_{CA} - 22$ bytes of the data signed are retrieved from the signature. Since the length of the second through the ninth data elements in Table 13 is 14 bytes, there are $N_{CA} - 22 - 14 = N_{CA} - 36$ bytes left for the data to be stored in the signature.

3. Check the Recovered Data Header. If it is not '6A', offline dynamic data authentication has failed.
4. Check the Certificate Format. If it is not '02', offline dynamic data authentication has failed.
5. Concatenate from left to right the second to the tenth data elements in Table 13 (that is, Certificate Format through Issuer Public Key or Leftmost Digits of the Issuer Public Key), followed by the Issuer Public Key Remainder (if present), and finally the Issuer Public Key Exponent.
6. Apply the indicated hash algorithm (derived from the Hash Algorithm Indicator) to the result of the concatenation of the previous step to produce the hash result.
7. Compare the calculated hash result from the previous step with the recovered Hash Result. If they are not the same, offline dynamic data authentication has failed.
8. Verify that the Issuer Identifier matches the leftmost 3-8 PAN digits (allowing for the possible padding of the Issuer Identifier with hexadecimal 'F's). If not, offline dynamic data authentication has failed.
9. Verify that the last day of the month specified in the Certificate Expiration Date is equal to or later than today's date. If the Certificate Expiration Date is earlier than today's date, the certificate has expired, in which case offline dynamic data authentication has failed.
10. Verify that the concatenation of RID, Certification Public Key Index, and Certificate Serial Number is valid. If not, offline dynamic data authentication has failed.[17]
11. If the Issuer Public Key Algorithm Indicator is not recognised, offline dynamic data authentication has failed.
12. If all the checks above are correct, concatenate the Leftmost Digits of the Issuer Public Key and the Issuer Public Key Remainder (if present) to obtain the Issuer Public Key Modulus, and continue with the next steps for the retrieval of the ICC Public Key.

[17] This step is optional and is to allow the revocation of the Issuer Public Key Certificate against a Certification Revocation List that may be kept by the terminal (see section 6.1.2).

6.4 Retrieval of ICC Public Key

1. If the ICC Public Key Certificate has a length different from the length of the Issuer Public Key Modulus obtained in the previous section, offline dynamic data authentication has failed.
2. In order to obtain the recovered data specified in Table 14, apply the recovery function as specified in Annex A2.1 on the ICC Public Key Certificate using the Issuer Public Key in conjunction with the corresponding algorithm. If the Recovered Data Trailer is not equal to 'BC', offline dynamic data authentication has failed.

Field Name	Length	Description	Format
Recovered Data Header	1	Hex Value '6A'	b
Certificate Format	1	Hex Value '04'	b
Application PAN	10	PAN (padded to the right with Hex 'F's)	cn 20
Certificate Expiration Date	2	MMYY after which this certificate is invalid	n 4
Certificate Serial Number	3	Binary number unique to this certificate assigned by the issuer	b
Hash Algorithm Indicator	1	Identifies the hash algorithm used to produce the Hash Result in the digital signature scheme [18]	b
ICC Public Key Algorithm Indicator	1	Identifies the digital signature algorithm to be used with the ICC Public Key [18]	b
ICC Public Key Length	1	Identifies the length of the ICC Public Key Modulus in bytes	b
ICC Public Key Exponent Length	1	Identifies the length of the ICC Public Key Exponent in bytes	b
ICC Public Key or Leftmost Digits of the ICC Public Key	$N_I - 42$	If $N_{IC} \leq N_I - 42$, consists of the full ICC Public Key padded to the right with $N_I - 42 - N_{IC}$ bytes of value 'BB' [19] If $N_{IC} > N_I - 42$, consists of the $N_I - 42$ most significant bytes of the ICC Public Key	b
Hash Result	20	Hash of the ICC Public Key and its related information	b
Recovered Data Trailer	1	Hex Value 'BC'	b

Table 14: Format of Data Recovered from ICC Public Key Certificate

[18] See Annex B for specific values assigned to approved algorithms.

[19] As can be seen in Annex A2.1, $N_I - 22$ bytes of the data signed are retrieved from the signature. Since the length of the second through the ninth data elements in Table 14 is 20 bytes, there are $N_I - 22 - 20 = N_I - 42$ bytes left for the data to be stored in the signature.

3. Check the Recovered Data Header. If it is not '6A', offline dynamic data authentication has failed.
4. Check the Certificate Format. If it is not '04', offline dynamic data authentication has failed.
5. Concatenate from left to right the second to the tenth data elements in Table 14 (that is, Certificate Format through ICC Public Key or Leftmost Digits of the ICC Public Key), followed by the ICC Public Key Remainder (if present), the ICC Public Key Exponent, and finally the static data to be authenticated specified in section 10.3 of Book 3. If the Static Data Authentication Tag List is present and contains tags other than '82', then offline dynamic data authentication has failed.
6. Apply the indicated hash algorithm (derived from the Hash Algorithm Indicator) to the result of the concatenation of the previous step to produce the hash result.
7. Compare the calculated hash result from the previous step with the recovered Hash Result. If they are not the same, offline dynamic data authentication has failed.
8. Compare the recovered PAN to the Application PAN read from the ICC. If they are not the same, offline dynamic data authentication has failed.
9. Verify that the last day of the month specified in the Certificate Expiration Date is equal to or later than today's date. If not, offline dynamic data authentication has failed.
10. If the ICC Public Key Algorithm Indicator is not recognised, offline dynamic data authentication has failed.
11. If all the checks above are correct, concatenate the Leftmost Digits of the ICC Public Key and the ICC Public Key Remainder (if present) to obtain the ICC Public Key Modulus, and continue with the actual offline dynamic data authentication described in the two sections below.

6.5 Dynamic Data Authentication (DDA)

6.5.1 Dynamic Signature Generation

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

1. The terminal issues an INTERNAL AUTHENTICATE command including the concatenation of the data elements specified by the DDOL according to the rules specified in section 5.4 of Book 3.

   The ICC may contain the DDOL, but there shall be a default DDOL in the terminal, specified by the payment system, for use in case the DDOL is not present in the ICC.

   It is mandatory that the DDOL contains the Unpredictable Number generated by the terminal (tag '9F37', 4 bytes binary).

   If any of the following cases occurs, DDA has failed.

   - The ICC does not contain a DDOL and the terminal does not contain a default DDOL.
   - The DDOL in the ICC does not include the Unpredictable Number.
   - The ICC does not contain a DDOL and the default DDOL in the terminal does not include the Unpredictable Number.

2. The ICC generates a digital signature as described in Annex A2.1 on the data specified in Table 15 using its ICC Private Key S_{IC} in conjunction with the corresponding algorithm. The result is called the Signed Dynamic Application Data.

Field Name	Length	Description	Format
Signed Data Format	1	Hex value '05'	b
Hash Algorithm Indicator	1	Identifies the hash algorithm used to produce the Hash Result [20]	b
ICC Dynamic Data Length	1	Identifies the length L_{DD} of the ICC Dynamic Data in bytes	b
ICC Dynamic Data	L_{DD}	Dynamic data generated by and/or stored in the ICC	—
Pad Pattern	$N_{IC} - L_{DD} - 25$	$(N_{IC} - L_{DD} - 25)$ padding bytes of value 'BB' [21]	b
Terminal Dynamic Data	var.	Concatenation of the data elements specified by the DDOL	—

Table 15: Dynamic Application Data to be Signed (i.e., input to the hash algorithm)

The length L_{DD} of the ICC Dynamic Data satisfies $0 \le L_{DD} \le N_{IC} - 25$. The 3-9 leftmost bytes of the ICC Dynamic Data shall consist of the 1-byte length of the ICC Dynamic Number, followed by the 2-8 byte value of the ICC Dynamic Number (tag '9F4C', 2-8 bytes binary). The ICC Dynamic Number is a time-variant parameter generated by the ICC (it can for example be an unpredictable number or a counter incremented each time the ICC receives an INTERNAL AUTHENTICATE command).

In addition to those specified in Table 12, the data objects necessary for DDA are specified in Table 16.

Tag	Length	Value	Format
'9F4B'	N_{IC}	Signed Dynamic Application Data	b
'9F49'	Var.	DDOL	b

Table 16: Additional Data Objects Required for Dynamic Signature Generation and Verification

[20] See Annex B for specific values assigned to approved algorithms.

[21] As can be seen in Annex A2.1, $N_{IC} - 22$ bytes of the data signed is recovered from the signature. Since the length of the first three data elements in Table 15 is three bytes, there are $N_{IC} - L_{DD} - 22 - 3 = N_{IC} - L_{DD} - 25$ bytes remaining for the data to be stored in the signature.

6.5.2 Dynamic Signature Verification

In this section it is assumed that the terminal has successfully retrieved the ICC Public Key. The verification of the dynamic signature takes place in the following steps.

1. If the Signed Dynamic Application Data has a length different from the length of the ICC Public Key Modulus, DDA has failed.
2. To obtain the recovered data specified in Table 17, apply the recovery function as specified in Annex A2.1 on the Signed Dynamic Application Data using the ICC Public Key in conjunction with the corresponding algorithm. If the Recovered Data Trailer is not equal to 'BC', DDA has failed.

Field Name	Length	Description	Format
Recovered Data Header	1	Hex value '6A'	b
Signed Data Format	1	Hex value '05'	b
Hash Algorithm Indicator	1	Identifies the hash algorithm used to produce the Hash Result in the digital signature scheme [22]	b
ICC Dynamic Data Length	1	Identifies the length of the ICC Dynamic Data in bytes	b
ICC Dynamic Data	L_{DD}	Dynamic data generated by and/or stored in the ICC	—
Pad Pattern	$N_{IC} - L_{DD} - 25$	($N_{IC} - L_{DD} - 25$) padding bytes of value 'BB' [23]	b
Hash Result	20	Hash of the Dynamic Application Data and its related information	b
Recovered Data Trailer	1	Hex value 'BC'	b

Table 17: Format of Data Recovered from Signed Dynamic Application Data

[22] See Annex B for specific values assigned to approved algorithms.

[23] As can be seen in Annex A2.1, $N_{IC} - 22$ bytes of the data signed are retrieved from the signature. Since the length of the second through the fourth data elements in Table 17 is 3 bytes, there are $N_{IC} - L_{DD} - 22 - 3 = N_{IC} - L_{DD} - 25$ bytes left for the data to be stored in the signature.

3. Check the Recovered Data Header. If it is not '6A', DDA has failed.
4. Check the Signed Data Format. If it is not '05', DDA has failed.
5. Concatenate from left to right the second to the sixth data elements in Table 17 (that is, Signed Data Format through Pad Pattern), followed by the data elements specified by the DDOL.
6. Apply the indicated hash algorithm (derived from the Hash Algorithm Indicator) to the result of the concatenation of the previous step to produce the hash result.
7. Compare the calculated hash result from the previous step with the recovered Hash Result. If they are not the same, DDA has failed.

If all the above steps were executed successfully, DDA was successful. The ICC Dynamic Number contained in the ICC Dynamic Data recovered in Table 17 shall be stored in tag '9F4C'.

6.6 Combined DDA/Application Cryptogram Generation (CDA)

CDA consists of a dynamic signature generated by the ICC (similar to DDA but including Application Cryptogram (AC) generation) followed by verification of the signature by the terminal.

It is applicable to both the first and second GENERATE AC commands and requires the retrieval of the relevant public keys as described in Sections 6.2, 6.3 and 6.4. Since the public keys are not required until the CDA signature is verified as part of processing the response to the first GENERATE AC, retrieval of the public keys may happen any time before verifying the CDA signature.

During retrieval of the public keys, errors may result in CDA failure (TVR bit for 'CDA failed' is set to 1). These errors include but are not limited to failure of public key retrieval and invalid format of records to be authenticated (see Book 3 Section 10.3).

For the first GENERATE AC command, and for the second GENERATE AC command in the case 'unable to go online', the cryptogram type requested by the terminal is always determined by the final Terminal Action Analysis preceding the GENERATE AC command. If any of the above errors are detected prior to the final Terminal Action Analysis, then the terminal shall not request CDA in the GENERATE AC command. When the GENERATE AC command is issued with a CDA request, then if any of the above errors are detected subsequently, the eventual result will be an offline decline in accordance with the paragraphs beginning "If CDA fails in conjunction" in Book 4 Section 6.3.2.

In sections 6.1.1 and 6.6.2 it is assumed that:

- Both the ICC and the terminal support CDA.
- The cryptogram to be requested is not an Application Authentication Cryptogram (AAC), i.e. Terminal Action Analysis has not resulted in offline decline.
- The TVR bit for 'CDA failed' is not set to 1 prior to final Terminal Action Analysis.
- Except when returning an AAC, the ICC always replies with a CDA signature when requested by the terminal.

In the case of the first GENERATE AC command:

- When requesting an ARQC, the terminal may request it with or without a CDA signature. When an ARQC is requested without a CDA signature, then the terminal shall set the TVR bit for 'Offline data authentication was not performed' to 1[24] prior to issuance of the GENERATE AC command. When an ARQC is requested without a CDA signature, the processes described in sections 6.6.1 and 6.6.2 are not performed.

- When requesting a TC, the terminal shall request it with a CDA signature.
- When requesting an AAC, the terminal shall request it without a CDA signature.

In the case of the second GENERATE AC command:

- The terminal shall set the TVR bit for 'Offline data authentication was not performed' to 0[25] prior to issuance of the GENERATE AC command. If the terminal is processing the transaction as ‘unable to go online’ then the TVR bit setting shall be done before the associated terminal action analysis.
- When requesting a TC:
  - If the terminal is processing the transaction as 'unable to go online' (and the result of terminal action analysis is to request a TC), then the terminal shall request a TC with a CDA signature.
  - If the terminal is not processing the transaction as ‘unable to go online’, then the terminal may request the TC with or without a CDA signature.
- When requesting an AAC, the terminal shall request it without a CDA signature.

6.6.1 Dynamic Signature Generation

The generation of the combined dynamic signature and Application Cryptogram takes place in the following steps.

1. The terminal issues a first or second GENERATE AC command with the 'CDA signature requested' bit in the GENERATE AC command set to 1 according to sections 6.5.5.4 and 9.3 of Book 3.
2. If the ICC is to respond with a TC or ARQC, the ICC performs the following steps:
   a. The ICC generates the TC or ARQC.
   b. The ICC applies the hash algorithm specified by the Hash Algorithm Indicator to the concatenation from left to right of the following data elements:

[24] This updated TVR is used if requested in CDOL1.

[25] This updated TVR is used if requested in CDOL2.

In the case of the first GENERATE AC command:

- The values of the data elements specified by, and in the order they appear in the PDOL, and sent by the terminal in the GET PROCESSING OPTIONS command.[26]
- The values of the data elements specified by, and in the order they appear in the CDOL1, and sent by the terminal in the first GENERATE AC command.[26]
- The tags, lengths, and values of the data elements returned by the ICC in the response to the GENERATE AC command in the order they are returned, with the exception of the Signed Dynamic Application Data.

In the case of the second GENERATE AC command:

- The values of the data elements specified by, and in the order they appear in the PDOL, and sent by the terminal in the GET PROCESSING OPTIONS command.[26]
- The values of the data elements specified by, and in the order they appear in the CDOL1, and sent by the terminal in the first GENERATE AC command.[26]
- The values of the data elements specified by, and in the order they appear in the CDOL2, and sent by the terminal in the second GENERATE AC command.
- The tags, lengths, and values of the data elements returned by the ICC in the response to the GENERATE AC command in the order they are returned, with the exception of the Signed Dynamic Application Data.

The 20-byte result is called the Transaction Data Hash Code.

c. The ICC applies the digital signature scheme as specified in Annex A2.1 on the data specified in Table 18 using its ICC Private Key S_{IC} in conjunction with the corresponding algorithm. The result is called the Signed Dynamic Application Data.

[26] At the time of issuance of the command, the terminal is required to store the values of these data elements to later perform the signature verification process as specified in section 6.6.2.

Field Name	Length	Description	Format
Signed Data Format	1	Hex Value '05'	b
Hash Algorithm Indicator	1	Identifies the hash algorithm used to produce the Hash Result [27]	b
ICC Dynamic Data Length	1	Identifies the length L_{DD} of the ICC Dynamic Data in bytes	b
ICC Dynamic Data	L_{DD}	Dynamic data generated by and/or stored in the ICC (See Table 19)	—
Pad Pattern	$N_{IC} - L_{DD} - 25$	($N_{IC} - L_{DD} - 25$) padding bytes of value 'BB' [28]	b
Unpredictable Number	4	Unpredictable Number generated by the terminal	b

Table 18: Dynamic Application Data to be Signed
(i.e., input to the hash algorithm in Annex A2.1.2)

The length L_{DD} of the ICC Dynamic Data satisfies $0 \le L_{DD} \le N_{IC} - 25$. The 32-38 leftmost bytes of the ICC Dynamic Data shall consist of the concatenation of the data specified in Table 19.

Length	Value	Format
1	ICC Dynamic Number Length	b
2-8	ICC Dynamic Number	b
1	Cryptogram Information Data	b
8	TC or ARQC	b
20	Transaction Data Hash Code	b

Table 19: 32-38 Leftmost Bytes of ICC Dynamic Data

[27] See Annex B for specific values assigned to approved algorithms.

[28] As can be seen in Annex A2.1, $N_{IC} - 22$ bytes of the data signed is recovered from the signature. Since the length of the first three data elements in Table 18 is three bytes, there are $N_{IC} - L_{DD} - 22 - 3 = N_{IC} - L_{DD} - 25$ bytes remaining for the data to be stored in the signature.

The ICC Dynamic Number is a time-variant parameter generated by the ICC (it can for example be an unpredictable number or a counter incremented each time the ICC receives the first GENERATE AC command during a transaction).

The ICC response to the first GENERATE AC command shall be coded according to format 2 as specified in section 6.5.5.4 of Book 3 (constructed data object with tag '77') and shall contain at least the mandatory data objects (TLV coded in the response) specified in Table 20, and optionally the Issuer Application Data.

Tag	Length	Value	Presence
'9F27'	1	Cryptogram Information Data	M
'9F36'	2	Application Transaction Counter	M
'9F4B'	N_{IC}	Signed Dynamic Application Data	M
'9F10'	Var. up to 32	Issuer Application Data	O

Table 20: Data Objects Included in Response to GENERATE AC for TC or ARQC

3. If the ICC responds with an AAC, the ICC response shall be coded according to either format 1 or format 2 as specified in section 6.5.5.4 of Book 3 and shall contain at least the mandatory data elements specified in Table 21, and optionally the Issuer Application Data.

Tag	Length	Value	Presence
'9F27'	1	Cryptogram Information Data	M
'9F36'	2	Application Transaction Counter	M
'9F26'	8	AAC	M
'9F10'	Var. up to 32	Issuer Application Data	O

Table 21: Data Objects Included in Response to GENERATE AC for AAC

6.6.2 Dynamic Signature Verification

In this section it is assumed that the terminal has successfully retrieved the ICC Public Key as described in sections 6.2, 6.3, and 6.4.

On receiving the GENERATE AC response, the terminal determines the type of Application Cryptogram by inspecting the cleartext CID in the response.

If the ICC has responded with an AAC, then the terminal shall decline the transaction.

If the ICC has responded with a TC or ARQC, the terminal retrieves from the response the data objects specified in Table 21 and executes the following steps:

1. If the Signed Dynamic Application Data has a length different from the length of the ICC Public Key Modulus, CDA has failed.
2. To obtain the recovered data specified in Table 22, apply the recovery function as specified in Annex A2.1 on the Signed Dynamic Application Data using the ICC Public Key in conjunction with the corresponding algorithm. If the Recovered Data Trailer is not equal to 'BC', CDA has failed.

Field Name	Length	Description	Format
Recovered Data Header	1	Hex Value '6A'	b
Signed Data Format	1	Hex Value '05'	b
Hash Algorithm Indicator	1	Identifies the hash algorithm used to produce the Hash Result in the digital signature scheme [29]	b
ICC Dynamic Data Length	1	Identifies the length of the ICC Dynamic Data in bytes	b
ICC Dynamic Data	L_{DD}	Dynamic data generated by and/or stored in the ICC	—
Pad Pattern	$N_{IC} - L_{DD} - 25$	($N_{IC} - L_{DD} - 25$) padding bytes of value 'BB' [30]	b
Hash Result	20	Hash of the Dynamic Application Data and its related information	b
Recovered Data Trailer	1	Hex Value 'BC'	b

Table 22: Format of Data Recovered from Signed Dynamic Application Data

3. Check the Recovered Data Header. If it is not '6A', CDA has failed.
4. Check the Signed Data Format. If it is not '05', CDA has failed.

[29] See Annex B for specific values assigned to approved algorithms.

[30] As can be seen in Annex A2.1, $N_{IC} - 22$ bytes of the data signed are retrieved from the signature. Since the length of the second through the fourth data elements in Table 22 is 3 bytes, there are $N_{IC} - L_{DD} - 22 - 3 = N_{IC} - L_{DD} - 25$ bytes left for the data to be stored in the signature.

5. Retrieve from the ICC Dynamic Data the data specified in Table 19.
6. Check that the Cryptogram Information Data retrieved from the ICC Dynamic Data is equal to the Cryptogram Information Data obtained from the response to the GENERATE AC command. If this is not the case, CDA has failed.
7. Concatenate from left to right the second to the sixth data elements in Table 22 (that is, Signed Data Format through Pad Pattern), followed by the Unpredictable Number.
8. Apply the indicated hash algorithm (derived from the Hash Algorithm Indicator) to the result of the concatenation of the previous step to produce the hash result.
9. Compare the calculated hash result from the previous step with the recovered Hash Result. If they are not the same, CDA has failed.
10. Concatenate from left to right the values of the following data elements:

    In the case of the first GENERATE AC command:

    - The values of the data elements specified by, and in the order they appear in the PDOL, and sent by the terminal in the GET PROCESSING OPTIONS command.
    - The values of the data elements specified by, and in the order they appear in the CDOL1, and sent by the terminal in the first GENERATE AC command.
    - The tags, lengths, and values of the data elements returned by the ICC in the response to the GENERATE AC command in the order they are returned, with the exception of the Signed Dynamic Application Data.

    In the case of the second GENERATE AC command:

    - The values of the data elements specified by, and in the order they appear in the PDOL, and sent by the terminal in the GET PROCESSING OPTIONS command.
    - The values of the data elements specified by, and in the order they appear in the CDOL1, and sent by the terminal in the first GENERATE AC command.
    - The values of the data elements specified by, and in the order they appear in the CDOL2, and sent by the terminal in the second GENERATE AC command.
    - The tags, lengths, and values of the data elements returned by the ICC in the response to the GENERATE AC command in the order they are returned, with the exception of the Signed Dynamic Application Data.
11. Apply the indicated hash algorithm (derived from the Hash Algorithm Indicator) to the result of the concatenation of the previous step to produce the Transaction Data Hash Code.

12. Compare the calculated Transaction Data Hash Code from the previous step with the Transaction Data Hash Code retrieved from the ICC Dynamic Data in Step 5. If they are not the same, CDA has failed.

If all the above steps were executed successfully, CDA was successful. The ICC Dynamic Number and the ARQC or TC contained in the ICC Dynamic Data recovered in Table 19 shall be stored in tag '9F4C' and in tag '9F26', respectively.

6.6.3 Sample CDA Flow

The figures on the following pages are an example of how a terminal might perform CDA. This sample flow provides a generalised illustration of the concepts of CDA. 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.

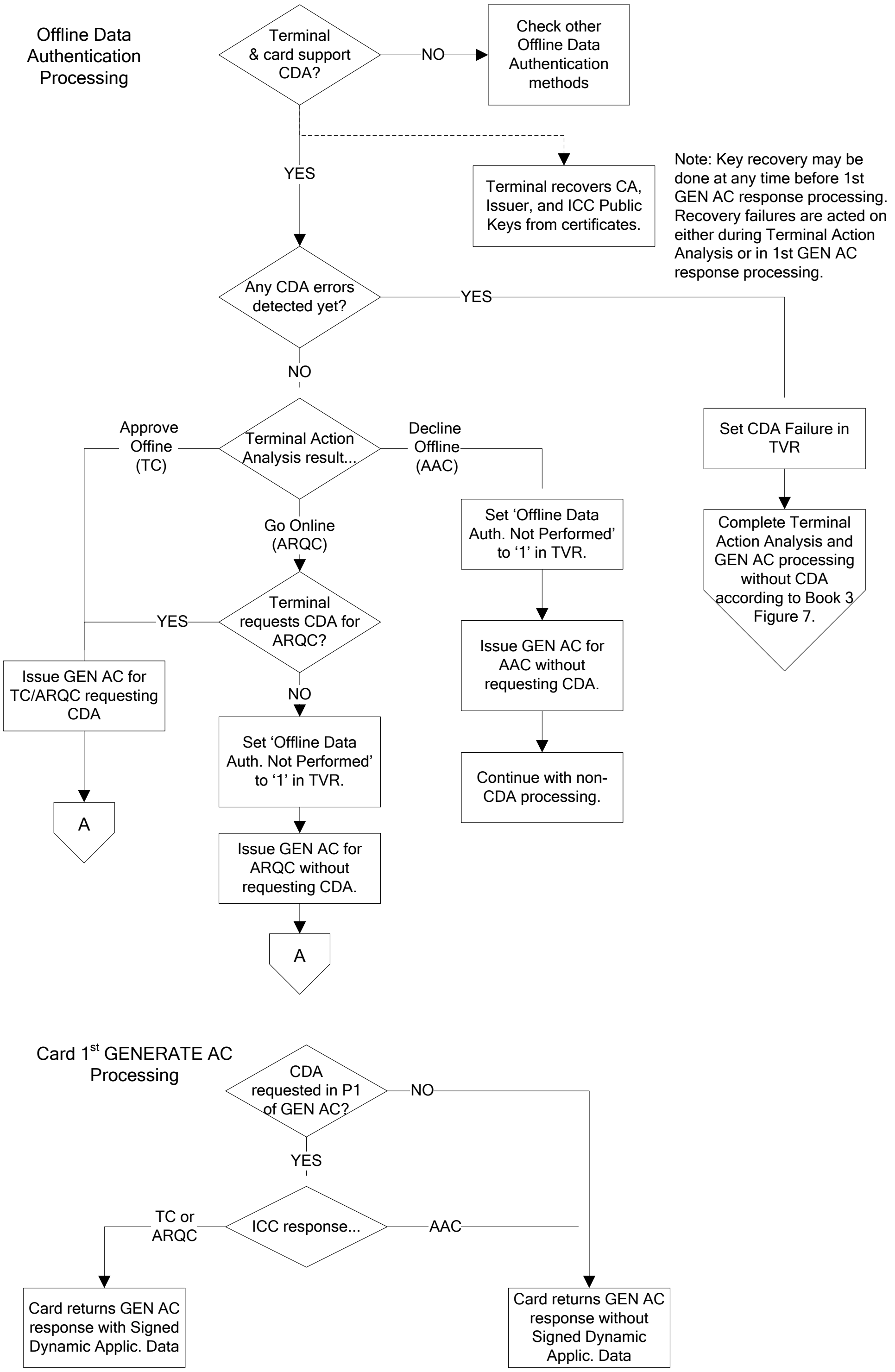

Offline Data Authentication Processing
Terminal & card support CDA?
NO
Check other Offline Data Authentication methods
YES
Terminal recovers CA, Issuer, and ICC Public Keys from certificates.
Note: Key recovery may be done at any time before 1st GEN AC response processing. Recovery failures are acted on either during Terminal Action Analysis or in 1st GEN AC response processing.
Any CDA errors detected yet?
YES
NO
Set CDA Failure in TVR
Complete Terminal Action Analysis and GEN AC processing without CDA according to Book 3 Figure 7.
Approve Offline (TC)
Terminal Action Analysis result...
Decline Offline (AAC)
Go Online (ARQC)
Set 'Offline Data Auth. Not Performed' to '1' in TVR.
Issue GEN AC for AAC without requesting CDA.
Continue with non-CDA processing.
Terminal requests CDA for ARQC?
YES
NO
Issue GEN AC for TC/ARQC requesting CDA
A
Set 'Offline Data Auth. Not Performed' to '1' in TVR.
Issue GEN AC for ARQC without requesting CDA.
A
Card 1st GENERATE AC Processing
CDA requested in P1 of GEN AC?
NO
YES
ICC response...
TC or ARQC
AAC
Card returns GEN AC response with Signed Dynamic Applic. Data
Card returns GEN AC response without Signed Dynamic Applic. Data

Figure 3: CDA Sample Flow Part 1 of 3

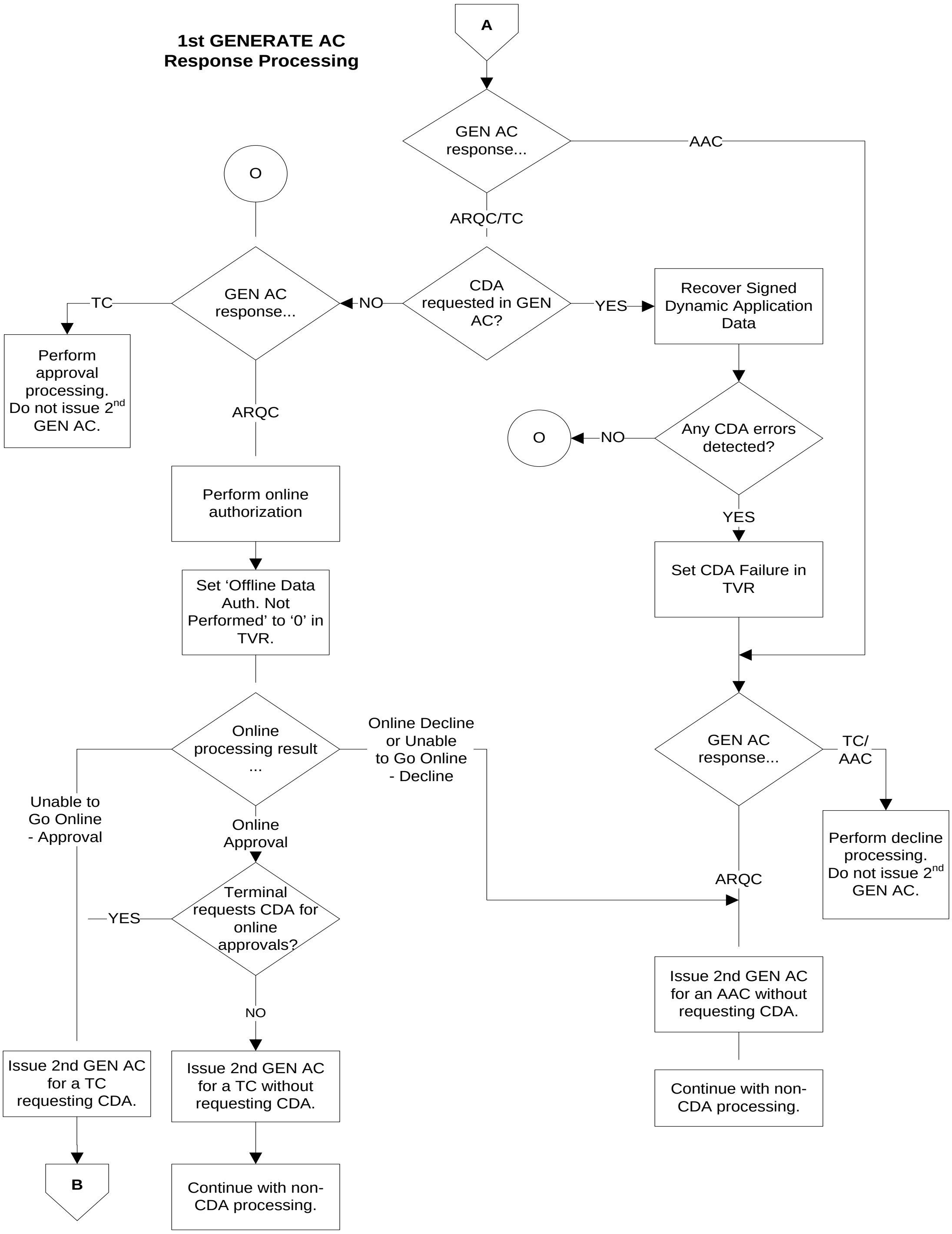

1st GENERATE AC
Response Processing
A
GEN AC response...
AAC
ARQC/TC
O
CDA requested in GEN AC?
NO
YES
Recover Signed Dynamic Application Data
GEN AC response...
TC
Perform approval processing. Do not issue 2nd GEN AC.
ARQC
Any CDA errors detected?
O
NO
YES
Perform online authorization
Set CDA Failure in TVR
Set ‘Offline Data Auth. Not Performed’ to ‘0’ in TVR.
Online processing result ...
Online Decline or Unable to Go Online - Decline
GEN AC response...
TC/ AAC
Unable to Go Online - Approval
Online Approval
Perform decline processing. Do not issue 2nd GEN AC.
ARQC
Terminal requests CDA for online approvals?
YES
NO
Issue 2nd GEN AC for an AAC without requesting CDA.
Continue with non-CDA processing.
Issue 2nd GEN AC for a TC requesting CDA.
Issue 2nd GEN AC for a TC without requesting CDA.
B
Continue with non-CDA processing.

Figure 4: CDA Sample Flow Part 2 of 3

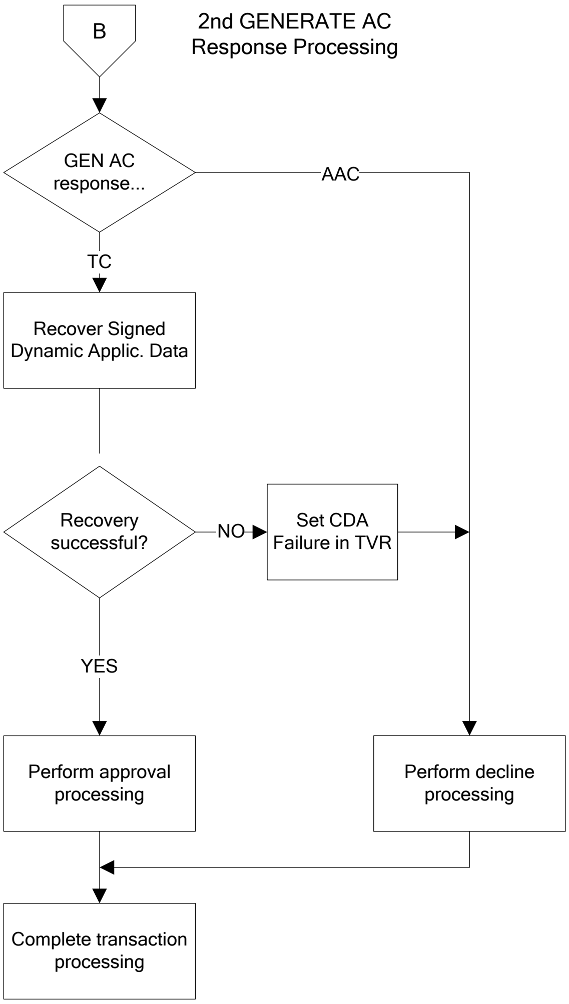


Figure 5: CDA Sample Flow Part 3 of 3

7 Personal Identification Number Encipherment

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

More precisely, the ICC shall own a public key pair associated with PIN encipherment. The public key is then used by the PIN pad or a secure component of the terminal (other than the PIN pad) to encipher the PIN, and the private key is used by the ICC to decipher the enciphered PIN for verification.

The PIN block used in the data field to be enciphered shall be 8 bytes as shown in section 6.5.12 of Book 3.

7.1 Keys and Certificates

If offline PIN encipherment is supported, the ICC shall own a unique public key pair consisting of a public encipherment key and the corresponding private decipherment key. This specification allows the following two possibilities.

1. The ICC owns a specific ICC PIN Encipherment Private and Public Key. The ICC PIN Encipherment Public Key shall be stored on the ICC in a public key certificate in exactly the same way as for the ICC Public Key for offline dynamic data authentication as specified in section 6.[31]

   The ICC PIN encipherment public key pair has an ICC PIN Encipherment Public Key Modulus of N_{PE} bytes, where $N_{PE} \leq N_I \leq N_{CA} \leq 248$, N_I being the length of the Issuer Public Key Modulus (see section 6.1). If $N_{PE} > (N_I - 42)$, the ICC PIN Encipherment Public Key Modulus is divided into two parts, one part consisting of the $N_I - 42$ most significant bytes of the modulus (the Leftmost Digits of the ICC PIN Encipherment Public Key) and a second part consisting of the remaining $N_{PE} - (N_I - 42)$ least significant bytes of the modulus (the ICC PIN Encipherment Public Key Remainder).

   The ICC PIN Encipherment Public Key Exponent shall be equal to 3 or $2^{16} + 1$.

   The ICC PIN Encipherment Public Key Certificate is obtained by applying the digital signature scheme as specified in Annex A2.1 on the data in Table 23 using the Issuer Private Key.

[31] In the case that the ICC owns a specific ICC PIN Encipherment Public Key, the format of the data recovered from the certificate is exactly the same as for dynamic data authentication (see Table 14 in section 6.4) however the data that is input to the hash algorithm when computing the certificate (see Table 23) does not include the Static Data to be Authenticated. Hence all the verification steps specified in section 6.4 are performed except in step 5 the static data to be authenticated is not included in the concatenation of data elements to be hashed in step 6.

Field Name	Length	Description	Format
Certificate Format	1	Hex Value '04'	b
Application PAN	10	PAN (padded to the right with Hex 'F's)	cn 20
Certificate Expiration Date	2	MMYY after which this certificate is invalid	n 4
Certificate Serial Number	3	Binary number unique to this certificate assigned by the issuer	b
Hash Algorithm Indicator	1	Identifies the hash algorithm used to produce the Hash Result in the digital signature scheme [32]	b
ICC PIN Encipherment Public Key Algorithm Indicator	1	Identifies the digital signature algorithm to be used with the ICC PIN Encipherment Public Key [32]	b
ICC PIN Encipherment Public Key Length	1	Identifies the length of the ICC PIN Encipherment Public Key Modulus in bytes	b
ICC PIN Encipherment Public Key Exponent Length	1	Identifies the length of the ICC PIN Encipherment Public Key Exponent in bytes	b
ICC PIN Encipherment Public Key or Leftmost Digits of the ICC PIN Encipherment Public Key	$N_I - 42$	If $N_{PE} \leq N_I - 42$, consists of the full ICC PIN Encipherment Public Key padded to the right with $N_I - 42 - N_{PE}$ bytes of value 'BB' If $N_{PE} > N_I - 42$, consists of the $N_I - 42$ most significant bytes of the ICC PIN Encipherment Public Key [33]	b
ICC PIN Encipherment Public Key Remainder	0 or $N_{PE} - N_I + 42$	Present only if $N_{PE} > N_I - 42$ and consists of the $N_{PE} - N_I + 42$ least significant bytes of the ICC PIN Encipherment Public Key	b
ICC PIN Encipherment Public Key Exponent	1 or 3	ICC PIN Encipherment Public Key Exponent equal to 3 or $2^{16} + 1$	b

Table 23: ICC PIN Encipherment Public Key Data to be Signed by Issuer (i.e. input to the hash algorithm)

[32] See Annex B for specific values assigned to approved algorithms.

[33] As can be seen in Annex A2.1, $N_I - 22$ bytes of the data signed are retrieved from the signature. Since the length of the first through the eighth data elements in Table 23 is 20 bytes, there are $N_I - 22 - 20 = N_I - 42$ bytes left for the data to be stored in the signature.

2. The ICC does not own a specific ICC PIN encipherment public key pair, but owns an ICC public key pair for offline dynamic data authentication as specified in section 6.1. This key pair can then be used for PIN encipherment. The ICC Public Key is stored on the ICC in a public key certificate as specified in section 6.1.

The first step of PIN encipherment shall be the retrieval of the public key to be used by the terminal for the encipherment of the PIN. This process takes place as follows.

1. If the terminal has obtained all the data objects specified in Table 24 from the ICC, then the terminal retrieves the ICC PIN Encipherment Public Key in exactly the same way as it retrieves the ICC Public Key for offline dynamic data authentication (see section 6).
2. If the terminal has not obtained all the data objects specified in Table 24, but has obtained all the data objects specified in Table 12, then the terminal retrieves the ICC Public Key as described in section 6.
3. If the conditions under points 1 and 2 above are not satisfied or if as described in Section 6.1.2 for dynamic data authentication, the Issuer Public Key Certificate has been revoked, then PIN encipherment has failed and the Offline Enciphered PIN CVM has failed.

Tag	Length	Value	Format
—	5	Registered Application Provider Identifier (RID)	b
'8F'	1	Certification Authority Public Key Index	b
'90'	N_{CA}	Issuer Public Key Certificate	b
'92'	$N_I – N_{CA} + 36$	Issuer Public Key Remainder, if present	b
'9F32'	1 or 3	Issuer Public Key Exponent	b
'9F2D'	N_I	ICC PIN Encipherment Public Key Certificate	b
'9F2E'	1 or 3	ICC PIN Encipherment Public Key Exponent	b
'9F2F'	$N_{PE} – N_I + 42$	ICC PIN Encipherment Public Key Remainder, if present	b

Table 24: Data Objects Required for Retrieval of ICC PIN Encipherment Public Key

7.2 PIN Encipherment and Verification

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

1. 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.
2. The terminal issues a GET CHALLENGE command to the ICC to obtain an 8-byte unpredictable number 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 CVM has failed.
3. The terminal generates random padding consisting of N – 17 bytes, where N is the length in bytes of the public key to be used for PIN encipherment retrieved as specified in section 7.1 (hence N = N_{PE} or N = N_{IC}). The value of the random padding shall be unpredictable from the perspective of an attacker (even given knowledge of previous values) and prior to encipherment shall only exist in hardware suitable for protecting the plaintext PIN. For each encryption all values should be equally likely to be generated. This may be achieved using a Random Number Generator compliant with ISO/IEC 18031 and tested using NIST SP 800-22A (see reference [4] in Annex C).
4. Using the PIN Encipherment Public Key or the ICC Public Key retrieved as specified in section 7.1, the terminal applies the RSA Recovery Function as specified in Annex B2.1.3 to the data specified in Table 25 in order to obtain the Enciphered PIN Data.

Field Name	Length	Description	Format
Data Header	1	Hex Value '7F'	b
PIN Block	8	PIN in PIN Block	b
ICC Unpredictable Number	8	Unpredictable number obtained from the ICC with the GET CHALLENGE command	b
Random Padding	N_{IC} – 17	Random Padding generated by the terminal	b

Table 25: Data to be Enciphered for PIN Encipherment

5. The terminal issues a VERIFY command including the Enciphered PIN Data obtained in the previous step.
6. With the ICC Private Key, the ICC applies the RSA Signing Function as specified in Annex B2.1.2 to the Enciphered PIN Data in order to recover the plaintext data specified in Table 25.

7. The ICC verifies whether the ICC Unpredictable Number recovered is equal to the ICC Unpredictable Number generated by the ICC with the GET CHALLENGE command. If this is not the case, Offline Enciphered PIN has failed.[34]
8. The ICC verifies whether the Data Header recovered is equal to '7F'. If this is not the case, Offline Enciphered PIN has failed.[34]
9. The ICC verifies whether the PIN included in the recovered PIN Block corresponds with the PIN stored in the ICC. If this is not the case, PIN verification has failed.[34]

If all the above steps were executed successfully, enciphered PIN verification was successful. The ordering of the steps 1 to 3 in Section 7.2 is representative, not mandatory. Key retrieval (as described in Section 7.1) and steps 1 to 3 in Section 7.2 can be conducted in any order, provided they are all complete before the terminal applies the RSA Recovery Function (as described in step 4 of Section 7.2).

[34] When recovery of the PIN Block has failed, the ICC should return SW1 SW2 = '6983' or '6984' as described in Book 3 section 10.5.1 to indicate that Offline Enciphered PIN processing has failed.

8 Application Cryptogram and Issuer Authentication

The aim of this section is to provide methods for the generation of the Application Cryptograms (TC, ARQC, or AAC) generated by the ICC and the Authorisation Response Cryptogram (ARPC) generated by the issuer and verified by the ICC. For more details on the role of these cryptograms in a transaction, see section 10.8 of Book 3.

Note that the methods provided in this specification are not mandatory. Issuers may decide to adopt other methods for these functions.

8.1 Application Cryptogram Generation

8.1.1 Data Selection

An Application Cryptogram consists of a Message Authentication Code (MAC) generated over data:

- referenced in the ICC's DOLs and transmitted from the terminal to the ICC in the GENERATE AC or other command, and
- accessed internally by the ICC.

The recommended minimum set of data elements to be included in Application Cryptogram generation is specified in Table 26.

Value	Source
Amount, Authorised (Numeric)	Terminal
Amount, Other (Numeric)	Terminal
Terminal Country Code	Terminal
Terminal Verification Results	Terminal
Transaction Currency Code	Terminal
Transaction Date	Terminal
Transaction Type	Terminal
Unpredictable Number	Terminal
Application Interchange Profile	ICC
Application Transaction Counter	ICC

Table 26: Recommended Minimum Set of Data Elements for Application Cryptogram Generation

8.1.2 Application Cryptogram Algorithm

The method for Application Cryptogram generation takes as input a unique ICC Application Cryptogram Master Key MK_{AC} and the data selected as described in section 8.1.1, and computes the 8-byte Application Cryptogram in the following two steps:

1. Use the session key derivation function specified in Annex A1.3 to derive an Application Cryptogram Session Key SK_{AC} from the ICC Application Cryptogram Master Key MK_{AC} and the 2-byte Application Transaction Counter (ATC) of the ICC.
2. Generate the 8-byte Application Cryptogram by applying the MAC algorithm specified in Annex A1.2 to the data selected and using the Application Cryptogram Session Key derived in the previous step. For AES the 8-byte Application Cryptogram is created by setting the parameter *s* to 8.

8.2 Issuer Authentication

Two methods are supported for generation of the ARPC used for issuer authentication. Each method can use either Triple DES or AES.

8.2.1 ARPC Method 1

ARPC Method 1 for the generation of an 8-byte ARPC using Triple DES consists of applying the Triple-DES algorithm as specified in Annex B1.1 to:

- the 8-byte ARQC as described in section 8.1[35]
- the 2-byte Authorisation Response Code (ARC)

using the Application Cryptogram Session Key SK_{AC} (see section 8.1) in the following way:

1. Pad the 2-byte ARC with six zero bytes to obtain the 8-byte number

$$X := (ARC \mid\mid '00' \mid\mid '00' \mid\mid '00' \mid\mid '00' \mid\mid '00' \mid\mid '00')$$

2. Compute $Y := ARQC \oplus X$.
3. The 8-byte ARPC is then obtained by

$$ARPC := DES3(SK_{AC})[Y]$$

[35] It is recommended that an ARPC is only returned if the ARQC verification is successful. However if an issuer still intends to return an ARPC when ARQC verification has failed, then the ARPC should not be calculated from the ARQC received from the network.

ARPC Method 1 for the generation of an 8-byte ARPC using AES is identical to the ARPC Method 1 using Triple DES (above) except that in step 3 the 8-byte ARPC is computed using AES as specified in Annex B1.2 and is defined to be the leftmost 8 bytes of:

$$AES(SK_{AC})[\ Y\ ||\ Y_0\ ]$$

where

$$Y_0 := ('00'\ ||\ '00'\ ||\ '00'\ ||\ '00'\ ||\ '00'\ ||\ '00'\ ||\ '00'\ ||\ '00').$$

8.2.2 ARPC Method 2

ARPC Method 2 for the generation of a 4-byte ARPC using Triple DES consists of applying the MAC algorithm as specified in Annex A1.2.1 to:

- the 8-byte ARQC as described in section 8.1[36]
- the 4-byte binary Card Status Update (CSU) [37]
- the 0-8 byte binary Proprietary Authentication Data

using the Application Cryptogram Session Key SK_{AC} (see section 8.1) in the following way:

1. Concatenate the ARQC, the CSU, and the Proprietary Authentication Data.

$$Y := ARQC\ ||\ CSU\ ||\ \text{Proprietary Authentication Data}$$

2. Generate a MAC over the data Y by applying the MAC algorithm specified in Annex A1.2 to the data defined above using the Application Cryptogram Session Key derived when computing the ARQC. For this application of the MAC algorithm, the MAC is computed according to ISO/IEC 9797-1 Algorithm 3, and the parameter *s* is set to 4, thereby yielding a 4-byte MAC.

$$ARPC := MAC := \text{MAC algorithm } (SK_{AC})[Y]$$

3. The Issuer Authentication Data (tag '91') is formed by concatenating the resulting 4-byte ARPC, the 4-byte CSU, and the Proprietary Authentication Data.

$$\text{Issuer Authentication Data} := ARPC\ ||\ CSU\ ||\ \text{Proprietary Authentication Data}$$

[36] It is recommended that an ARPC is only returned if the ARQC verification is successful. However if an issuer still intends to return an ARPC when ARQC verification has failed, then the ARPC should not be calculated from the ARQC received from the network.

[37] See Annex A of Book 3 for a definition of this data item.

ARPC Method 2 for the generation of a 4-byte ARPC using AES is identical to the ARPC Method 2 using Triple DES (above) except that in step 2 the MAC is computed using AES and the CMAC algorithm as specified in Annex A1.2.2 The parameter *s* is set to 4, thereby yielding a 4-byte MAC.

8.3 Key Management

The mechanisms for Application Cryptogram and Issuer Authentication require the management by the issuer of the unique ICC Application Cryptogram Master Keys. Annex A1.4 specifies two optional methods for the derivation of the ICC Application Cryptogram Master Keys from the Primary Account Number (PAN) and the PAN Sequence Number.

9 Secure Messaging

The objectives of secure messaging are to ensure data confidentiality, data integrity, and authentication of the sender. Data integrity and issuer authentication are achieved using a MAC. Data confidentiality is achieved using encipherment of the data field.

9.1 Secure Messaging Format

Secure messaging shall be according to one of the following two formats.

- **Format 1:** Secure messaging format according to ISO/IEC 7816-4, where the data field of the affected command uses Basic Encoding Rules-Tag Length Value (BER-TLV) encoding and encoding rules of ASN.1/ISO 8825-1 apply strictly. This is explicitly specified in the lowest significant nibble of the class byte of the command, which is set to 'C'. This also implies that the command header is always integrated in MAC calculation.
- **Format 2:** Secure messaging format where the data field of the affected command does not use BER-TLV encoding for secure messaging, but may use it for other purposes. In this case, the data objects contained in the data field and corresponding lengths of these data objects shall be known by the sender of a command using secure messaging and known by the currently selected application. In compliance with ISO/IEC 7816-4, secure messaging according to Format 2 is explicitly specified in the lowest significant nibble of the class byte of the command, which is set to '4'.

9.2 Secure Messaging for Integrity and Authentication

9.2.1 Command Data Field

9.2.1.1 Format 1

The data field of the secured command is composed of the following TLV data objects as shown in Figure 6.

If the command to be secured has command data, this command data is carried in the first data object[38] either as plaintext data or, if secure messaging for confidentiality is applied, as a cryptogram.

If the command data is carried as plaintext data then:

- If the unsecured command data is not BER-TLV encoded, then the data shall be encapsulated under tag '81'.
- If the unsecured command data is BER-TLV encoded and if the tag of any data element lies in the context specific class (range '80' to 'BF') reserved for SM-related data objects, then the command data shall be encapsulated in a constructed data object under tag 'B3'.
- If the unsecured command data is BER-TLV encoded and no tag lies in the context specific class (range '80' to 'BF') reserved for SM-related data objects, then ISO/IEC 7816-4 permits that the command data may be included without encapsulation. However if encapsulated then the command data shall be encapsulated in a constructed data object under tag 'B3'.

Note: If it is not always apparent that the data is BER-TLV encoded then the data may be encapsulated under tag '81'.

If the command data is carried as a cryptogram then it shall be encapsulated in a data object for confidentiality as described in section 9.3.1.1.

The second data object is the MAC. Its tag is '8E', and its length shall be in the range of four to eight bytes. Note that if the command to be secured has no command data (e.g. Application Block) then this MAC is the first and only data object of the secured command.

[38] EMV anticipates one data object preceding the MAC data object. Depending on the command data of the unsecured command there could be more than one such data object. For these constructions please refer to ISO/IEC 7816-4.

Tag 1	Length 1	Value 1	Tag 2	Length 2	Value 2
T	L	Value (L bytes)	'8E'	'04'–'08'	MAC (4–8 bytes)

Figure 6: Format 1 Command Data Field for Secure Messaging for Integrity and Authentication

An example is provided in Annex D2.

9.2.1.2 Format 2

The data elements (including the MAC) contained in the data field and the corresponding lengths shall be known by the sender of a command using secure messaging and known by the currently selected application. The MAC is not BER-TLV coded and shall always be the last data element in the data field and its length shall be in the range of 4 to 8 bytes (see Figure 7).

Value 1	Value 2
Command data (if present)	MAC (4-8 bytes)

Figure 7: Format 2 Command Data Field for Secure Messaging for Integrity and Authentication

9.2.2 MAC Session Key Derivation

The first step of the MAC generation for secure messaging for integrity consists of deriving a unique MAC Session Key from the ICC's unique MAC Master Key and card-controlled data unique to the transaction. A method to do this is specified in Annex A1.2.2.

9.2.3 MAC Computation

The MAC is computed by applying the mechanism described in Annex A1.2 with the MAC Session Key derived as described in section 9.2.2 to the message to be protected.

If secure messaging is according to Format 1, the message to be protected shall be constructed from the header of the command APDU (CLA INS P1 P2) and the command data (if present) according to the rules specified in ISO/IEC 7816-4.

Note that for Format 1 the rules specified in ISO/IEC 7816-4 already define padding, so the padding of the first step of the MAC computation defined in Annex A1.2 shall be omitted. Specifically, the message MSG used in the MAC calculation is padded after the command header (CLA INS P1 P2 with CLA set to indicate secure messaging) and also after the data object carrying the command data if present. This data object is either a plaintext data object or, if secure messaging for confidentiality is applied, a data object for confidentiality (see section 9.3.1.1). The padding in each situation consists of one mandatory byte of '80' added to the right and then the smallest number of '00' bytes is added to the right so that the length of the resulting string is a number of bytes equal to a multiple of the block size (e.g. 8 bytes for DES or Triple DES and 16 bytes for AES).

If secure messaging is according to Format 2, the message to be protected shall be constructed according to the payment system proprietary specifications. It shall however always contain the header of the command APDU and the command data (if present).

In all cases, if the MAC used for secure messaging has been specified as having a length *s*, then the MAC is obtained by taking the leftmost (most significant) *s* bytes from the result of the calculation described above.

9.2.3.1 Format 1 MAC Chaining

If secure messaging is according to Format 1 and chaining of MACs from one command to the next is supported, the recommended method for chaining the MACs is as follows:

A value whose length is equal to the block size of the cipher (e.g. 8 bytes for DES or Triple DES and 16 bytes for AES) is inserted at the beginning of the message to be protected.[39] This value is:

- for the first or only script command, the Application Cryptogram generated by the card for the first GENERATE AC command
  - for 8-byte block ciphers this is the 8-byte Application Cryptogram

[39] In the terms of ISO/IEC 7816-4 and in the case that Algorithm 3 of ISO/IEC 9797-1 is used with Single DES this is equivalent to using an auxiliary block in the initial stage where this auxiliary block is the single DES encryption of the Application Cryptogram or MAC of the preceding command.

  - for 16-byte block ciphers this is the 8-byte Application Cryptogram padded to the right with 8 bytes of hexadecimal zeros
    Application Cryptogram | | '00' | | '00' | | '00' | | '00' | | '00' | | '00' | | '00' | | '00'

- for subsequent script commands, the full cryptogram value generated by the MAC algorithm of the preceding script command (this is the full output prior to any truncation that occurs when shorter MACs are transmitted).

Note: Issuers should be aware that when multiple issuer scripts in a single response are supported, the failure of a command in one script may result in a gap in the MAC chain. This gap will cause MAC failures for commands in subsequent scripts.

9.3 Secure Messaging for Confidentiality

9.3.1 Command Data Field

9.3.1.1 Format 1

The format of a data object for confidentiality in the command data field of a secured command is shown in Figure 8.

Tag	Length	Value
T	L	Cryptogram (enciphered data field) or Padding Indicator Byte		Cryptogram (enciphered data field)

Figure 8: Format 1 - Data Object for Confidentiality

ISO/IEC 7816-4 specifies the tags which may be allocated to the cryptogram resulting from the encipherment of the data field of the unsecured command. An odd-numbered tag shall be used if the object is to be integrated in the computation of a MAC; an even-numbered tag shall be used otherwise.

If tag '86' or '87' is allocated to the data object for confidentiality, the value field of the data object for confidentiality contains the padding indicator byte followed by the cryptogram. The padding indicator byte shall be encoded according to ISO/IEC 7816-4. If another tag is used, the value field of the data object for confidentiality contains the cryptogram only.

An example is provided in Annex D2.

9.3.1.2 Format 2

Data encipherment is applied to the full plaintext command data field with the exception of a MAC (see Figure 9).

Value1	Value2
Cryptogram (enciphered data)	MAC (if present)

Figure 9: Format 2 Command Data Field for Secure Messaging for Confidentiality

9.3.2 Encipherment Session Key Derivation

The first step of the encipherment/decipherment for secure messaging for confidentiality consists of deriving a unique Encipherment Session Key from the ICC's unique Encipherment Master Key and card-controlled data unique to the transaction. A method to do this is specified in Annex A1.3.

9.3.3 Encipherment/Decipherment

Encipherment/decipherment of the plain/enciphered command data field takes place according to the mechanism described in Annex A1.1 with the Encipherment Session Key derived as described in section 9.3.2.

9.4 Key Management

The secure messaging mechanisms require the management by the issuer of the unique ICC MAC and Encipherment Master Keys. Annex A1.4 specifies methods for the derivation of the ICC MAC and Encipherment Master Keys from the Primary Account Number (PAN) and the PAN Sequence Number.

10 Certification Authority Public Key Management Principles and Policies

This section defines a framework for principles and policies for a payment system for the management of the Certification Authority Public Keys used for offline static and dynamic data authentication as specified in this specification.

Principles are concepts identified as the basis for implementing Certification Authority Public Key management. These principles can give rise to policies that may be shared across the payment systems, or policies that are adopted by individual payment systems. Each payment system will develop its own set of procedures to implement these policies.

10.1 Certification Authority Public Key Life Cycle

10.1.1 Normal Certification Authority Public Key Life Cycle

The life cycle of a Certification Authority Public Key in normal circumstances can be divided into the following consecutive phases:

- Planning
- Generation
- Distribution
- Key Usage
- Revocation (Scheduled)

10.1.1.1 Planning

During the planning phase, the payment system investigates the requirements for the introduction of new Certification Authority Public Key pairs in the near future. These requirements are related to the number of keys required and the parameters of these keys.

An important part of the planning phase is the security review to determine the life expectancy of existing and potential new keys. This review is to lead to the setting of lengths and expiration dates for new keys and the potential modification of the expiration dates of existing keys, and a roll-out schedule of replacement keys.

10.1.1.2 Generation

If the results of the planning phase require the introduction of new Certification Authority Public Key pairs, these will be generated by the payment system. More precisely, the payment system certification authority (a physically and logically highly secured infrastructure operated by the payment system) will generate in a secure way the necessary Certification Authority Private/Public Key pairs for further use.

Subsequent to generation the secrecy of the Certification Authority Private Keys will be maintained, and the integrity of both Certification Authority Public and Private Keys will also be maintained.

10.1.1.3 Distribution

In the key distribution phase, the payment system certification authority will distribute newly generated Certification Authority Public Keys to its member Issuers and Acquirers for the following purposes (see Figure 10):

- To issuers, to verify Issuer Public Key Certificates supplied by the payment system certification authority during the key usage phase (see section 10.1.1.4).
- To acquirers, for secure loading of the Certification Authority Public Keys in its merchant terminals.

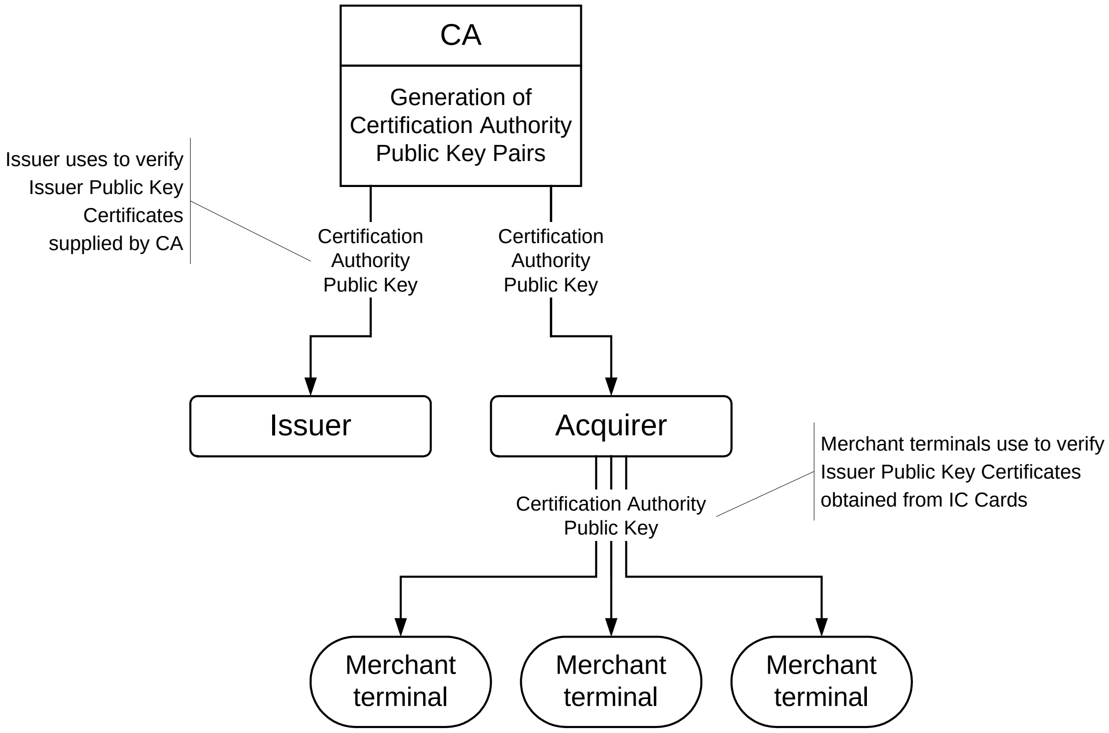

CA
Generation of Certification Authority Public Key Pairs
Issuer uses to verify Issuer Public Key Certificates supplied by CA
Certification Authority Public Key
Certification Authority Public Key
Issuer
Acquirer
Merchant terminals use to verify Issuer Public Key Certificates obtained from IC Cards
Certification Authority Public Key
Merchant terminal
Merchant terminal
Merchant terminal

Figure 10: Certification Authority Public Key Distribution

In order to prevent the introduction of fraudulent Certification Authority Public Keys, the interfaces between the payment system certification authority and the issuers and acquirers need to ensure the integrity of the Certification Authority Public Keys distributed.

10.1.1.4 Key Usage

The Certification Authority Public Key is used in the merchant terminals to perform offline static or dynamic data authentication as specified in sections 5 and 6 and to perform Offline Enciphered PIN processing (as specified in section 7).

The Certification Authority Private Key is used by the payment system certification authority for the generation of the Issuer Public Key Certificates. More precisely, the following interactions take place (see Figure 11):

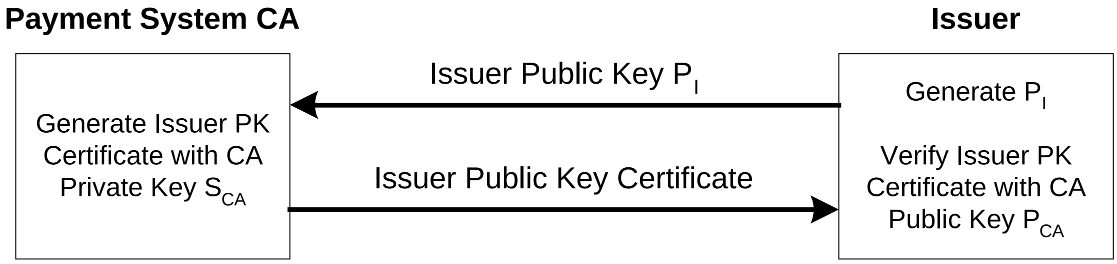

Payment System CA
Issuer
Generate Issuer PK Certificate with CA Private Key S_{CA}
Issuer Public Key P_I
Issuer Public Key Certificate
Generate P_I
Verify Issuer PK Certificate with CA Public Key P_{CA}

Figure 11: Issuer Public Key Distribution

- The issuer generates its Issuer Public Key and sends it to the payment system certification authority.
- The payment system certification authority signs the Issuer Public Key with the Certification Authority Private Key to obtain the Issuer Public Key Certificate that is returned to the issuer.
- With the Certification Authority Public Key, the issuer verifies the correctness of the received Issuer Public Key Certificate. If it is correct, the issuer can then include it as part of the personalisation data for its IC Cards.

In order to prevent the introduction of fraudulent Issuer Public Keys, the interfaces between the issuer and the payment system certification authority need to ensure the integrity of the Issuer Public Keys submitted for certification.

10.1.1.5 Revocation (Scheduled)

Once a Certification Authority Public Key pair has reached its planned expiration date set during the planning phase, it must be removed from service. Practically speaking, this means the following.

- As of that expiration date, Issuer Public Key Certificates produced with the Certification Authority Private Key will no longer be valid. Issuers should therefore ensure that IC Cards personalised with such Issuer Public Key Certificates expire no later than the expiration date of the Certification Authority Public Key pair.
- An appropriate time prior to that expiration date, the payment system certification authority will stop signing Issuer Public Keys with the corresponding Certification Authority Private Key.
- As of that expiration date, acquirers need to remove the Certification Authority Public Keys from service in their terminals within a specific grace period after expiration.

10.1.2Certification Authority Public Key Pair Compromise

In the event of a Certification Authority Public Key pair compromise, an emergency process needs to be put in place that in the end may lead to the accelerated revocation of the Certification Authority Public Key pair before its planned expiration. In this case, there are additional phases in the key life cycle:

- Detection
- Assessment
- Decision
- Revocation (Accelerated)

These phases are described below.

10.1.2.1 Detection

The compromise of a Certification Authority Public Key pair can be either:

- Actual: For example a confirmed security breach at the payment system certification authority, or a confirmed breaking of the key by cryptanalysis.
- Suspected: System monitoring or member and cardholder complaint indicates that fraudulent transactions have occurred which could be due to key compromise, but this is not confirmed.
- Potential: Cryptanalytic techniques, for example factorisation, have developed such that with resources available any key of a given length could be compromised, but there is no evidence that this has occurred.

Detection of a key compromise may vary from awareness of an actual physical break-in of the payment system certification authority, through the reporting of fraudulent off-line transactions by the fraud and risk management systems put in place by the payment system and its members, to intelligence on factorisation advances gathered from the cryptographic community.

10.1.2.2 Assessment

The assessment of a (potential) Certification Authority Public Key pair compromise will include technical, risk and fraud, and, most importantly, business impacts for the payment system and its members. The results of the assessment will include the confirmation of the compromise, the determination of possible courses of action against costs and risk of the compromise, and presenting results of the assessment to support a decision.

10.1.2.3 Decision

Based on the results of the assessment phase, the payment system will decide on a course of action that will be taken for a key compromise. In the worst case, this decision will consist of the actual unplanned revocation of a Certification Authority Public Key before its planned expiration date.

10.1.2.4 Revocation (Accelerated)

The decision to revoke a Certification Authority Public Key will lead to the communication to the payment system members of a new expiration date of that key. The process after that is the same as for the planned revocation described in section 10.1.1.5.

10.2 Principles and Policies by Phase

10.2.1 General Principles

- Support of Certification Authority Public Key revocation is a requirement for each payment system's IC Card credit and debit products.
- Payment systems will align policies, procedures, and schedules for Certification Authority Public Key revocation where practical.
- EMVCo will use a common definition of the phases of the Certification Authority Public Key revocation process and a common terminology in internal and member communications.
- Each payment system operates as a closed system with regard to any legal requirements relative to Certification Authority Public Key pairs.

10.2.2 Planning Phase

10.2.2.1 Phase Definition

The Planning phase involves review and planning of Certification Authority Public Key pairs. Existing keys are reviewed for resistance to attack, and new key planning is undertaken. Length and expiration dates of existing and new keys are reviewed by risk and cryptography experts to confirm that the key life expectancy is considered secure. Lengths of new keys are determined, and a rollout schedule of replacement keys is maintained.

10.2.2.2 Principles

- Key sizes should reflect maximum feasible security consistent with terminal capability and POS operational timing.
- Payment systems should synchronize the expiration date of keys of a particular length where practical. Final decision authority for key revocation rests with each payment system.
- In the event of announcement of an accelerated revocation by a member of EMVCo, the member may request an EMVCo planning session be convened to address the revocation, the key compromise, and its impacts.

10.2.2.3 Shared Policies

- 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. A member of EMVCo may request an emergency meeting for key review at any time.

- EMVCo will prepare "best information" estimates of relative key strength for existing key lengths based on current evaluation criteria, and will make recommendations for rollout of new key lengths.
- The recommendations of this review process will be circulated to the payment systems, which will use them to set their individual policies. Each payment system will identify areas where payment system differentiation is required.
- Payment systems will use EMVCo recommendations as a factor in determining policy on number and length of live keys, exponent value, expiry date, and planned revocation schedule. Payment systems will publish these details to their members within 90 days of receipt of EMVCo recommendations.
- Key introduction and revocation will normally be on a planned, scheduled basis, but can be accelerated based on results of key life review.
- All Certification Authority Public Keys will have December 31st as planned expiration date.
- Acquirers have a six month grace period starting from the planned expiration date (until June 30th of the following calendar year) to withdraw an expired key from all terminals. Enforcement of key withdrawal is not expected to occur until after the end of the grace period and may be deferred at payment system discretion.
- All new Certification Authority Public Keys will be distributed prior to December 31st.
- Acquirers have a six month grace period (until June 30th of the following calendar year) to install any new keys in all terminals. Whenever possible the new keys will be distributed well in advance of December 31st, thereby giving a longer period for key installation.
- Payment systems will not enable the new keys to be used for valid transactions until January 1st of the following year.
- In the event of an accelerated revocation, a six-month grace period will similarly be maintained for key withdrawal in all terminals, but the fixed date of December 31st is not applicable.
- Notification to members and timing for any key revocation is the responsibility of each payment system.

10.2.3Generation Phase

10.2.3.1 Phase Definition

Key generation is the process of a payment system generating a Certification Authority Public Key pair.

10.2.3.2 EMV Principles

- Certification Authority Public Key pairs shall be generated in a secure environment according to accepted industry best practice.
- Within each RID, the Certification Authority Public Key Index is a unique value pointing to a particular Certification Authority Public Key pair. The value of a Certification Authority Public Key Index for a specific key shall not be changed.

10.2.3.3 Shared Payment System Policies

None Identified.

10.2.4Distribution Phase

10.2.4.1 Phase Definition

Key distribution is the process of circulating the public component of a Certification Authority Public Key Pair to get it into the marketplace. Certification Authority Public Keys will ultimately appear in merchant terminals. Certification Authority Private Keys will be used to produce Issuer Public Key Certificates, and are to be kept in the secure environment of the payment system certification authority.

10.2.4.2 EMV Principles

- Key distribution will ensure key integrity and origin authenticity.

10.2.4.3 Shared Payment System Policies

- Payment systems will support distribution of their public keys from the certification authority to acquirers and issuers via physical and/or electronic means.
- All new Certification Authority Public Keys will be distributed for receipt by recipients before December 31st.
- Payment systems will include a method allowing a recipient to validate a received public key, regardless of method of transmission.
- Certification Authority Public Keys will be distributed to acquirers with adequate lead time to allow installation in terminals before the corresponding private key is used to sign Issuer Public Keys.
- Certification Authority Public Keys will be distributed to issuers so that they may validate the Issuer Public Key Certificates produced by the certification authority.
- Each payment system certification authority will ensure that it does not distribute more than the maximum number (six) of keys that can be stored per RID in a terminal (see section 10.2.5).

10.2.5Key Usage Phase

10.2.5.1 Phase Definition

This phase is concerned with the normal day-to-day use of the Certification Authority Public Key pairs. Copies of the Certification Authority Public Keys will be used by terminals to perform offline static or dynamic data authentication or offline PIN encipherment during transactions with the appropriate payment system branded cards. The Certification Authority Private Keys will be held in the payment system certification authority and used to sign Issuer Public Keys, creating Issuer Public Key Certificates which the issuer will personalize onto its cards.

10.2.5.2 EMV Principles

- Terminals that support offline static or dynamic data authentication or offline PIN encipherment shall provide support for six Certification Authority Public Keys per RID for EMVCo member debit/credit applications based on this specification. Terminals shall support keys up to 1984 bits (248 bytes) in length, as specified in this specification.
- Terminals shall support the ability to install a Certification Authority Public Key, and the ability to withdraw a key from service as of a given date.
- Terminals shall provide the ability to validate Certification Authority Public Key integrity and should do so periodically.
- Payment systems will be responsible for ensuring the security of their Certification Authority Public Key pairs.

10.2.5.3 Shared Payment System Policies

- Payment systems will validate the integrity and origin of Issuer Public Keys prior to issuing a certificate.
- A payment system certification authority will begin using the private component of a Certification Authority Public Key pair no sooner than 6 months after the distribution of that key to acquirers.
- The expiry date of any issued IC Card shall be no later than the expiry date of the Issuer Public Key Certificate on that IC Card, and shall be no later than the published (at the time of card issuance) revocation date of the Certification Authority Public Key pair used to produce the Issuer Public Key Certificate.
- The expiry date of an Issuer Public Key Certificate shall be no later than the published (at the time of certificate issuance) revocation date of the Certification Authority Public Key pair used to produce the Issuer Public Key Certificate.
- The expiry date of an IC Card Public Key Certificate shall be no later than the expiry date of the Issuer Public Key used to produce the IC Card Public Key Certificate.

10.2.6Detection Phase

10.2.6.1 Phase Definition

Detection is the process that enables an entity to recognize that a Certification Authority Public Key pair has been, or is suspected of being compromised. There are multiple types of physical and logical compromise, including suspected, potential, and actual.

10.2.6.2 EMV Principles

- EMVCo will provide a forum for members of EMVCo to share evaluation of cryptanalytic advances that might lead to potential compromise of the digital signature scheme specified in this specification.
- Monitoring of key integrity and detection of suspected or potential Certification Authority Public Key pair compromise is the responsibility of each payment system.

10.2.6.3 Shared Payment System Policies

- Members shall notify a payment system of conditions or transactions that indicate possible or suspected compromise of a specific Certification Authority Public Key pair from that payment system.

10.2.7 Assessment Phase

NOTE: This phase applies only to accelerated revocations.

10.2.7.1 Phase Definition

If a Certification Authority Public Key compromise is detected or suspected, the owning payment system will assess the impact to business operations. Assessment includes confirming the compromise, determining possible courses of action, evaluating the cost of action against costs and risk of the compromise, and presenting results of the assessment to support a decision.

10.2.7.2 EMV Principles

- Assessment of suspected or potential Certification Authority Public Key pair compromise is the responsibility of each payment system.
- Payment systems will develop assessment policies and procedures that follow generally accepted best practices in risk management.
- There are different levels of compromise requiring different sets of actions depending on the compromise and a business assessment.

10.2.7.3 Shared Payment System Policies

- Payment system assessment will include actual and reputational costs to the payment system and to members. Potential courses of action will include an assessment of member and marketplace impact.

10.2.8 Decision Phase

NOTE: This phase applies only to accelerated revocations.

10.2.8.1 Phase Definition

As a result of the assessment phase, a payment system decides on a course of action that will be taken for a Certification Authority Public Key pair compromise.

10.2.8.2 EMV Principles

- The decision to revoke a specific Certification Authority Public Key Pair is at the sole discretion of the payment system that operates the certification authority for that key.
- Payment systems will develop and publish to their members a set of policies and procedures that detail the decision-making process for accelerated key revocation. These policies will include a method of notification to all affected issuers and acquirers.

10.2.8.3 Shared Payment System Policies

None identified.

10.2.9Revocation Phase

10.2.9.1 Phase Definition

Revocation is the key management process of withdrawing a key from service and dealing with the legacy of its use. Key revocation can be on schedule or accelerated. In the case of Certification Authority Public Key pairs, revocation means that the private key is no longer used to produce Issuer Public Key Certificates and that copies of the public key are withdrawn from service in terminals. Issuer Public Key Certificates signed with the private key are (as of a specific date) no longer valid in circulation on IC Cards.

10.2.9.2 EMV Principles

- Certification Authority Public Key revocation will be according to a previously published schedule unless a payment system has detected an imminent threat to product security. All scheduled revocations will conform to the "revocation window" dates developed by EMVCo.
- In case of an accelerated revocation, payment systems will take member impact into account, including terminal access, card re-issuance, and increased network traffic. Lead times for member activities shall be the same as during a scheduled revocation.

10.2.9.3 Shared Payment System Policies

- Revocation policies and procedures will be the same as for scheduled and accelerated revocations, wherever practical.
- All Certification Authority Public Keys will have December 31st as their planned expiration date. Acquirers shall have a six month grace period (until June 30th of the following calendar year) to withdraw the revoked key. Enforcement of key withdrawal is not expected to occur until after the end of the grace period and may be deferred at payment system discretion.
- Revocation of a Certification Authority Public Key pair requires that the public key component is withdrawn from service in all terminals within a six-month timeframe, consistent with payment system rules.
- In the case of an accelerated revocation, the introduction and withdrawal lead times will be the same as for scheduled revocations, however, the revocation date will be determined at the discretion of the payment system.

10.3 Sample Timelines

The following diagrams present sample timelines for the revocation and introduction of Certification Authority Public Keys, based on the principles and policies detailed in this Book. Each timeline represents a scheduled key introduction or withdrawal. In the case of an accelerated introduction or withdrawal, lead times for tasks would remain the same, but the month of the actual key introduction date and key revocation would be at the discretion of the payment system.

10.3.1Key Introduction

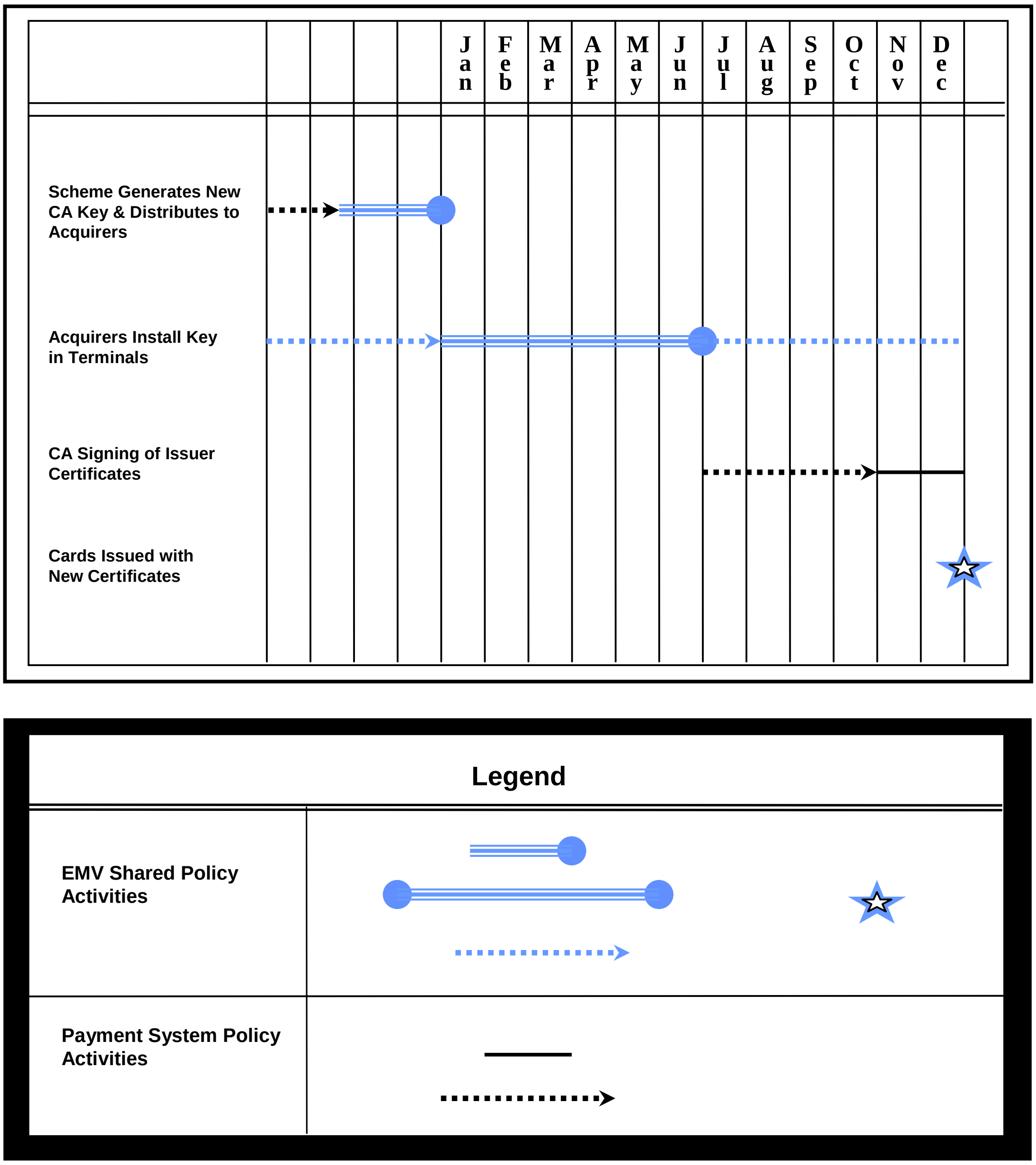


Figure 12: Key Introduction Example Timeline

10.3.2Key Withdrawal

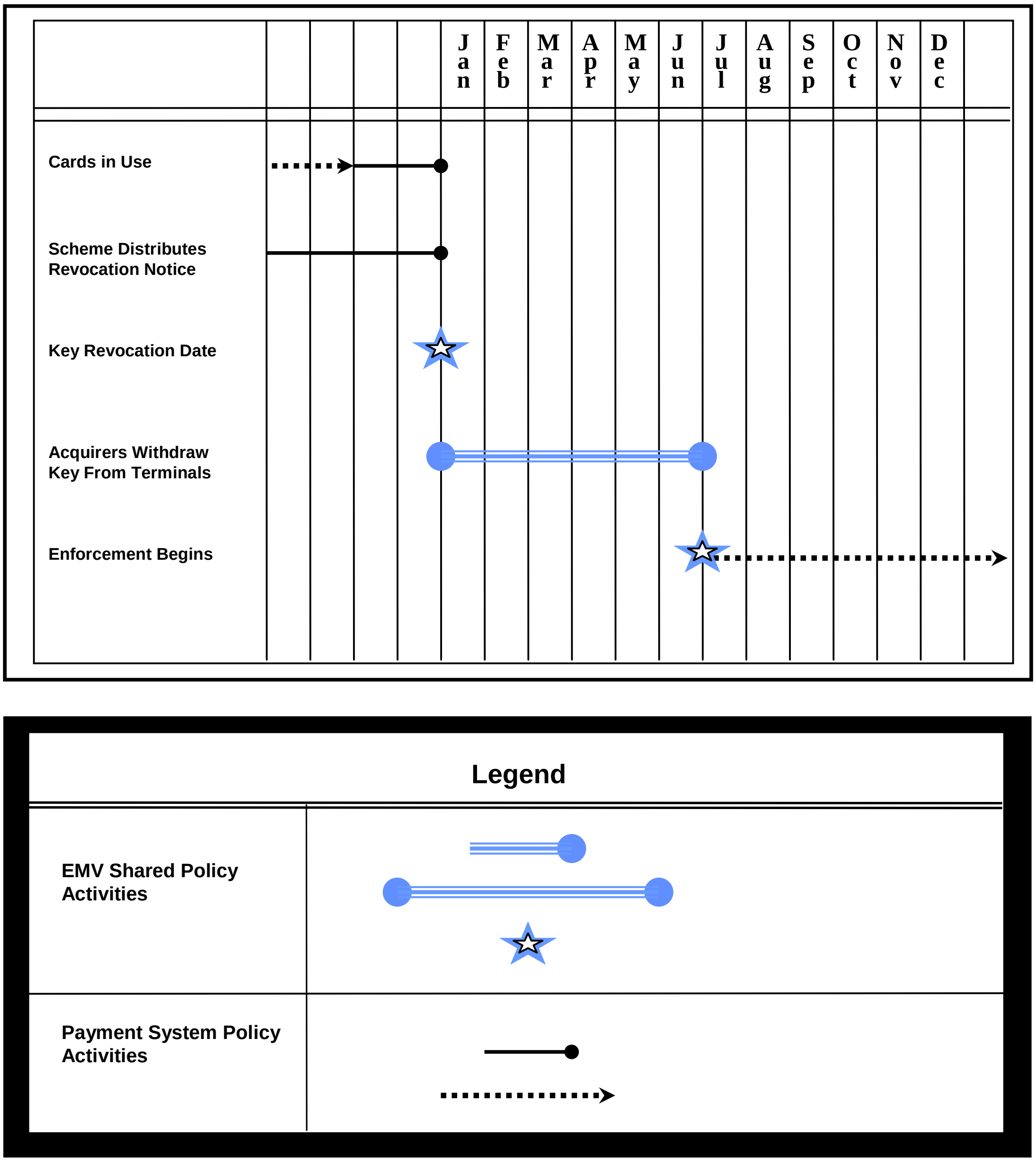


Figure 13: Key Withdrawal Example Timeline

11 Terminal Security and Key Management Requirements

This section describes the general terminal requirements for handling sensitive data, such as cryptographic keys. More specifically, it addresses key management requirements for Certification Authority Public Keys. Security requirements for PIN pads are addressed by individual payment systems.

11.1 Security Requirements for PIN Pads

A PIN pad shall be a tamper evident device and shall support entry of a 4-12 digit PIN. Security requirements for PIN pads are addressed by individual payment systems. For general PIN management and security principles, refer to ISO 9564.

11.2 Key Management Requirements

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

11.2.1Certification Authority Public Key Introduction

When a payment system has decided that 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 to each acquirer. 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 its terminals.

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.

11.2.2 Certification Authority Public Key Storage

Terminals that support offline static or dynamic data authentication shall provide support for six Certification Authority Public Keys per RID for EMVCo member debit/credit 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.

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

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.

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 should be verified periodically.

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 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	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 [40]	20	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	b

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

11.2.3Certification Authority Public Key Usage

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

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

11.2.4Certification Authority Public Key Withdrawal

When a payment system has decided to 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 dates coded into the terminals but instead remove keys according to the expiry 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.
- The terminal shall be able to verify that the received withdrawal notification originated from its legitimate acquirer.
- 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.

Part III

Annexes

Annex A Security Mechanisms

A1 Symmetric Mechanisms

A1.1 Encipherment

Encipherment of data uses an *n*-byte block cipher ALG either in Electronic Codebook (ECB) Mode or in Cipher Block Chaining (CBC) mode according to ISO/IEC 10116.

Encipherment of a message MSG of arbitrary length with Encipherment Session Key K_S takes place in the following steps.

1. Padding and Blocking

   - If the message MSG has a length that is not a multiple of *n* bytes, add one '80' byte to the right of MSG, and then add the smallest number of '00' bytes to the right such that the length of resulting message $\underline{\text{MSG}}$:= (MSG | | '80' | | '00' | | '00' | | . . . | | '00') is a multiple of *n* bytes.
   - If the message MSG has a length that is a multiple of *n* bytes, the following two cases can occur depending on pre-defined rules.
     - No padding takes place: $\underline{\text{MSG}}$:= MSG.
     - MSG is padded to the right with the *n*-byte block

       ('80' | | '00' | | '00' | | ... | | '00' | | '00' | | '00' | | '00')

       to obtain $\underline{\text{MSG}}$.

   $\underline{\text{MSG}}$ is then divided into *n*-byte blocks X_1, X_2, . . . , X_B.

2. Cryptogram Computation

   ECB Mode

   Encipher the blocks $X_1, X_2, \ldots, X_B$ into the *n*-byte blocks $Y_1, Y_2, \ldots, Y_B$ with the block cipher algorithm in ECB mode using the Encipherment Session Key K_S. Hence compute for i = 1, 2, . . . , B:

   $$Y_i := ALG(K_S)[X_i]$$

   CBC Mode

   Encipher the blocks $X_1, X_2, \ldots, X_B$ into the *n*-byte blocks $Y_1, Y_2, \ldots, Y_B$ with the block cipher algorithm in CBC mode using the Encipherment Session Key K_S. Hence compute for i = 1, 2, . . . , B:

   $$Y_i := ALG(K_S)[X_i \oplus Y_{i-1}]$$

   with *n*-byte initial value

   Y_0 := ('00' | | '00' | | '00' | | ... | | '00' | | '00' | | '00' | | '00').

   Notation:

   $$Y := (Y_1 \mid\mid Y_2 \mid\mid \ldots \mid\mid Y_B) = ENC(K_S)[MSG]$$

Decipherment is as follows.

1. Cryptogram Decipherment

   ECB Mode

   Compute for i = 1, 2, . . . , B:

   $$X_i := ALG^{-1}(K_S)[Y_i\,]$$

   CBC Mode

   Compute for i = 1, 2, . . . , B:

   $$X_i := ALG^{-1}(K_S)[Y_i\,] \oplus Y_{i-1}$$

   with *n*-byte initial value

   Y_0 := ('00' | | '00' | | '00' | | ... | | '00' | | '00' | | '00' | | '00').

2. To obtain the original message MSG, concatenate the blocks $X_1, X_2, \ldots, X_B$ and if padding has been used (see above) remove the trailing

   ('80' | | '00' | | '00' | | . . . | | '00') byte-string from the last block X_B.

   Notation:

   $$MSG = DEC(K_S)[Y]$$

A1.2 Message Authentication Code

A1.2.1 MAC Algorithms using an 8-byte block cipher

The computation of an s-byte MAC ($4 \leq s \leq 8$) is according to ISO/IEC 9797-1 using an 8-byte (64-bit) block cipher algorithm (ALG) such as Triple DES in CBC mode. More precisely, the computation of a MAC S over a message MSG consisting of an arbitrary number of bytes with a MAC Session Key K_S takes place in the following steps.

1. Padding and Blocking

   Pad the message MSG according to ISO/IEC 7816-4 (which is equivalent to method 2 of ISO/IEC 9797-1); hence add a mandatory '80' byte to the right of MSG, and then add the smallest number of '00' bytes to the right such that the length of resulting message

   $\underline{MSG}$:= (MSG || '80' || '00' || '00' || . . . || '00') is a multiple of 8 bytes.

   $\underline{MSG}$ is then divided into 8-byte blocks $X_1, X_2, \ldots, X_B$.

2. MAC Session Key

   The MAC Session Key K_S consists of either only a leftmost key block $K_S = K_{SL}$ or the concatenation of a leftmost and a rightmost key block $K_S = (K_{SL} \| K_{SR})$.

3. Cryptogram Computation

   Process the 8-byte blocks $X_1, X_2, \ldots, X_B$ with the block cipher in CBC mode using the leftmost MAC Session Key block K_{SL}:

   $$H_i := ALG(K_{SL})[X_i \oplus H_{i-1}], \text{ for } i = 1, 2, \ldots, B$$

   with initial value

   H_0 := ('00' || '00' || '00' || '00' || '00' || '00' || '00' || '00').[41]

   Compute the 8 byte block H_{B+1} in one of the following two ways.

   - According to ISO/IEC 9797-1 Algorithm 1:

     $$H_{B+1} := H_B$$

   - According to ISO/IEC 9797-1 Algorithm 3:

     $$H_{B+1} := ALG(K_{SL})[ALG^{-1}(K_{SR})[H_B]]$$

   The MAC S is then equal to the s most significant bytes of H_{B+1}.

[41] Note that pre-pending the MSG with the previous MAC (8 bytes) as a chaining block (see section 9.2.3) is equivalent to using an initial value equal to the previous MAC processed by Triple DES (Algorithm 1) or Single DES (Algorithm 3).

A1.2.2 MAC Algorithms using a 16-byte block cipher

The computation of an s-byte MAC ($4 \leq s \leq 8$) is according to ISO/IEC 9797-1:2011 Algorithm 5 (CMAC) using a 128-bit (16-byte) block cipher algorithm (ALG) such as AES in CBC mode. More precisely, the computation of a MAC S over a message MSG consisting of an arbitrary number of bytes with a MAC Session Key K_S takes place in the following steps.

1. Padding and Blocking

   If the length of the message MSG is a multiple of 16-bytes then no padding is added.[42]

   If the length of the message MSG is not a multiple of 16-bytes then add a mandatory '80' padding byte to the right of MSG, and then add the smallest number of '00' bytes to the right such that the length of resulting message

   MSG := (MSG || '80' || '00' || '00' || . . . || '00') is a multiple of 16 bytes.

   MSG is then divided into 16-byte blocks X_1, X_2, . . . , X_B.

2. MAC Session Key

   The MAC Session Key K_S consists of a k-bit block (see Annex B1).

   In accordance with ISO/IEC 9797-1 Algorithm 5 (CMAC), two k-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 k-bit string comprises the k-1 rightmost bits of the old string, appended with a zero bit)

   "msb" means the most significant bit (i.e. the leftmost bit)

   Note that all intermediate values must be kept secret (e.g. not used by the card as an unpredictable number).

3. Cryptogram Computation

   Mask the final block with sub-key K_1 if no padding was added

[42] Note that this will always be the case with Format 1 Secure Messaging as the message is padded prior to MACing.

$X_B := X_B \oplus K_1$

and with sub-key K_2 if padding was added

$X_B := X_B \oplus K_2$.

Process the 16-byte blocks $X_1, X_2, \ldots, X_B$ with the block cipher in CBC mode using the MAC Session Key K_S:

$H_i := ALG(K_S)[X_i \oplus H_{i-1}]$, for i = 1, 2, . . . , B

with 16-byte initial value

$H_0 :=$ ('00' | | '00' | | ... | | '00' | | '00' | | '00' | | '00' | | '00').[43]

The MAC S is then equal to the *s* most significant bytes of H_B.

A1.3 Session Key Derivation

Session keys K_S are derived from unique Master Keys K_M using diversification data R as follows

$$K_S := F(K_M)[R]$$

To prevent replay attacks, the diversification data R should have a high probability of being different for each session key derivation.

An important requirement for the diversification function F is that the number of possible outputs of the function is sufficiently large and uniformly distributed to prevent an exhaustive key search on the session key.

Annex A1.3.1 specifies a method for the derivation of session keys for Application Cryptogram generation, issuer authentication, and secure messaging (see sections 8 and 9) from ICC Master Keys. This session key derivation method ensures that different transactions use different session keys.

Note that the session key derivation method provided in Annex A1.3.1 is not mandatory. Issuers may decide to adopt another method for this function.

[43] Note that pre-pending the MSG with the previous MAC (16 bytes) as a chaining block (see section 9.2.3) is equivalent to using an initial value equal to the previous MAC

A1.3.1 Common Session Key Derivation Option

The common session key derivation option generates a unique session key for each transaction performed by the application. It does this by enciphering an *n*-byte diversification value with the *k*-bit ICC Master Key (MK) to produce a *k*-bit ICC Session Key (SK) using an *n*-byte block cipher algorithm ALG in ECB mode.

The *n*-byte diversification value is represented as

$$R = R_0 \mid\mid R_1 \mid\mid R_2 \mid\mid \ldots \mid\mid R_{n-1} .$$

For the session key used to generate and verify the Application Cryptogram and the ARPC, the diversification value is the ATC followed by *n*-2 bytes of '00':

$$R := \text{ATC} \mid\mid \text{'00'} \mid\mid \text{'00'} \mid\mid \ldots \mid\mid \text{'00'} \mid\mid \text{'00'} \mid\mid \text{'00'}.$$

For the session keys used for secure messaging, the diversification value R is the Application Cryptogram returned in the response to the first GENERATE AC command followed by n-8 bytes of '00':

$$R := \text{Application Cryptogram} \ldots \mid\mid \text{'00'} \mid\mid \text{'00'} \mid\mid \text{'00'}.$$

For an *n*-byte block cipher ALG using a *k*-bit key where $k = 8n$ (AES with k=128) the derivation function F is computed as follows:

$$SK := \text{ALG (MK) [R]}.$$

For an *n*-byte block cipher ALG using a *k*-bit key where $16n \geq k > 8n$ (Triple DES with k=128 or AES with k=192 or 256) R is used to create two *n*-byte blocks as follows:

$$F1 = R_0 \mid\mid R_1 \mid\mid \text{'F0'} \mid\mid \ldots \mid\mid R_{n-1}.$$

$$F2 = R_0 \mid\mid R_1 \mid\mid \text{'0F'} \mid\mid \ldots \mid\mid R_{n-1}.$$

and

SK := Leftmost *k*-bits of {ALG (MK) [F1] | | ALG (MK) [F2] }.

The same session key is used for all commands in a single transaction.

A1.4 Master Key Derivation

This annex specifies three optional methods for the derivation by the issuer of a *k*-bit ICC Master Key used for Application Cryptogram generation, issuer authentication, and secure messaging. The first two methods use the 8-byte block cipher Triple DES in ECB mode and the third method uses the 16-byte block cipher AES in ECB mode. The AES method supports 128-bit, 192-bit and 256-bit keys.

Note that none of these methods are mandatory. Issuers may decide to adopt alternative methods for this function.

These methods take as input the PAN and PAN Sequence Number, plus a *k*-bit Issuer Master Key IMK, and produce the *k*-bit ICC Master Key MK in the following way:

A1.4.1 Option A

Option A is only applicable when the block cipher is Triple DES.

1. Concatenate from left to right the decimal digits of the Application PAN with the PAN Sequence Number (if the PAN Sequence Number is not present, then it is replaced by a '00' byte). If the result X is less than 16 digits long, pad it to the left with hexadecimal zeros in order to obtain an 8-byte number Y in numeric format. If X is at least 16 digits long, then Y consists of the 16 rightmost digits of X in numeric format.
2. Compute the two 8-byte numbers

   $$Z_L := \text{DES3(IMK)[Y]}$$

   and

   $$Z_R := \text{DES3(IMK)[Y} \oplus (\text{'FF'} \mid\mid \text{'FF'} \mid\mid \text{'FF'} \mid\mid \text{'FF'} \mid\mid \text{'FF'} \mid\mid \text{'FF'} \mid\mid \text{'FF'} \mid\mid \text{'FF'})]$$

   and define

   $$Z := (Z_L \mid\mid Z_R)$$

The 128-bit ICC Master Key MK is then equal to Z, with the exception of the least significant bit of each byte of Z which is set to a value that ensures that each of the 16 bytes of MK has an odd number of nonzero bits (this to conform with the odd parity requirements for DES keys).

A1.4.2 Option B

Option B is only applicable when the block cipher is Triple DES.

If the Application PAN is equal to or less than 16 decimal digits, use Option A.
If the Application PAN is greater than 16 decimal digits, do the following:

1. Concatenate from left to right the decimal digits of the Application PAN and the PAN Sequence Number (if the PAN Sequence Number is not present, it is replaced by a '00' byte). If the Application PAN has an odd number of decimal digits then concatenate a ‘0’ padding digit to the left thereby ensuring that the result is an even number of digits.
2. Hash the result of the concatenation using the SHA-1 hashing algorithm to obtain the 20-byte hash result X.
3. Select the first 16 decimal digits (0 to 9) starting from the left side of the 20-byte (40-nibble) hash result X and use as the value Y. If this does not provide for 16 decimal digits in Y, convert the non-decimal nibbles in X to decimal digits by means of the following decimalization table:

Input nibble of X	A	B	C	D	E	F
Decimalized nibble	0	1	2	3	4	5

Figure 14: Decimalization for Master Key Derivation

Add the converted digits starting from the left side of X to the end of Y until Y contains 16 digits.

Example 1: Hash result X contains 16 or more decimal digits

X = '12 30 AB CD 56 78 42 D4 B1 79 F2 CA 34 5D 67 89 A1 7B 64 BB'

Y = first 16 decimal digits of X = '12 30 56 78 42 41 79 23'

Example 2: Hash result X contains less than 16 decimal digits

X = '1B 3C AB CD D6 E8 FA D4 B1 CD F2 CA D4 FD C7 8F A1 7B 6E BB'

Y = decimal digits from X = '13 68 41 24 78 17 6' plus the required number of converted digits '1 20' (from 'B', 'C', and 'A'), giving:

Y = '13 68 41 24 78 17 61 20'

4. Continue with the processing specified for Option A starting at Step 2.

A1.4.3 Option C

Option C is only applicable when the *n*-byte block cipher is AES.

Concatenate from left to right the decimal digits of the Application PAN with the PAN Sequence Number (if the PAN Sequence Number is not present, then it is replaced by a '00' byte). Pad it to the left with hexadecimal zeros in order to obtain a 16-byte number Y in numeric format.

The k-bit ICC Master Key MK is then equal to

- In the case $k = 8n$:

  MK := AES(IMK)[Y]

- In the case $16n \geq k > 8n$:

  MK := Leftmost k–bits of {AES(IMK)[Y] | | AES(IMK)[Y*]}

  where Y* = Y ⊕ ('FF' | | 'FF' | | 'FF' | | 'FF' | | 'FF' | | 'FF' | | 'FF' | | 'FF').

A2 Asymmetric Mechanisms

A2.1 Digital Signature Scheme Giving Message Recovery

This section describes the special case of the 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.

A2.1.1 Algorithms

The digital signature scheme uses the following two types of algorithms.

- A reversible asymmetric algorithm consisting of a signing function Sign(S_K)[ ] depending on a Private Key S_K, and a recovery function Recover(P_K)[ ] depending on a Public Key P_K. Both functions map N-byte numbers onto N-byte numbers and have the property that

  $$\text{Recover}(P_K)[\text{Sign}(S_K)[X]] = X$$

  for any N-byte number X.

- A hashing algorithm Hash[ ] that maps a message of arbitrary length onto an 20-byte hash code.

A2.1.2 Signature Generation

The computation of a signature S on a message MSG consisting of an arbitrary number L of at least N – 21 bytes takes place in the following way.

1. Compute the 20-byte hash value H := Hash[MSG] of the message M.
2. Split MSG into two parts MSG = (MSG_1 | | MSG_2), where MSG_1 consists of the N – 22 leftmost (most significant bytes) of MSG and MSG_2 of the remaining (least significant) L – N + 22 bytes of MSG.

3. Define the byte B := '6A'.
4. Define the byte E := 'BC'.
5. Define the N-byte block X as the concatenation of the blocks B, MSG_1, H, and E, hence:

$$X := (B \,||\, MSG_1 \,||\, H \,||\, E)$$

6. The digital signature S is then defined as the N-byte number

$$S := Sign(S_K)[X]$$

A2.1.3 Signature Verification

The corresponding signature verification takes place in the following way:

1. Check whether the digital signature S consists of N bytes.
2. Retrieve the N-byte number X from the digital signature S:

$$X = Recover(P_K)[S]$$

3. Partition X as X= $(B \,||\, MSG_1 \,||\, H \,||\, E)$, where:
   - B is one byte long
   - H is 20 bytes long
   - E is one byte long
   - MSG_1 consists of the remaining N – 22 bytes
4. Check whether the byte B is equal to '6A'.
5. Check whether the byte E is equal to 'BC'.
6. Compute MSG = $(MSG_1 \,||\, MSG_2)$ and check whether H = Hash[MSG].

If and only if these checks are correct is the message accepted as genuine.

Annex B Approved Cryptographic Algorithms

B1 Symmetric Algorithms

These specifications use block cipher algorithms with key size *k* and block size *n*. For various reasons the key size is expressed in bits and the block size in bytes. Note that the mechanisms specified in Annex A only support block cipher algorithms satisfying $16n \geq k \geq 8n$.

B1.1 Data Encryption Standard (DES) 8-byte block cipher

The double-length key triple DES encipherment algorithm (see ISO/IEC 18033-3) is the approved cryptographic algorithm to be used in the encipherment and MAC mechanisms specified in Annex A1 where an 8-byte block cipher is used. The algorithm is based on the (single) DES algorithm standardised in ISO 16609.

Triple DES encipherment involves enciphering an 8-byte plaintext block in an 8-byte ciphertext block with a double-length (128-bit) secret key $K = (K_L \mid\mid K_R)$ as follows:

$$Y = DES3(K)[X] = DES(K_L)[DES^{-1}(K_R)[DES(K_L)[X]]]$$

Decipherment takes place as follows:

$$X = DES^{-1}(K_L)[DES(K_R)[DES^{-1}(K_L)[Y]]]$$

Single DES is only approved for usage with the version of the MAC mechanism specified in Annex A1 using Algorithm 3 of ISO/IEC 9797-1 (Triple DES applied to the last block).

B1.2 Advanced Encryption Standard (AES) 16-byte block cipher

The AES encipherment algorithm (see ISO/IEC 18033-3) is the approved cryptographic algorithm to be used in the encipherment and MAC mechanisms specified in Annex A1 where a 16-byte block cipher is used. 128-bit, 192-bit and 256-bit key length versions of AES are supported, but the key length should be fixed for a given implementation.

B2 Asymmetric Algorithms

B2.1 RSA Algorithm

This reversible algorithm (see reference [2] in Annex C) is the approved algorithm for encipherment and digital signature generation as described in Annex A2. The only values allowed for the public key exponent are 3 and $2^{16} + 1$.

The algorithm produces a cryptogram or digital signature whose length equals the size of the modulus used. The mandatory upper bounds for the size of the modulus are specified in Table 28.

Description	Max. Length
Certification Authority Public Key Modulus	248 bytes
Issuer Public Key Modulus	248 bytes
ICC Public Key Modulus	248 bytes
ICC PIN Encipherment Public Key Modulus	248 bytes

Table 28: Mandatory Upper Bound for Size in Bytes of Moduli

Furthermore, the length N_{CA} of the Certification Authority Public Key Modulus, the length N_I of the Issuer Public Key Modulus, the length N_{IC} of the ICC Public Key Modulus, and the length N_{PE} of the ICC PIN Encipherment Public Key Modulus shall satisfy $N_{IC} \leq N_I \leq N_{CA}$ and $N_{PE} \leq N_I \leq N_{CA}$.

In the choice of the lengths of the public key moduli, one should take into account the lifetime of the keys compared to the expected progress in factoring during that lifetime. The ranges (upper and lower bounds) for the key lengths mandated by each of the payment systems are specified in their corresponding proprietary specifications.

The value of the Issuer Public Key Exponent and the ICC Public Key Exponent is determined by the issuer. The Certification Authority, Issuer, and ICC Public Key Exponents shall be equal to 3 or $2^{16} + 1$.

The Public Key Algorithm Indicator for this digital signature algorithm shall be coded as hexadecimal '01'.

The keys and signing and recovery functions for the RSA algorithm with odd-numbered public key exponent are specified below.

B2.1.1 Keys

The private key S_K of the RSA digital signature scheme with an odd-numbered public key exponent e consists of two prime numbers p and q such that p – 1 and q – 1 are co-prime to e and a private exponent d such that:

$$ed \equiv 1 \bmod (p - 1)(q - 1)$$

The corresponding public key P_K consists of the public key modulus n = pq and the public key exponent e.

B2.1.2 Signing Function

The signing function for RSA with an odd-numbered public key exponent e is defined as:

$$S = \mathrm{Sign}(S_K)[X] := X^d \bmod n,\ 0 < X < n$$

where X is the data to be signed and S the corresponding digital signature.

B2.1.3 Recovery Function

The recovery function for RSA with an odd-numbered public key exponent e is equal to:

$$X = \mathrm{Recover}(P_K)[S] := S^e \bmod n$$

B2.1.4 Key Generation

Payment systems and issuers shall be responsible for the security of their respective RSA public/private key generation processes. Examples of secure key generation methods can be found in reference [1] in Annex C.

B3 Hashing Algorithms

B3.1 Secure Hash Algorithm (SHA-1)

This algorithm is standardised in ISO/IEC 10118-3. SHA-1 takes as input messages of arbitrary length and produces a 20-byte hash value.

The Hash Algorithm Indicator for this hashing algorithm shall be coded as hexadecimal '01'.

Annex C Informative References

1. A. Bosselaers and B. Preneel (eds.), *Integrity Primitives for Secure Information Systems*, Final Report of the RACE Integrity Primitives Evaluation (RIPE, RACE R1040), LNCS 1007, Springer-Verlag, 1995.
2. R. L. Rivest, A. Shamir, and L. Adleman, 'A method for obtaining digital signatures and public key cryptosystems,' *Communications of the ACM*, vol. 21, 1978, pp. 120-126.
3. *EMV Issuer and Application Security Guidelines*.
4. *NIST Special Publication 800-22*, 'A statistical test suite for random number and pseudorandom number generation for cryptographic applications'.
5. *EMV Acquirer and Terminal Security Guidelines*.

Annex D Implementation Considerations

D1 Issuer and ICC Public Key Length Considerations

This specification allows the Issuer Public Key length to be equal to or less than the CA Public Key length up to a maximum of 248 bytes (1984 bits) and allows the ICC Public Key and ICC PIN Encipherment Public Key lengths to be equal to or less than the Issuer Public Key length up to a maximum of 248 bytes (sections 5.1 and 6.1).

However, Book 3 section 7 states that records are limited to 254 bytes including tag and length and as a consequence, if an ICC public key pair is required, the Issuer and ICC key lengths need to be less than the maximum of 248 bytes.

Book 1 section 9.4.1 says that the maximum number of data bytes that may be sent with a command is 255 and the maximum number of data bytes for a response is 256. If dynamically signed data is included in a response from the ICC, then the latter restriction limits the maximum length of the ICC keys (see section D1.2.2).

D1.1 Issuer Public Key Restriction

For card applications supporting DDA, CDA, or Offline Enciphered PIN, the TLV encoded template containing the ICC Public Key Certificate needs to fit within the 254 byte record limit. To accommodate the tags and lengths of the certificate and the record template in the record containing this certificate, the maximum size of the ICC Public Key Certificate is restricted to 247 bytes (1976 bits), and consequently the Issuer Public Key, which is the same length as the certificate, is also restricted to 247 bytes.

D1.2 ICC Public Key Restriction

D1.2.1 CDA

The following restriction applies for card applications supporting CDA:

To ensure that the GENERATE APPLICATION CRYPTOGRAM response data length (format 2) is within the 256 byte constraint, the value portion of the Signed Dynamic Application Data needs to be limited in accordance with the other data elements contained within the template. This is achieved by limiting the size of the ICC public key, since owing to the properties of the cryptographic calculation, signature results are the same length as the key.

The lengths of the data in the GENERATE APPLICATION CRYPTOGRAM response are shown in Table 29:

		Length in Bytes			
		Tag	**Length**	**Value**	**Total Length**
Response Template		1 ('77')	2	—	3
	Cryptogram Information Data	2 ('9F27')	1	1	4
	Application Transaction Counter	2 ('9F36')	1	2	5
	Signed Dynamic Application Data	2 ('9F4B')	2	N_{IC}	N_{IC} plus 4
	Issuer Application Data (optional)	2 ('9F10')	1	0 to 32	0 to 35
	Other optional data				Var.

Table 29: Data Lengths in GENERATE AC Response

The tag and length of the response template, together with the tags, lengths, and values of the Cryptogram Information Data and Application Transaction Counter, and the tag and length of the Signed Dynamic Application Data are fixed in size and occupy 16 bytes. Thus without Issuer Application Data, the maximum size of the Signed Dynamic Application Data and consequently the ICC Public Key is 240 bytes (1920 bits).

If Issuer Application Data is included, then the maximum size of the Signed Dynamic Application Data needs to be reduced accordingly. Including Issuer Application Data of tag, length, and 32 bytes of value (the maximum) results in a maximum size of 205 bytes (1640 bits) for the Signed Dynamic Application Data and consequently the ICC Public Key.

Note: If other optional data is appended in the response, then the length of this data and its associated tag and length field further restricts the length of the ICC Public Key.

D1.2.2 DDA

The following restriction applies for card applications supporting INTERNAL AUTHENTICATE Format 2:

To ensure that the INTERNAL AUTHENTICATE response data length is within the 256 byte limit, the length of the Signed Dynamic Application Data plus the length of the TLV encoded optional data (if present) shall not exceed 249 bytes. The length of the ICC Public Key is the same as the Signed Dynamic Application Data. The additional 7 bytes in the response are used for the tags and lengths of the response template and the Signed Dynamic Application Data.

D2 Format 1 Secure Messaging Illustration

Below is an illustration of Format 1 Secure Messaging as defined in section 9 using a command where the command data of the unsecured command is not considered to be BER-TLV encoded and using an 8-byte block cipher. The command data is included in the computation of the MAC as a data object in accordance with section 9.2.3. This is either the plaintext data object with tag '81' or, if secure messaging for confidentiality is applied, the data object for confidentiality with tag '87'.

D2.1 Securing the Command APDU

Before application of secure messaging, a command APDU is called an 'unsecured' command. After secure messaging is applied, a command APDU is called a 'secured' command.

Throughout this section, L_c is the length of the unsecured command data field; that is, the length of the command data before encipherment, BER-TLV encoding or inclusion of a MAC. New L_c is the length of the secured command data field; that is, the command data length after encipherment (if applied), BER-TLV encoding and inclusion of the MAC.

The unsecured command APDU has either the following structure:

'X0'	INS	P1	P2	L_c	Unsecured command data field

or, if there is no data (e.g. Application Block command), the following structure:

'X0'	INS	P1	P2

The secured command APDU has the following structure:

'XC'	INS	P1	P2	New L_c	Secured command data field

The following sections describe the construction of the secured command data field and the value of New L_c. Firstly for the case where integrity[44] (only) is required and secondly for the case where both confidentiality and integrity are required.

[44] For the purposes of this Annex the term ‘integrity’ represents the combination of integrity and authentication.

D2.1.1 Secure Messaging for Integrity

The secured command data field:

If secure messaging for integrity (only) is applied, the secured command data field is TLV-coded in the following way:

Tag 1	Length 1	Value 1	Tag 2	Length 2	Value 2
'81'	L	Unsecured command data field	'8E'	'04'-'08'	MAC (4-8 bytes)

In this case L has the same value as L_c.

The value of New L_c:

L_c is always coded on one byte, whereas L is coded on one or two bytes, depending on the length of the unsecured command data.

- If L is coded on one byte, the value of New L_c will range from 8+L_c to 12+L_c, depending on the length of the MAC.
- If L is coded on two bytes, the value of New L_c will range from 9+L_c to 13+L_c, depending on the length of the MAC.

If the command to be secured has no command data (e.g. Application Block) then the only data in the secured command is the MAC. In this case the secured command data field is TLV-coded in the following way:

Tag 1	Length 1	Value 1
'8E'	'04' - '08'	MAC (4-8 bytes)

and the value of New L_c will range from 6 to 10 depending on the length of the MAC.

D2.1.2 Secure Messaging for Confidentiality and Integrity

The secured command data field:

If secure messaging for both confidentiality and integrity is applied, the secured command data field is TLV-coded in the following way:

Tag 1	Length 1	Value 1	Tag 2	Length 2	Value 2
'87'	L	'01' || enciphered data field	'8E'	'04'-'08'	MAC (4-8 bytes)

In this case:

- The first byte in the value field of the cryptogram data object for confidentiality with tag '87' is the padding indicator byte. The value '01' indicates that the plaintext data is padded according to ISO/IEC 7816-4 before encipherment.
- Since the plaintext data is padded to be a multiple of 8 bytes (see D2.2), the resulting enciphered data field will range from 1+L_c to 8+L_c. Consequently L will range from 2+L_c to 9+L_c.

The value of New L_c:

L_c is always coded on one byte, whereas L is coded on one or two bytes, depending on the length of the unsecured command data.

- If L is coded on one byte, the value of New L_c will range from 10+L_c to 21+L_c, depending on the length of padding appended to the unsecured command data and on the length of the MAC.
- If L is coded on two bytes, the value of New L_c will range from 11+L_c to 22+L_c, depending on the length of padding appended to the unsecured command data and on the length of the MAC.

If the command to be secured has no command data (e.g. Application Block) then there is no data to be enciphered and so secure messaging for integrity (only) is applied.

Notes

1. The plaintext data is transported in the value field of a plaintext data object with tag '81'.

   The enciphered data is transported in the value field of a cryptogram data object for confidentiality with tag '87'.
2. The fact that the tag of the data object (whether plaintext or cryptogram) is odd-numbered indicates that the data object is included in the MAC computation.
3. The padding indicator byte is the mandatory first byte in the value field of a cryptogram data object for confidentiality with tag '87' (see ISO/IEC 7816-4.)

D2.2 Encipherment

If secure messaging for confidentiality is applied to the command, the unsecured command data field is enciphered in the following way:

- Padding and blocking of the data field is performed according to Step 1 of Annex A1.1. A value of '01' of the padding indicator indicates that padding according to ISO/IEC 7816-4 always takes place even if the data field is a multiple of 8 bytes. Thus the unsecured command data field is always padded to a multiple of 8 bytes prior to encipherment with a minimum of one byte of padding always present; if the length of the unsecured command data field is a multiple of 8 bytes, it is padded with 8 bytes ('80 00 00 00 00 00 00 00') prior to encipherment.
- The padded data is enciphered according to Step 2 of Annex A1.1 using the Encipherment Session Key derived according to section 9.3.2.

D2.3 MAC Computation

MAC computation is performed in two steps:

- Padding of the input data (for use in this computation)
- Applying a MAC algorithm to the padded input data.

D2.3.1 Padding of the Input Data

Padding of the input data is performed according to ISO/IEC 7816-4:

- The command header of the secured command APDU

'XC'	INS	P1	P2

  is padded with '80 00 00 00'.

- If the unsecured command APDU contains a data field, a mandatory '80' byte is added to the right of
  - the plaintext data object (tag '81') or
  - the cryptogram data object for confidentiality (tag '87')

  that will be contained in the secured command data field. Then the smallest number of '00' bytes is added to the right such that the length of the resulting string is a multiple of 8 bytes.

The padded input data consists of the concatenation of the padded command header and the padded plaintext data object or the padded cryptogram data object for confidentiality (if present).

If MAC chaining is implemented then an 8-byte value is inserted to the left of the padded input data. This 8-byte value is:

- The Application Cryptogram generated by the card for the first or only script command,
- The MAC (the full 8 bytes prior to any optional truncation) of the preceding script command for all following script commands.

If MAC chaining is not implemented then the 8-byte Application Cryptogram generated by the card is inserted to the left of the padded input data.

D2.3.2 Cryptogram Computation

A MAC is computed over the padded input data according to Step 3 of Annex A1.2 using the MAC Session Key derived according to section 9.2.2.

D3 Application Transaction Counter Considerations

This specification describes a two byte (16 bit) counter (the ATC) that is incremented during each transaction from a nominal starting value of '0000' to a maximum of 'FFFF'. With one increment per card session it gives an expected card life of 65,535 transactions.

The counter results in uniqueness to the cryptograms and provides tracking values for the host verification services, allowing replayed transactions and cloned cards to be identified. It may also be used in session key derivation schemes, such as the scheme described in Annex A1.3.

To avoid attacks based on session truncation, the counter should be incremented at the start of each transaction (for example during processing of the GET PROCESSING OPTIONS command). To prevent attacks based on duplicate data the counter should not be allowed to roll-over and the application should be blocked once the counter reaches 'FFFF'. Issuers should be aware that few, if any, cards in normal use will approach the 65,535 transaction limit (60 per day every day for a 3 year card) and that cards with a high count may have been subject to attack. If a card with a shorter lifetime is desired, consideration may be given to a lower limit, or to starting the counter at an intermediate value.

D4 CDA Modes

Following publication of EMV Specification Update Bulletin 44 (SU44), EMV permits flexible terminal CDA behaviour that can potentially improve transaction performance. These include the selective use of CDA for online authorisations and public key retrieval relative to Terminal Action Analysis (TAA).

CDA for online authorisations

Terminals supporting CDA have the following options:

- Request or not request CDA on ARQCs
- Request or not request CDA on 2nd GENERATE AC (TC) after an approved online authorisation

As part of the EMV type approval, a terminal kernel configuration supporting CDA must now identify which of the above options the terminal supports.

Thus an EMV terminal configuration supporting CDA will operate in one of four modes:

Mode	Request CDA on ARQC	Request CDA on 2nd GEN AC (TC) after approved online authorisation
1	Yes	Yes
2	Yes	No
3	No	No
4	No	Yes

Table 30: CDA Modes

Public Key retrieval

Before publication of SU44, terminals experiencing CDA failure prior to TAA decline the transaction. Following publication of SU44, terminals that comply with SU44 that experience CDA failure prior to TAA shall proceed with TAA to decide whether to decline or send the transaction online.

One possible reason for CDA failing is a problem retrieving the public keys. According to SU44, terminals that find this problem before TAA will proceed with TAA to decide whether to decline or send the transaction online. Thus an online authorisation (without CDA) is possible, rather than the decline that was previously required.

SU44 also clarifies that keys can be retrieved before or after TAA which can lead to performance improvements for terminals operating predominantly online. This is because if the TAA results in an online authorisation and if the terminal requests an ARQC without CDA (i.e. it is operating in Mode 3 or 4), then the retrieval of the issuer and ICC public keys need not be completed, saving the RSA processing.[45]

Recommendations

The following recommendations apply to terminals supporting CDA.

All terminals should verify before TAA that they contain the Certification Authority Public Key identified by the card. Such verification does not involve time-consuming cryptographic processing. If the correct key is not present, then online terminals have an opportunity to send the transaction online without requesting CDA in the GENERATE AC.

For online capable terminals that are able to perform certificate verification quickly, it is recommended that the terminal retrieve the issuer and ICC public keys before TAA. This is to ensure that in the unlikely event that key retrieval fails then the terminal can request an ARQC without CDA rather than decline the transaction. Exceptions to this recommendation might be slower terminals which gain efficiencies by overlapping terminal certificate verification with card signature generation or Mode 3 terminals that normally send transactions online (i.e. request an ARQC rather than a TC at the first GENERATE AC) and for which a fast transaction is critical.

As Mode 4 does not provide significant benefit, terminal vendors are recommended not to implement Mode 4. If CDA is needed on all 2nd GENERATE AC commands requesting a TC, then Mode 1 can be used.

[45] If Offline Enciphered PIN is performed then this will force the retrieval of the issuer and ICC public keys to happen before PIN verification is performed.

Part IV

Common Core Definitions

Common Core Definitions

This Part describes an optional extension to this Book, to be used when implementing the Common Core Definitions (CCD).

These Common Core Definitions specify a minimum common set of card application implementation options, card application behaviours, and data element definitions sufficient to accomplish an EMV transaction. Terminals certified to be compliant with the existing EMV specifications will, without change, accept cards implemented according to the Common Core Definitions, since the Common Core Definitions are supported within the existing EMV requirements.

To be compliant with the Common Core Definitions, an implementation shall implement all the additional requirements in the Common Core Definitions Parts of all affected Books.

Changed Sections

Each section heading below refers to the section in this Book to which the additional requirements apply. The text defines requirements for a common core implementation, in addition to the requirements already specified in the referenced section of EMV.

Part II - Security and Key Management Techniques

6 Offline Dynamic Data Authentication

6.5 Dynamic Data Authentication (DDA)

6.5.1 Dynamic Signature Generation

An ICC that supports DDA shall contain a DDOL. The DDOL shall contain only the Unpredictable Number generated by the terminal (tag '9F37', 4 bytes binary).

6.6 Combined DDA/Application Cryptogram Generation (CDA)

6.6.1 Dynamic Signature Generation

For a CCD-compliant application that supports CDA, the following requirements shall apply.

The ICC response to the GENERATE AC command for a TC or ARQC shall contain only the data objects specified in Table CCD 1 (which, for CCD, supplants Table 20).

Tag	Length	Value	Presence
'9F27'	1	Cryptogram Information Data	M
'9F36'	2	Application Transaction Counter	M
'9F4B'	N_{IC}	Signed Dynamic Application Data	M
'9F10'	32	Issuer Application Data	M

Table CCD 1: Data Objects in Response to GENERATE AC for TC or ARQC

3. If the ICC responds with an AAC, the ICC response shall be coded according to format 2 as specified in section 6.5.5.4 of Book 3 and shall contain only the data elements specified in Table CCD 2 (which, for CCD, supplants Table 21).

Tag	Length	Value	Presence
'9F27'	1	Cryptogram Information Data	M
'9F36'	2	Application Transaction Counter	M
'9F26'	8	Application Authentication Cryptogram	M
'9F10'	32	Issuer Application Data	M

Table CCD 2: Data Objects in Response to GENERATE AC for AAC

8 Application Cryptogram and Issuer Authentication

8.1 Application Cryptogram Generation

8.1.1 Data Selection

Table CCD 3 lists the set of data elements to be included in the Application Cryptogram for a cryptogram defined by the Common Core Definitions with a Cryptogram Version of '5' (which uses Triple DES) or '6' (which uses AES). The data elements shall be included in the order shown in Table CCD 3 [which, for CCD, supplants Table 26].

Value	Source
Amount, Authorised (Numeric)	Terminal
Amount Other (Numeric)	Terminal
Terminal Country Code	Terminal
Terminal Verification Results	Terminal
Transaction Currency Code	Terminal
Transaction Date	Terminal
Transaction Type	Terminal
Unpredictable Number	Terminal
Application Interchange Profile	ICC
Application Transaction Counter	ICC
Issuer Application Data	ICC

Table CCD 3: Data Elements for Application Cryptogram Generation

8.1.2 Application Cryptogram Algorithm

The Application Cryptogram shall be 8-bytes. EMV-defined Application Cryptograms shall be generated using one of two methods:

- using the MAC algorithm specified in Annex A1.2.1 and ISO/IEC 9797-1 Algorithm 3 with DES, and *s*=8. This method shall be used for a Cryptogram Version of '5'.
- using the MAC algorithm specified in Annex A1.2.2 with AES and *s*=8. This method shall be used for a Cryptogram Version of '6'.

For an application defined by the Common Core Definitions with a Cryptogram Version of '5' or '6', the AC Session Key shall be derived using the method specified in Annex A1.3.1.

8.2 Issuer Authentication

The CCD-compliant application shall support Issuer Authentication according to ARPC Method 2 specified in section 8.2.2.

8.2.2 ARPC Method 2

For a cryptogram defined by the Common Core Definitions with a Cryptogram Version of '5' or '6', the Card Status Update (CSU) data element shall be coded according to Annex C8 in the CCD part of Book 3.

The default value for Proprietary Authentication Data is zero.

If the ‘Proprietary Authentication Data Included’ bit in the CSU has the value 0b, then the length of Proprietary Authentication Data included in generation and validation of the ARPC shall be 0 bytes.

Note: If the ‘Proprietary Authentication Data Included’ bit in the CSU has the value 0b, the Proprietary Authentication Data is not protected by Issuer Authentication. The card should not take action based on the settings of any Proprietary Authentication Data that is not protected by Issuer Authentication.

If the ‘Proprietary Authentication Data Included’ bit in the CSU has the value 1b, then the Proprietary Authentication Data included in the Issuer Authentication Data shall be used in generation and validation of the ARPC.

8.3 Key Management

For a cryptogram defined by the Common Core Definitions with a Cryptogram Version of '5', the ICC Master Key shall be derived using the Option B method described in Annex A1.4.2.

For a cryptogram defined by the Common Core Definitions with a Cryptogram Version of '6', the ICC Master Key shall be derived using the Option C method described in Annex A1.4.3.

9 Secure Messaging

9.1 Secure Messaging Format

All commands using Secure Messaging shall use Secure Messaging Format 1 as described in this Book.

9.2 Secure Messaging for Integrity and Authentication

9.2.1 Command Data Field

All commands using Secure Messaging for integrity and authentication:

- shall use Secure Messaging Format 1 as described in section 9.2.1.1
- shall chain the MACs from one command to the next according to the method recommended in section 9.2.3.1.

9.2.1.1 Format 1

All command data shall be included in the computation of the MAC.

Data enciphered for confidentiality shall be encapsulated with tag '87'. Data not enciphered for confidentiality shall be encapsulated with tag '81'.

The CCD-compliant application shall accept 4-byte MACs, and the issuer can only rely on support of 4-byte MACs.

9.2.2 MAC Session Key Derivation

For an application with a cryptogram defined by the Common Core Definitions with a Cryptogram Version of '5' or '6', the MAC Session Key shall be derived using the method specified in Annex A1.3.1.

9.2.3 MAC Computation

Secure Messaging is according to Secure Messaging Format 1.

The CCD-compliant application shall accept 4-byte MACs, and the issuer can only rely on support of 4-byte MACs.

The MAC for Cryptogram Version '5' shall be generated using the MAC algorithm specified in Annex A1.2.1, Step 3, second bullet using DES as the block cipher.

The MAC for Cryptogram Version '6' shall be generated using the MAC algorithm specified in Annex A1.2.2 using AES as the block cipher.

9.3 Secure Messaging for Confidentiality

9.3.1 Command Data Field

All commands using secure messaging for confidentiality shall use Secure Messaging Format 1 as described in section 9.3.1.1.

9.3.1.1 Format 1

Data enciphered for confidentiality shall be encapsulated with Tag '87'.

Data that is enciphered in the Issuer Script Command data field shall always be padded before encipherment. The Padding Indicator byte shown in Figure 8 shall be included and shall be set to the value '01' to indicate padding is present.

9.3.2 Encipherment Session Key Derivation

For an application with a cryptogram defined by the Common Core Definitions with a Cryptogram Version of '5' or '6', the Encipherment Session Key shall be derived using the method specified in Annex A1.3.1.

9.3.3 Encipherment/Decipherment

Encipherment/decipherment of the command data field shall use the Cipher Block Chaining (CBC) Mode described in Annex A1.1. For Cryptogram Version '5' the Triple DES algorithm specified in Annex B1.1 is used. For Cryptogram Version '6' the AES algorithm specified in Annex B1.1 is used. For both versions the Padding Indicator byte is set to the value '01' to indicate that padding is present.

9.4 Key Management

For an application with a cryptogram defined by the Common Core Definitions with a Cryptogram Version of '5', the ICC MAC and Encipherment Master Keys shall be derived using the Option B method described in Annex A1.4.2.

For an application with a cryptogram defined by the Common Core Definitions with a Cryptogram Version of '6', the ICC MAC and Encipherment Master Keys shall be derived using the Option C method described in Annex A1.4.3.

Index