N684r18

EMV
Issuer and Application Security Guidelines

EMVCo, LLC

Version 2.4
April, 2014

TABLE OF CONTENTS

1 Scope

These Security Guidelines are designed to assist issuers of EMV payment cards with key management and issuance processes. The issuer is liable for both the accuracy and the protection of the data used in the personalisation of its cards. Such data includes, in addition to the cardholder and account information, cryptographic keys and other cardholder secrets, that, if revealed to unauthorised parties, could result in the creation of counterfeit cards and fraudulent transactions. To protect against the unauthorised disclosure of this information, issuers must create and manage these types of data in a secure environment. Such an environment is one where the appropriate physical and logical security controls have been implemented and where sufficient audit trails have been established to assure procedural consistency relative to the issuer's security objectives.

The guidance presented in the following pages is applicable to all phases of card personalisation, authorisation, and transactional clearing. Where issuers use third party agents to perform these functions, the same principles are also applicable. Application of these principles may also be useful for other payment applications that require and use similar sensitive data.

The materials contained in this document are intended primarily for issuers and their agents. This document is not intended to supersede the requirements and specifications of any Payment System. The document is to be used as a “guideline”, assisting the issuer, the issuer's agent and other third parties regarding the secure implementation of an EMV specification.

2 References

Throughout this document, the following references have been used. These references include the most current version at the time of preparation. For future use the most current versions of these documents should be used.

EMV Version 4.3	Integrated Circuit Card Specifications for Payment Systems. Book 2 Security and Key Management
ISO 9564-1	Financial Services – PIN management and security – Part 1: Basic principles and requirements for PINs in card-based systems.
ISO 11568	Banking – Key management (retail)
ISO 13491	Banking – Secure cryptographic devices (retail)
ISO 16609	Banking – Requirements for message authentication using symmetric techniques
ISO/IEC 18031	Information technology - Security techniques – Random bit generation
ISO/IEC 18032	Information technology - Security techniques - Prime number generation

3 Definitions

Authentication — A cryptographic process that validates the identity and integrity of data.

Certificate (public key) — The public key and the identity of an entity together with some other information, made unforgeable by the signing of the certificate with the private key of the certification authority issuing the certificate.

Certification Authority — The entity that is trusted by one or more other entities to create and assign certificates.

Cryptographic Algorithm — A set of rules, setting forth procedures necessary to authenticate or protect data, e.g. to perform encipherment and decipherment of data. The algorithm is specified in a manner that it is not possible to determine any of the secret control parameters; i.e., the secret or private key, except by exhaustive search.

Digital Signature — The cryptographic transformation of data which provides:

- origin authentication, and
- data integrity.

Dual Control — The process of utilising two or more separate entities to protect sensitive information or functions, such that no single entity is able to access or utilise the information or functions.

Hash Function — A function, which maps values from a large domain into a smaller one. The function satisfies the following properties:

1. It is computationally infeasible to find for a given output, an input that maps to this output.
2. It is computationally infeasible to find for a given input, a second input that maps to the same output.

IC Card (ICC) — A card with an embedded integrated circuit (chip) that communicates with a point of interaction (terminal).

Key Component — One of at least two parameters having the characteristics of randomness and format of a cryptographic key that is combined with one or more like parameters, forming the cryptographic key.

Key Pair — When used in public key cryptography, a public key and its corresponding private key.

Key Space — A set of all possible keys used by a cryptographic algorithm.

Keying Material	The data (e.g. keys, certificates, initialisation vectors) necessary to establish and maintain cryptographic keys.
Payment System	A Payment System includes a number of participants where the issuer and the acquirer distribute responsibilities amongst the different parties according to Payment System rules and according to the allocation of risks.
Physically Secure Device	A module that has a negligible probability of entry without detection or erasure of its contents.
Private Key	In an asymmetric algorithm (public key) cryptosystem, the key of an entity's key pair that is known only to that entity. This is not the same as the secret key used in a symmetric algorithm.
Pseudo-random	A process that produces numbers that are statistically random and essentially unpredictable although generated by an algorithmic process.
Public Key	In an asymmetric key system, the key of an entity that is publicly known.
Secret Key	A key that is used in a symmetric cryptographic algorithm and cannot be disclosed publicly without compromising the security of the system. This is not the same as the *private* key in a public/private key pair.
Secure Cryptographic Device	A device that provides physically and logically protected cryptographic services and storage (e.g. PIN Entry device or Hardware Security Module), and which may be integrated into a larger system such as a point of sale device.
Split Knowledge	A condition whereby two or more parties; i.e., key custodians, separately and confidentially have custodial control of a constituent part of a cryptographic key that individually conveys no knowledge of the resultant cryptographic key.

4 Abbreviations and Notations

AC	Application Cryptogram
AES	Advanced Encryption Standard
AIP	Application Interchange Profile
ARPC	Authorisation Response Cryptogram
ARQC	Authorisation Request Cryptogram
ATC	Application Transaction Counter
ATM	Automated Teller Machine
CA	Certificate Authority
CDA	Combined DDA/Application Cryptogram Generation
CDOL	Card Risk Management Data Object List
CPA	Common Payment Application
CRL	Certificate Revocation List
CRT	Chinese Remainder Theorem
DDA	Dynamic Data Authentication
DES	Data Encryption Standard
HSM	Hardware Security Module
IAD	Issuer Application Data
IC	Integrated Circuit
ICC	Integrated Circuit Card
ICCCert	ICC Public Key Certificate
IDN	ICC Dynamic Number
IIN	Issuer Identification Number, also known as Bank Identification Number (BIN)
IMK_{AC}	Issuer Master Key for Application Cryptogram computation
IMK_{SMC}	Issuer Master Key for Secure Messaging for Confidentiality
IMK_{SMI}	Issuer Master Key for Secure Messaging for Integrity
MAC	Message Authentication Code
MK	Card Master Key
MK_{AC}	Card Master Key for Application Cryptogram computation
MK_{SMC}	Card Master Key for Secure Messaging for Confidentiality
MK_{SMI}	Card Master Key for Secure Messaging for Integrity
PAN	Application Primary Account Number
PECert	ICC PIN Encipherment Public Key Certificate
POS	Point of Sale

RSA	Rivest, Shamir, Adleman algorithm
SCD	Secure Cryptographic Device
SDA	Static Data Authentication
SDAD	Signed Dynamic Application Data
SK_{AC}	Session Key for Application Cryptogram computation
SK_{SMC}	Session Master Key for Secure Messaging for Confidentiality
SK_{SMI}	Session Master Key for Secure Messaging for Integrity
TC	Transaction Certificate
TDES	Triple DES (referred to as DES3 in EMV Book 2)
TVR	Terminal Verification Results

5 EMV and Cryptography – Overview

5.1 Introduction

The purpose of this overview is to provide a framework for the issuer security guidelines. The EMV payment system model is described together with an outline of the roles of the entities within the model.

5.2 Payment System Model

The Payment System (as outlined in Figure 1) consists of the following types of entity:

- Cardholders,
- Merchants,
- Issuers,
- Acquirers, and
- Payment System brands (e.g. Amex, Discover, JCB, MasterCard, UnionPay and Visa).

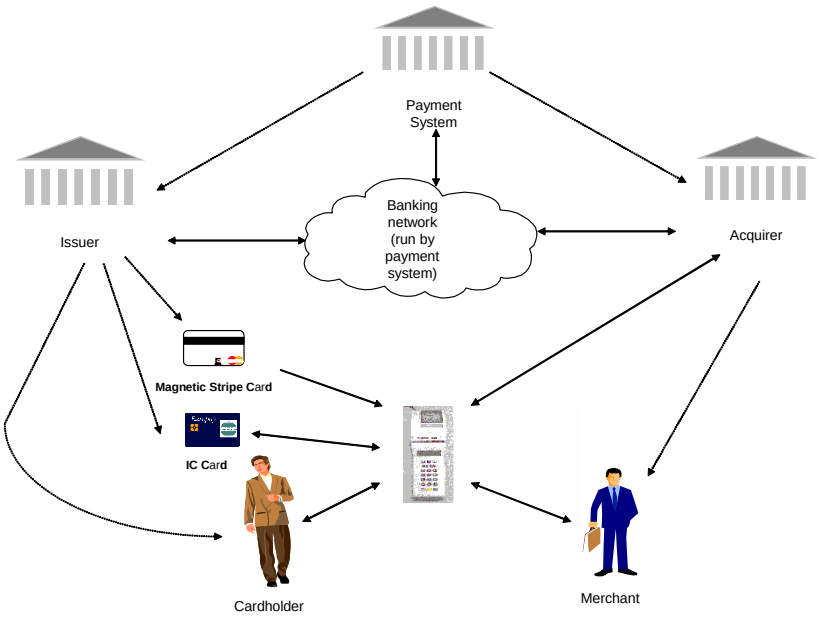


Figure 1 - System Model

The main role of each of these entities is as follows.

5.2.1 The Cardholder

The role of the cardholder includes the following:

- To obtain a chip card containing the payment product application by contracting with an issuer.
- To choose, remember and possibly update his/her PIN.

- To present his/her chip card to devices accepting the payment product for payment (ATM, Merchant POS, vending machines, payphones, etc.).

5.2.2 The Merchant

The role of the merchant includes the following:

- To obtain payment terminals accepting chip cards by contracting with an acquirer.
- To accept chip cards containing the payment products for payment.
- To obtain reimbursement for the purchases by collecting and transmitting payment transaction details to the acquirer.

5.2.3 The Issuer

The role of the issuer includes the following:

- To contract with the cardholder, and to personalise and issue a chip card containing the application to the cardholder. This includes the generation and installation of the necessary cryptographic keys in the card to support the application.
- To process online transactions. This includes verification of the data and cryptogram from the card together with data from the terminal, plus generation of a cryptogram allowing the card to authenticate the issuer. It also includes verification of online cardholder PINs as part of standard authorisation processing.
- To generate update scripts to the card application when appropriate.
- To process clearing messages including verification of the data and associated Transaction Certificate, when appropriate. In some circumstances this could be deferred and only checked in the case of dispute.
- To reimburse the acquirer for payment transactions.
- To securely transmit to any other parties the necessary cryptographic keys needed for the correct operation of the system.

5.2.4 The Acquirer

The role of the acquirer includes the following:

- To contract with merchants and to deploy payment terminals. This includes the installation and management of Payment System public keys, adequately protected for integrity.
- To process payment transactions and to pay the merchant for them.
- To transmit the completed transaction records to the issuer in order to obtain the settlement.
- To manage the risk conditions relating to online/offline acceptance.

5.2.5 The Payment System

The role of the Payment System includes the following:

- To specify the system rules for the products and services and to verify compliance with them.
- To generate and distribute Payment System public keys.
- To certify Issuer Public Keys used within the system.
- To operate on-line communication networks between acquirers and issuers.
- To perform clearing and settlement for transactions on this network.

5.3 Cryptographic Basics

Historically, cryptography has been used to provide data confidentiality and today includes additional cryptographic functions such as data integrity, authentication, and non-repudiation. International standards have been developed to facilitate interoperability of products and services between different vendors and various cryptographic implementations. The materials contained in these guidelines represent the best practices drawn from these different standards.

Modern cryptography depends on two basic components: (1) the algorithm, and (2) the cryptographic key, with overall security dependant on public access to the algorithm and secure management of the secret and private keys over the key lifecycle. The algorithms define how ciphertext is obtained from plaintext and vice versa and how data is signed and verified. Algorithms are typically published and have been extensively studied by cryptographers – it is the use of unique keys for every user that ensures that unauthorised parties are unable to decrypt sensitive data or forge another parties' digital signature.

There are two basic types of cryptographic algorithms: (1) symmetric or secret key algorithms, and (2) asymmetric or public/private key algorithms and both are used within the context of EMV.

5.3.1 Symmetric Algorithms

Symmetric or ‘secret key’ algorithms require that the secret key used for the encryption process also be used in the decryption process. Therefore, the security of the encryption process depends entirely on protection of this secret key.

The EMV application supports the use of the Data Encryption Standard (DES), a symmetric algorithm which is widely used in the financial services industry today. DES belongs to the family of encryption algorithms called “block” ciphers because they process data in blocks. The DES algorithm takes an input block of 64-bits and maps it to a 64-bit output block, using a 56-bit key in an iterative process of sixteen rounds. As an option, the more recent AES algorithm is also supported. The MAC length is retained at 8 bytes for backwards compatibility.

All references in the text of this document regarding the use of the DES algorithm are to be read as references to the use of Triple DES (TDES), typically 2 key Triple DES or 3 key Triple DES if issuer chooses, in which a single DES encryption is replaced

with three DES operations. Similarly, references to DES keys are to be read as pairs (or triples) of DES keys, as used in conjunction with Triple DES. Therefore, all EMV DES keys are 16 bytes or 128 bits in length (24 bytes or 192 bits in length).

Potential risks to symmetric keys include:

- The physical compromise of the secret key.
- Side channel attacks on keys held in chip cards.
- Exhaustive key search attacks – currently computationally infeasible for TDES.

5.3.2 Asymmetric Algorithms

Asymmetric or 'public key' algorithms require the communicating endpoints to use two different, but linked, keys: a "public" key and a "private" key. The RSA asymmetric algorithm is used by EMV to create digital signatures and for offline PIN encipherment. In a digital signature scheme the private key (sometimes referred to as the signature key) is used to generate the signature and the public key (sometimes referred to as the verification key) is used to verify the signature. For offline PIN encipherment, the public key is used for encipherment and the private key is used for decipherment.

Public key algorithms are generally based on a "hard" mathematical problem and have a design goal that there should be no better way to attack the scheme other than solving the hard problem. RSA is based on the hard problem of factorisation; that is, for a number consisting of two prime numbers multiplied together, find the primes given only the product, known as the modulus. The longer the modulus (key length), the "harder" the problem of breaking the key.

5.3.2.1 Asymmetric (RSA) Keys

The security of the private (signature) keys used with the RSA algorithm depends on a number of factors including:

- The length of the RSA key modulus; e.g. 1024, 1152, 1408, and 1984 bit keys.
- The physical security of the private (signature) key from unauthorised access and exposure/compromise whilst in storage, in transit or in use.
- The quality of the prime numbers making up the public/private key modulus.

Potential risks to the private (signature) key include:

- Physical compromise.
- Factorisation of the RSA modulus.
- Side channel attacks on keys held in chip cards
- Collapse of the underlying algorithm – most unlikely after several decades of cryptanalysis.

The EMVCo Security Working Group conducts an annual review of Payment System key lifetimes based on independent analyses by the participating Payment Systems. Using the recommendations from the review, the Payment Systems may update their Payment System key lifetimes.

The lifetimes for the longer keys are currently considered to be 'anticipated lifetimes' in order to reflect the situation that when looking more than ten years into the future, the variation in prediction becomes too large for a reliable date to be given. Over time these dates are also expected to move out, until a lifetime of ten years or less is predicted at which time the date will be considered as an expiry date.

Note that all key lifetimes are subject to change.

5.3.2.2 Certificates and Certification Authorities

In traditional cryptographic systems, key management has primarily been concerned with the establishment and maintenance of shared secret keys. With the growing use of public key cryptography, the nature of the key management problem has changed. Public keys can be widely disseminated while the corresponding private key needs to be kept secret by its owner.

Public keys need to be distributed in a reliable way to those parties that use them. Although these public keys do not need to be kept confidential, the recipient of a public key needs to know that the public key came unaltered from the purported owner. EMV uses the common practice of certificates to validate the source of the public key.

To understand what a certificate is, we need to consider one specific type of public key cryptosystem, namely a digital signature scheme. With a digital signature scheme, an entity's private key can be used to sign a message, i.e. to generate a string of bits known as a digital signature, which is a function of the message being signed and the entity's private key. The recipient of a message with a digital signature can verify the signature by using the signer's public key.

A certificate is a form of digital signature and can be used to check the origin and integrity of a public key. A certificate consists of a public key concatenated with other related data (including the user name or identifier and expiration date) and signed with the private key of a trusted entity known as a Certification Authority (CA). Any entity with a trusted copy of the CA's Public Key can then verify all certificates generated by that CA and thereby obtain trusted copies of other users' public keys.

In the EMV environment, the Payment System acts as a Certification Authority and creates Issuer Public Key certificates by signing each issuer's public keys. Issuers act as Certification Authorities and create ICC Public Key certificates by signing each ICC Public Key. The Payment Systems CA's Public Keys are distributed to the terminals through the acquirers for verifying issuer certificates, thereby yielding trusted copies of issuers' public keys, used in turn to verify ICC Public Keys.

5.4 EMV Authentication Methods

EMV supports offline and online methods for authenticating that a card is genuine and that the data on the card has not been altered since it was personalised by the issuer.

5.4.1 Offline Data Authentication

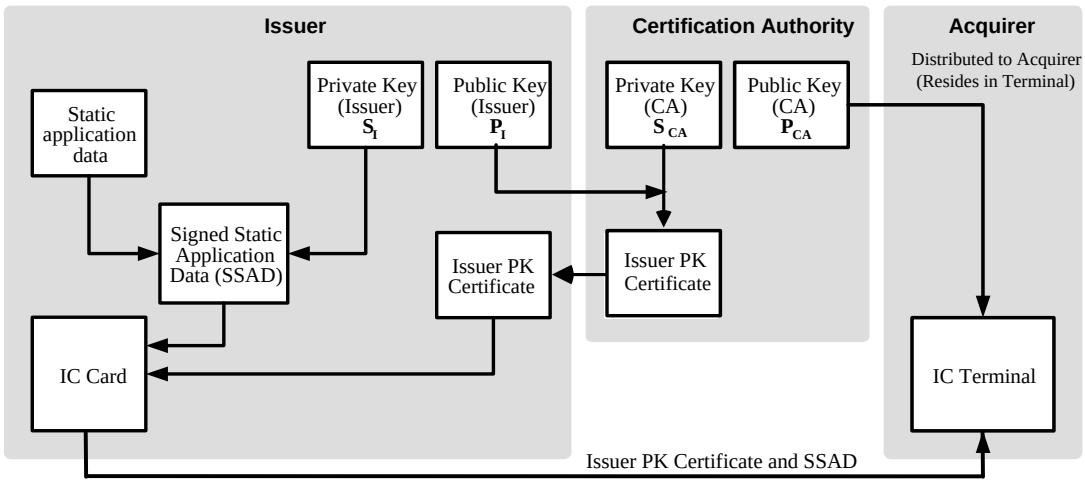
Static Data Authentication (SDA), Dynamic Data Authentication (DDA) and Combined DDA/Application Cryptogram Generation (CDA) are the offline authentication methods specified in EMV. This means that the terminal rather than the issuer uses these methods to authenticate the card and card data.

- With SDA the terminal verifies a static signature (i.e. the same for every transaction) of card data, in order to assure that this data has not been altered.
- With DDA the terminal verifies a dynamic signature (i.e. different for each transaction) generated by the card using its private key, in order to ensure that the card is not counterfeit and that the card data has not been altered.
- With CDA the card generates a dynamic signature of transaction data including the online cryptogram, in order to provide the protection of DDA while also ensuring that an intermediate (wedge) device has not altered important data going between the card and terminal.

5.4.1.1 Static Data Authentication (SDA)

SDA is a mechanism where the terminal uses a digital signature based on public key techniques to confirm the legitimacy of critical ICC-resident static data, the Signed Static Application Data (SSAD).

The relationship between the data and the cryptographic keys is shown in Figure 2. For further information, please refer to EMV Book 2.



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

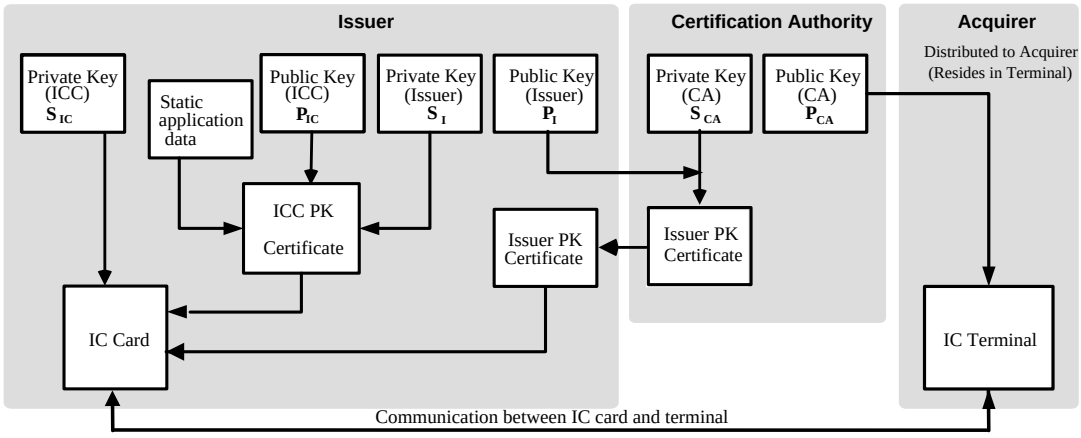
Figure 2 - SDA

Given the static nature of SDA, it is possible for the relevant data to be copied (cloned) from a legitimate card and applied to a counterfeit product[1]. Such a cloned card will work in offline environments, but will fail if the transaction is sent online.

5.4.1.2 Dynamic Data Authentication (DDA / CDA)

DDA and CDA are mechanisms providing dynamic data authentication where the terminal uses a digital signature based on public key techniques to authenticate the ICC and to confirm the legitimacy of critical ICC-resident data. This prevents the cloning of cards able to pass offline dynamic data authentication.

The relationship between the data and the cryptographic keys is shown in Figure 3. For further information, please refer to EMV Book 2.



Card provides to Terminal:

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

Terminal:

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

Figure 3 - DDA / CDA

CDA performs similar functions to DDA except that the Application Cryptogram and other transaction data (e.g. approval status) are included in the Signed Dynamic Application Data (the dynamic signature), rather than being returned in a separate operation. This allows the terminal to verify that the above information was generated by the card that produced the signature and to detect alterations made by an intermediate or 'wedge' device inserted between the card and terminal.

5.4.2 Online Authentication

EMV's online authentication methods are used to validate the card to the issuer and the issuer to the card as well as to prove the authenticity of received data.

- With Online Card Authentication (CAM) the issuer online system validates a cryptogram (an Application Cryptogram called an ARQC) generated by the card from important transaction data using its unique secret key, to show that the card is not counterfeit and that the data has not been altered.

[1] This has to be a "blank" chip obtained by the fraudster as unlike magnetic-stripe, where legitimate products can be re-encoded, approved EMV chips cannot be re-written.

- With Online Issuer Authentication, the card validates an issuer-generated cryptogram and then performs internal card management functions, such as the reset of offline counters.
- With secure messaging the issuer sends a script update to the card protected by a MAC. The card only applies the updates if the MAC is valid. Secure messaging is also used to encipher confidential data, such as a replacement PIN value during transport between the issuer and the card.
- For approved transactions the terminal sends a cryptogram (an Application Cryptogram called a TC) generated by the card with the clearing information for verification by the issuer as evidence of the validity of the completed transaction.

5.4.3 Comparison of Authentication Methods

The following tables illustrate the protection provided by these authentication methods and the varying level of impact.

	SDA	**DDA**	**CDA**	**Online Card Authentication**
Detects fabricated account data	✓	✓	✓	✓
Detects altered static data	✓	✓	✓	
Detects cloned data		✓	✓	✓
Can prevent wedge attacks			✓	✓
Available for offline transactions	✓	✓	✓	

Table 1 - EMV Data/Card Authentication Methods

It should be noted that the three offline methods provide increasing levels of security and the following table shows the impacts of each of these methods on the card, terminal, host processing, and transaction times.

	SDA	**DDA**	**CDA**	**Online Card Authentication**
Card		Must support RSA	Must support RSA	Must support TDES (typically)
Terminal	Must support RSA	Must support RSA	Must support RSA	Must be online capable
Host Systems		Must personalise ICC key	Must personalise ICC key	Must support ICC authorisations
Transaction Time	2 terminal RSA operations	3 terminal RSA operations 1 ICC RSA operation	3 terminal RSA operations 1 ICC RSA operation	Requires an online authorisation

Table 2 – Impacts of EMV Data/Card Authentication Methods

As the cost of RSA capable cards has fallen over time, then migration is occurring away from SDA to either DDA or CDA. For many implementations DDA is now the recommended minimum CAM that should be supported.

5.4.4 Cardholder Verification Methods

EMV supports the following methods for verifying a legitimate cardholder:

- Signature
- Online PIN, where a cardholder-entered value is enciphered by the terminal/PIN pad and sent with an online authorisation request to the issuer for validation.
- Offline PIN, where a cardholder-entered value is sent to the card and the ICC compares this value to a reference PIN stored securely on the card. The terminal is informed of the success or failure of the verification.

EMV provides for two types of offline PIN verification:

- Offline Plaintext PIN, where the cardholder-entered value sent to the ICC is unencrypted.
- Offline Enciphered PIN, where the cardholder-entered value is RSA-enciphered by the terminal and deciphered by the ICC

To prevent PIN probing attacks, dual interface (and contactless-only) cards should not respond to the VERFIY command over the contactless interface.

Offline PIN encryption may be supported using either the DDA/CDA keys or dedicated keys. For security reasons the use of dedicated keys is a best practice, however dedicated keys may impact key management and transaction performance.

The following tables illustrate the protections provided by these verification methods and the impacts to consider when choosing the methods to support.

	Signature	**Offline Plaintext PIN**	**Offline Enciphered PIN**	**Online PIN**
Available with offline transactions	✓	✓	✓	
Available at unattended devices		✓	✓	✓
PIN is always enciphered during transit			✓	✓

Table 3 - EMV Cardholder Verification Methods

Issuers should be aware of the following statement made by EMVCo in early 2011:

> *"The EMV Specifications for payment cards and terminals provide interoperability and security features, which act as building blocks for the payment systems and financial institutions to design their products and processes according to their wider risk management and acceptance requirements. In response to the report in March 2011 'Chip and PIN is Definitely Broken', it is EMVCo's view that when the full payment process is taken into account, suitable countermeasures are available.*
>
> *For example, it is well known that PINs can be stolen by the use of a variety of techniques (e.g. PIN pad overlays, hidden cameras, shoulder surfing, bogus terminals, social engineering). Using a rogue shim in a terminal supporting offline plaintext PIN (possibly subverting the card's PIN encipherment preferences and causing an offline card authentication failure or even a decline) is another technique. The mitigation against this threat is that no transaction can be performed without also stealing the card where card cryptography operations are required for a successful transaction. This allows normal lost and stolen payment system protections to apply. Conversely the*

mitigation against a genuine card being abused if lost or stolen is that the thief will not have access to the PIN, hence the PIN has a role to play despite the 'eavesdropping threat' and remains an important tool for protecting against lost and stolen fraud. "

All of the PIN methods involve a precise interaction with the cardholder, the result of which is either right or wrong, whereas signature requires a human comparison that is subjective.

The following table shows the impacts of each of these methods on the card, terminal, host processing, and transaction times.

	Signature	**Offline Plaintext PIN**	**Offline Enciphered PIN**	**Online PIN**
Card			Must support RSA	
Terminal	Must retain signature	Must support PIN pad	Must support PIN pad and RSA	Must support PIN pad and be online capable
Acquirer Host Systems				Must support secure (enciphered) transport of Online PIN to authorisation system.
Issuer Host Systems		Must personalise card with PIN	Must personalise card with PIN & ICC key	Must support online PIN verification as part of authorisation
Transaction Time	Signature collection and checking time.	PIN entry time	PIN entry and RSA time	PIN entry and online authorisation time

Table 4 - Impacts of EMV Cardholder Verification Methods

5.5 The Authorisation System

Authorisation is a process whereby an issuer or a representative of the issuer approves or declines a transaction in response to an online authorisation request from a merchant via an acquirer.

The online authorisation request includes a card generated authorisation cryptogram (ARQC) which the issuer validates to ensure that the card is authentic and the transaction data is unaltered. The online request also includes card and terminal indicators of the results of offline processing.

In response to the ARQC, the issuer optionally creates an authorisation response cryptogram (ARPC). The card validates the ARPC to assure that the authorisation response came unaltered from the issuer.

In addition to the ARPC described above, issuers can perform post-issuance updates of cards using issuer script commands. For example the issuer can change the Offline PIN or update a card's risk parameters. The issuer protects these script commands from undetected alteration by generating a cryptogram (MAC) from the command data. The card validates the MAC before applying the changes. Confidential data is enciphered.

6 Security for EMV Card Issuance

This section addresses the security related functions that need to be performed by an EMV card issuer.

- The generation, management and secure storage of the asymmetric issuer public/private key pairs.
- The transfer of the Issuer Public Keys to the Payment System CA for certification.
- The storage of Issuer Public Key certificates and the Payment System public keys for verification of these certificates.
- The generation of ICC public/private key pairs for use in DDA / CDA and offline PIN encipherment.
- The use of an issuer private key to certify ICC Public Keys or to sign application data for use in SDA.
- The optional use of the Certificate Revocation List (CRL) process.
- The generation and secure storage of symmetric Issuer Master Keys.
- The use of Issuer Master Keys to derive ICC Master Keys used for online authentication and secure messaging.
- The secure transport of keying material necessary for card personalization, including the import/export of Issuer Master Keys (only necessary if other entities are to perform 'on-behalf' card verification services).

6.1 Issuing Portfolio

Issuers perform the following activities during the life of a card issuance programme:

- **Preparation -** to be completed prior to any card issuance,
- **Card production (SDA)** - the steps for issuing cards employing Static Data Authentication,
- **Card production (DDA and CDA)** - the steps for issuing cards employing dynamic data authentication,
- **Card issuance** - the steps to provide cardholders with newly produced EMV cards,
- **Online transaction processing -** the procedures for supporting the ongoing use of EMV cards, including validation of the online cryptogram from EMV cards, verification of online PINs, and update of card application using issuer script commands.
- **Transaction clearing** - the processes for support of clearing and settlement of EMV transactions.
- **Obsolescence** - during the life and at the end of a card issuance programme, various keys will become obsolete and should be destroyed.

6.1.1 Preparation

The following activities need to be performed by an issuer prior to any card issuance. They also need to be performed when keys change or certificates expire.

6.1.1.1 Asymmetric (RSA) Keys

Key Pair Generation. The issuer needs to securely generate and store one or more public/private key pairs. This requires the use of protected memory in a physically secure device, utilising a random or pseudo-random number generator and primality-checking routines.

Asymmetric Keys include:

- *Issuer Key Pairs* – the private key signs card static data and ICC Public Key certificates. The public key is sent to the Payment System CA to obtain an Issuer Public Key certificate.
- *ICC Key Pairs* – the private key is personalised on the card and used for DDA/CDA and/or offline PIN decryption. The public key is signed by the Issuer Private Key to produce the ICC Public Key certificate.

[6.1] *The length |N| in bits of the RSA key moduli N (e.g. 1024, 1152, 1408, and 1984 bit keys) should be adequate for the planned active life of the cards to be issued under a given combination of scheme key, issuer key and card key. It is strongly recommended that Issuers select moduli of sufficient size that they are resistant to computational attacks, i.e., factoring of the public key modulus. For performance reasons it is recommended that the Payment System key chosen for certifying the Issuer Public Key be commensurate with the lifetime of the cards to be issued rather than the longest possible. The issuer key may then be the same length as the Payment System key or in accordance with the Bulletin recommendations for Payment System Public Keys and the card key length chosen to optimize performance balanced against risk. Cards that will be used (i.e. in the field) in 2018 or beyond are recommended to use card keys at least 1024 bits long*.

[6.2] *Issuers should periodically review the length of their RSA key pairs and when deemed no longer fit for service should move to a longer key. Equally, public exponents should also be reviewed, particularly with respect to card keys.*

Issuers should note that whilst it might be reasonably argued that it is unlikely that all the resources required to break a key would be brought to bear on a single card and thus cards are not of high concern, an issuer with a large portfolio under a given issuer key would be a more attractive target, even if only in terms of causing reputational damage from what "might be done". Such reputational damage would not be solely restricted to the Issuer in question, but would reflect badly on the payments industry in general.

[6.3] *Primes generated for RSA key generation should be chosen randomly from at least 2^{100} primes. The two primes p and q for any given key pair should differ by at least $2^{|N|/2 - 100}$.*

[6.4] *The generation of primes should be performed using a recognised secure algorithm. For guidance on prime number generation see ISO/IEC-18032.*

[6.5] *When selecting the value of the public exponents consideration should be given to performance issues. In general terminals will take longer to compute when using e = 2^{16} + 1 than when using e = 3 and this may be quite noticeable in lower end devices[2]. The formats used within EMV for both signature generation and PIN encryption are considered to be secure for both values of e when deployed in ICCs satisfying the current security evaluations.*

[6.6] *The random or pseudo-random process should be such that it is not possible to predict any key or determine that certain keys within the key space are significantly more probable than any other. For further information on random or pseudo-random number generators see ISO/IEC 18031.*

[6.7] *The physically secure device used to create the RSA key pairs and to secure the private key should be tamper responsive and satisfy the security requirements set forth in Section 9.1 Secure Cryptographic Devices (SCD).*

[6.8] *Where the Payment System permits, it is recommended that the issuer should assign separate key pairs per Issuer Identification Number (IIN) to limit the number of cards using a single key pair[3] and thus the number of cards impacted by a compromise.*

[6.9] *The Issuer Public Key should be managed in such a way that it is unchanged when sent to the CA for certification.*

[6.10] *Issuer public/private key pairs should be unique to each Payment System.*

Procedural guidance on key generation can be found in Section 8.

Receive the Payment System Public Key(s). The issuer needs to receive and securely store one or more Payment System CA Public Keys.

[6.11] *The issuer should verify the integrity and origin of the CA Public Keys in accordance with the Payment System requirements.*

Request and Receive Issuer Public Key Certificates. The issuer needs to transfer each Issuer Public Key to the Payment System CA and receive in return a signed public key certificate.

[6.12] *The Issuer Public Keys should be transferred in such a way that the Payment System CA can verify their integrity and origin.*

[6.13] *Upon receipt of a public key certificate from the Payment System CA, the issuer should verify it using the relevant Payment System Public Key.*

6.1.1.2 Symmetric (DES) Keys

Issuer Master Key Generation. The issuer needs to securely generate and store one or more master derivation keys to be used to derive the ICC Master Keys unique to

[2] Other values than 3 or 2^{16} + 1 are mathematically acceptable, but EMV terminal type approval is limited to the two recommended values, carried in 1 byte and 3 bytes respectively and therefore acceptance of other exponents is not assured.

[3] Also in the event of a single issuer private key compromise the number of cards impacted through the Certificate Revocation List (CRL) process is reduced.

each card application. This requires the use of protected memory in a physically secure device, utilising a random or pseudo-random number generator.

DES keys in the EMV specification are used for specific transaction functions. Card DES keys are derived from an Issuer master derivation key at the time of personalisation. The resultant card level keys are unique.

Issuer Master Keys include:

- *Issuer Master Key (IMK_{AC})* - used to derive the card keys that are used to generate MACs known as Application Cryptograms (AC).
- *Issuer Master Key for Secure Messaging for Integrity (IMK_{SMI})* - used to derive card keys used in the secure messaging for integrity of post issuance processes between the card and the authorisation system; e.g., card blocking, application blocking/unblocking, updating card specific data, and PIN changes.
- *Issuer Master Key for Secure Messaging for Confidentiality (IMK_{SMC})* - used to derive card keys used in secure messaging for confidentiality of post issuance processes between the card and the authorisation system; e.g. PIN change.

Issuers may use several keys for the same purpose, for example across IIN ranges. Payment systems may provide for additional key separation per IIN using issuer Master Key sets containing multiple keys. The selected key used may be identified by means of an index, stored on the card and returned in the Issuer Application Data associated with the cryptogram.

[6.14] *DES keys should be generated either inside a physically secure device protected by tamper responsive mechanisms,* ***OR***

[6.15] *by authorised personnel in component form through a process of combining components, each party generates a component that is as long as the key being generated. The key combination process takes place inside a physically secure device. Moreover, the method of combining the components must be such that knowledge of any subset of the components yields no knowledge about the key value. Where keys are generated in component form at a card personaliser, at least one component should be generated by an employee of the issuer.*

[6.16] *A random or pseudo-random process should be used such that it is not possible to predict any key or determine that certain keys within the key space are significantly more probable than any other. Further information concerning random or pseudo-random number generators can be found in ISO/IEC 18031.*

[6.17] *It is recommended that issuers assign separate keys per Issuer Identification Number (IIN) in order to limit the number of cards using a single key.*

[6.18] *A key should only be used for the cryptographic purpose for which it was intended and not for any other purpose, e.g., separate master derivation keys should be used to generate the card keys for application cryptogram generation, secure messaging integrity and secure messaging encryption.*

[6.19] *Issuer symmetric master keys should be changed periodically (e.g. annually). This does not mean that keys must be updated in the deployed card base, but rather that the issuer must be able to manage several key versions at a time, each key version being destroyed once the last card containing its derived form has expired and disputes are no longer expected.*

Further general guidance on key generation is given in Section 8.1.

- **Transfer of Issuer Master Keys**. If the issuer delegates responsibility for card personalisation or generation and verification of online cryptograms to a third party, or to different systems within his own processing center, then the issuer needs to securely transfer the issuer master key(s) used to derive the ICC keys to the third party or other system (see also Section 8.1.2).
- **Backup of Issuer Master Keys.** In some situations it may be necessary to recover the issuer master keys used for the personalisation of cards and for the authorisation of card transactions.

[6.20] *When backing-up critical master keys it is recommended that a key either be enciphered under another key of at least equal size, or maintained as two or more components, secured using the principles of dual control and split knowledge. This process should be audited.*

6.1.2 Security Counters

If key use limits are provided by security counters then values need to be established to control the situation if cards are abused.

[6.21] *Recommended values for a typical CPA contact online-offline environment are:*

- *Offline PIN Decipherment Error Counter Limit = 500*
- *ATC maximum usage = 20,000*
- *AC Session Key Counter Limit = 1,000*
- *SMI Session Key Counter Limit = 1,000.*

Note that individual Payment Systems may provide alternative values. The Offline PIN Decipherment Error Counter is only incremented when PIN decipherment fails and the counter is never reset. The AC Session Key Counter is incremented every time an AC session key is derived and is reset when an ARPC is successfully verified. The SMI Session Key Counter is only incremented when the verification of a Secure Messaging MAC fails and is never reset.

6.1.3 Card Production (SDA)

The following security relevant steps need to be performed by an issuer for each SDA card issued.

- **Static data preparation**. The issuer generates the data for card personalisation. The magnetic-stripe check value (CVC/CVV/CSC/CVN) in the Track 2 Equivalent Data should be calculated differently from the value on the magnetic-stripe in accordance with payment system rules.
- **Signing of static data**. The issuer signs selected card static data using an issuer private key to produce the Signed Static Application Data. Two of the important data elements to sign are the SDA Tag List (tag 9F4A) and consequently, the AIP (tag 82).

  [6.22] *Issuers should follow Payment System recommendations and requirements for the data elements to be signed. It is strongly recommended that these include the SDA Tag List with AIP.*

- **ICC Master Key derivation**. The ICC Master Keys for application cryptograms and script message security must be derived from the appropriate issuer master key using card data such as Application PAN and Application PAN Sequence Number.
- **PIN generation**. A Reference PIN must be generated for the IC Card if offline PIN is supported. Its value should be the same as the online PIN value.
- **Provide data to card personalisation process.** Make arrangements for the Issuer Public Key Certificate, the Certificate Authority Public Key Index, Signed Static Application Data, derived secret keys, the Derivation Key Index (if used), and Reference PIN to be personalised on the cards.

All data for personalisation needs to be transferred securely to the card personaliser for writing to the IC Card.

[6.23] *All data should be protected (e.g. by a MAC or signature) from the point at which it leaves the system that created the data, to the point at which it is stored on the card.*

[6.24] *Secret keys and PINs must in addition be kept confidential.*

6.1.4 Card Production (DDA and CDA)

The following security relevant steps need to be performed by an issuer for each card supporting dynamic data authentication or Offline Enciphered PIN.

- **Static data preparation**. The issuer generates the data for card personalisation.
- **Signing of static data**. If SDA is to be supported on a DDA/CDA card (optional) then the issuer produces the Signed Static Application Data as per 6.1.3. Note that the same static data is also signed as part of signing the ICC Public Key (see below).
- **ICC Key Pair generation**. A unique public/private key pair must be securely generated for each ICC. If a separate key is to be used for offline PIN encipherment then this must also be securely generated.
- **Signing of ICC Public Key to produce ICC Public Key Certificate**. The ICC Public Key (and associated data) together with selected card static data must be signed using one of the Issuer Private Keys to form the ICC Public Key Certificate. The private key used for signing the certificate must correspond to the Issuer Public Key Certificate being personalised on the card. A similar certificate is produced in the case that a separate key is used for offline PIN encipherment.
- **ICC Master Key derivation**. The ICC Master Keys for application cryptograms and script message security must be derived from the appropriate Issuer Secret Key using card data such as Application PAN and Application PAN Sequence Number.
- **PIN generation**. A Reference PIN must be generated for the IC Card if offline PIN is supported. Its value should be the same as the online PIN value.
- **Provide data to card personalisation process.** Make arrangements for Certificate Authority Public Key Index, Signed Static Application Data, Derivation Key Index (if used), Issuer Public Key Certificate, ICC Public Key Certificate, ICC Private Key, derived secret keys, and Reference PIN to be provided to card personalisation process.
- **Card Signature Process –** cards that use CRT (Chinese Remainder Theorem) for the signature calculation require a specific evaluation that it is implemented securely (see Section 9.2).

All data for personalisation needs to be transferred securely to the card personaliser for writing to the IC Card.

[6.25] *All data should be kept confidential by procedural processes and in addition should be cryptographically protected for integrity (e.g. by a MAC or signature) from the point at which it leaves the system that created the data, to the point at which it is stored on the card.*

[6.26] *Secret and private keys and PINs must in addition be kept cryptographically confidential (e.g. by encryption).*

[6.27] *After personalisation both the IC Card issuer and the personaliser must erase all records of the ICC private key*

6.1.5 CVM Choice

Issuers must personalise each IC card with a prioritised list of CVMs supported by the IC card (e.g. online PIN, offline enciphered PIN, offline plaintext PIN, signature, No CVM).

If the IC card supports offline PIN then the card should be personalised with the value of the cardholder's PIN (note that the PIN may also be loaded into the IC card post-issuance using script commands). If the IC card supports offline enciphered PIN then the IC card must be personalised with an RSA private key and public key certificate (see Section 6.1.4).

6.1.6 Card Issuance

The personalised ICC and PIN must be securely and separately transferred to the cardholder following the requirements of ISO 9564-1.

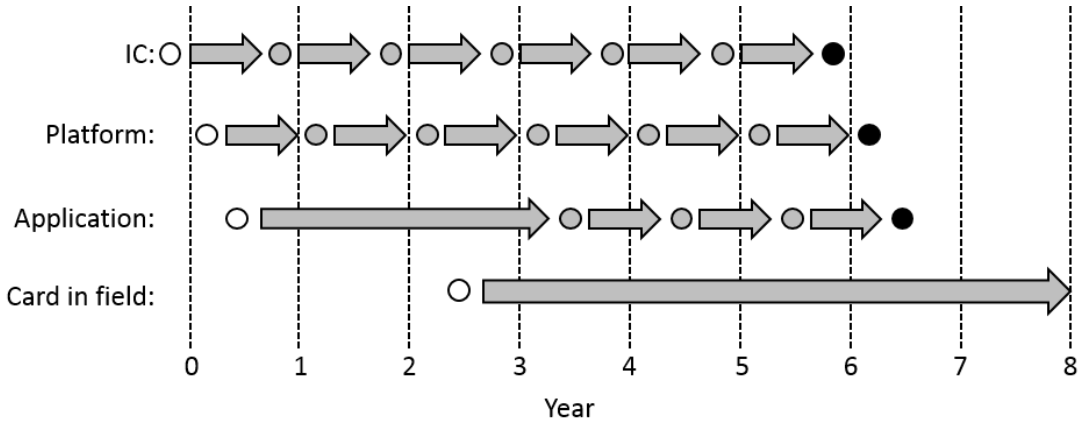
6.1.6.1 Card Lifetime

EMVCo recommends that at the time a chip card is issued, the corresponding Chip and Platform (chip + OS) Security Approvals should have been granted less than 3 years previously as depicted in the figure below. The primary driver for this recommendation is that due to advances in the threats to chip technologies, chip cards that are secure against known attacks when first designed may not remain secure against attacks discovered after issuance. The longer a chip card remains in the field, the more likely it is that an economically viable attack will become available to fraudsters.

- EMVCo Chip (IC) and Platform security approvals are granted for a period of 1 year, and annually reviewed thereafter for a maximum of 6 years, unless the approval certificate is withdrawn or the product is superseded by newer products This timeframe represents a balance between anticipated improvements in chip and platform design, new threats, and commercial considerations.
- Additionally, EMVCo Common Payment Application security approvals are granted for a period of 3 years, and annually reviewed thereafter for a maximum of 6 years.
- When choosing an EMVCo-approved Chip or Platform, the initial approval date of the product should be taken into careful consideration to ensure that the chosen product is suitable for the planned lifetime of the card.

Card inventory, issuing cycles and other related production issues all impact the effective lifetime of a card. When these factors are taken into consideration, it could be as long as 6 years from chip design to actual issuance, which would mean that the chip in a 5 year card would be in the field for approximately 11 years.

This recommendation becomes even more important in environments that use off-line authentication because of the increased incentive for successful attacks.



6.1.6.2 Privacy issues

Issuers should be aware that there may be privacy concerns with data that is available over the contactless interface of a card. It is therefore recommended that personal data sent over this interface is minimised and in particular the cardholder's name should not be transmitted.

6.2 Key Obsolescence

This section provides guidance for the destruction of obsolete keys and associated keying material.

6.2.1 Key Retention

The basic principle is that a key may be retained until it is known that it is no longer required. It should then be destroyed.

[6.28] *Issuer asymmetric keys should be retained as follows:*

- *Issuer Private Keys – until no more cards will be personalised using this key.*

[6.29] *Issuer symmetric keys should be retained as follows:*

- *Issuer master derivation keys – retain until no disputes may be expected from any card with a key derived with this key.*

6.2.2 Key Destruction

[6.30] *Obsolete keying material should be destroyed using methods that are appropriate for the medium containing the keying material(s).*

- *For keying material stored on an ICC, the chip should be physically destroyed (e.g. by punching a hole) after using any available on-chip erasure methods.*
- *For keying material stored on magnetic media, the media should be purged to appropriate magnetic remanence standards. Removable media should be physically destroyed (e.g. shredded). Note that modern hard drives have sector remapping facilities that may reduce the effectiveness of purging so physical destruction of the platters as well as purging should be considered.*

- *For keying material stored on other electronic devices, the device should be purged by a security deletion method that ensures adequate deletion with no residual memory. Consideration should be given to the physical destruction of the device, including and in particular the memory chip or chips.*
- *Obsolete Keys stored in a SCD should be erased by the SCD security mechanisms.*
- *Issuers should encourage their cardholders to destroy expired cards (e.g. by shredding) but it must be expected that many cards will not be disposed of on expiry and they would continue to function if activated, although live terminals would not accept transactions from them.*

[6.31] *An independent third party; e.g., an internal auditor, should witness the destruction of keying materials, documenting the process. The resultant documentation should be retained for a period consistent with the issuer's documentation retention policies.*

6.3 Certificate Revocation Lists

If an issuer private key is compromised, fraudulent cards that pass offline data authentication could be created using this key. They would be accepted at terminals for offline transactions until the certificate for the compromised key expires, which for some keys could be many years into the future.

The impact of the issuer key compromise can be mitigated through the use of Certificate Revocations Lists (CRLs). With CRLs each EMV terminal is required to accommodate 30 negative certificate entries per Payment System. When processing the issuer certificate for offline data authentication, the terminal checks whether the certificate read from the card is listed amongst the CRL entries and if so, offline data authentication will fail.

It should be noted that in addition to any fraudulent cards, all legitimate cards made with the key will also be identified by the terminal and rejected for offline working. Unless there is high confidence that terminals will be able to go online and maintain acceptance, the effected cards should be re-issued prior to adding the certificate to the CRL.

[6.32] *In the event an issuer detects that their private (signature) key corresponding to their Issuer Public Key certificate has been compromised, it is strongly recommended that the compromised key pair be replaced and re-issuance schedules for cards with public key certificates created by the compromised key be established and implemented to mitigate the issuer's potential fraud risk.*

[6.33] *The decision whether to request posting of a certificate on the CRL needs to be taken against the background of the level of fraud, the time before the compromised certificate expires and the card settings and terminal capabilities for going online when offline data authentication fails.*

Note that CRLs may not be universally available in the general acceptance environment.

7 Issuer Transaction Processing

7.1 Online Transaction Processing

The following security relevant steps need to be performed by an issuer for each online transaction.

- **Cryptogram exchanges**. As part of the security procedures surrounding a card transaction at a merchant (with on-line capability), the IC Card or the merchant terminal may require an online authorisation including card authentication.
  - Card authentication involves passing an Application Cryptogram (ARQC) from the card to the issuer. The card generates the cryptogram by enciphering card, terminal, and transaction data with its ICC Master Key (that was derived during personalisation from the Issuer Master Key) or session key derived from the ICC Master Key. The issuer validates the ARQC using the same key.
  - The issuer may generate a response cryptogram (ARPC) using the same secret key used for ARQC validation. The ARPC is the result of encrypting the ARQC and additional data such as the Authorisation Response Code or the Card Status Update. The card validates the ARPC to prove that the response came unaltered from the valid issuer.
- **Secure messaging**. As part of the overall card management process, *secure messages* (cryptographically protected using derived ICC keys) may be sent from the issuer to IC Card to update data on the card. These updates can change the PIN, reset the PIN Try Counter, block or unblock the application, block the card, and change card risk management data.
- **Cardholder verification**. In some online environments (e.g. ATMs) the issuer will be required to verify an encrypted PIN transmitted with the authorisation message.

7.2 Fraud Detection

7.2.1 By Issuer

The issuer should use the chip data in the authorisation request in its risk evaluation. The results of offline chip processing are shown in the authorisation request's Terminal Verification Results (TVR) and Issuer Application Data (IAD). Negative results such as offline PIN failures should be considered in the authorisation decision. Other fields in the authorisation request, such as the ATC, should be evaluated for reasonableness. Incoming values that were personalised on the card such as the Application Interchange Profile (AIP) should be checked to assure they match the personalised value.

[7.1] *Issuers should proactively monitor EMV transactions to detect fraudulent use of IC Cards and terminals and possibly counterfeit cards. Some areas to monitor could include:*

- *Offline indicators sent online – when making the decision regarding whether to approve or decline the transaction, consider using indicators that are sent online in the Issuer Application Data and other online data elements. These*

may show offline data authentication failures, offline PIN failures, bypass of PIN entry, and other offline occurrences that may be indicators of fraud.

- *Online cryptograms – validate the cryptogram for both the authorisation and clearing messaging to assure the validity of the card. 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.*
- *ATC – check for duplicate ATCs, ATCs that are smaller than the ATCs of previous transactions and large gaps between the current ATC and the ATCs of recent/previous transactions. Please note that duplicate ATCs may legitimately occur with authorisation request repeats and authorisation requests for subsequent validation of the online PIN when the previously entered PIN has failed validation. An ATC received out of sequence may be caused by a deferred authorization.*
- *Magnetic stripe reads of chip cards at chip terminals – these can be an indicator of fraud but also may be due to problems with the chip or chip reader.*
- *Inconsistent chip data – check that the chip data received in the authorisation and clearing messages is consistent with what was personalised or updated on the card.*
- *Script commands – assure that chip-read transactions are not being received from cards or applications that were previously blocked using script commands.*

[7.2] *The following actions should be taken to prevent fraud and credit losses:*

- *Block the application or card when the card is stolen.*
- *Adjust velocity limits to reflect the cardholder's current offline purchasing power.*

[7.3] *To assist in dispute resolution, issuers should retain online transaction data in accordance with payment system recommendations and requirements. Issuers who have issued cards that log transaction data should consider using these logs to assist in dispute resolution.*

7.2.2 By Card

Issuers have the option to deploy card products that request transactional information such as CVM Results which can then be checked against the card's understanding of the transaction status. The results can be used as part of the card's risk management.

7.3 Transaction Clearing

The following security relevant step needs to be performed by an issuer for each approved transaction.

- **TC processing**. During all approved IC Card transactions, the IC Card provides the merchant terminal with a Transaction Certificate (TC) generated using an IC Card secret key. This TC is passed by the merchant to the acquirer and is then passed from the acquirer to the issuer as part of the clearing process. It is recommended that issuers verify the correctness of the TC as part

of clearing and settlement. Such checks would help detect potential SDA clones and might identify systemic problems that cause unjustified failures in the TC validation.

- **EMV Dispute Processing**. In general the overall issuer responsibilities for chargeback and dispute resolution are the same for both magnetic stripe and EMV chip card transactions. However migration from magnetic stripe cards to EMV cards and any associated liability shifts may provide additional information for use in dispute resolution.

  It is generally acknowledged that as EMV migration matures and stabilizes then the corresponding number of customer disputes should be reduced due to the improved authentication delivered by the technology. However issuers should take note of those response codes that are impacted by EMV transactions with specific attention to detail on EMV transactional data and the corresponding cryptography that supports authentication and data integrity services within the protocol. All issuers should look to their respective payment systems for the specifics of EMV-related dispute rules and reason codes.

- It is not the intent of this document to indicate which specific chargeback codes for which specific payment system are impacted or how they are impacted by EMV transactions. Rather we have noted several chargeback codes that in all likelihood are impacted by EMV transactions. Issuers should work with their respective payment systems to ensure that the correct procedures, process and liability shift for chargeback codes are in place for EMV disputed transactions.

- Issuer Best Practices for EMV disputes:

  1) train back-office staff on EMV transactional flows and data elements.

  2) ensure back-office staff are aware of the reason codes that are impacted by EMV and those that are not.

  3) create EMV back-office job aids or tool kits to assist staff when reviewing EMV related disputes and chargebacks.

  4) automate EMV dispute analytics as much as possible. Dispute resolution may require analysis of several EMV related data elements.

  5) several data elements and processes are critical in supporting EMV dispute resolution. Especially important is the Application Cryptogram (ARQC or TC) which assures that the genuine card was present during transaction processing and that the data used in the cryptogram generation was delivered unaltered from the card to the issuer. Other data elements that could be used in dispute resolution include ATC, TVR, CVM Results, and Issuer Application Data.

  6) to assist in dispute resolution issuers should retain online transaction data in accordance with payment system recommendations and requirements. Typically this will include ARQCs, TCs and supporting data.

8 Key Management Practices

A secure implementation depends on the issuer securely generating and managing the cryptographic keys required by the specification according to the following general requirements.

Further guidance on Key Management for retail banking may be found in ISO 11568.

8.1 Key Generation – General Guidance

The following gives some general guidance on procedures that should be considered for use in key generation and is applicable to both asymmetric and symmetric key generation. A number of the processes require random bit generation and/or prime number generation.

Further guidance may be found in ISO/IEC 18031 for random bit generation and ISO/IEC 18032 for prime number generation.

[8.1] *All key generation should take place according to a process that is defined in advance and all necessary role holders must be present at the appropriate time. The exact process used depends on the systems that are in place. The environment in which key generation is to take place must be secure. The process must be appropriate to the methods and equipment being used. The configuration of the equipment must not leave it exposed to attacks such as electromagnetic eavesdropping.*

[8.2] *All key material should be clearly identified as to its purpose and lifetime. This is to ensure that:*

- *keys are not reused for other purposes.*
- *test and live key material is not mixed.*
- *keys are replaced when required.*

[8.3] *The procedure for the creation of each type of key should detail the location, methods, equipment and role holders needed for the session. The procedure should be explicit and easy to follow for all role holders, bearing in mind that key component holders (often referred to as key custodians) may not have intimate knowledge of the processes being used and may have performed them infrequently.*

[8.4] *The procedure should state who is required to authorise its execution. It should define how the system for key generation is established and what actions are needed after use (if any) for secure purging of the system. It should state how key components are to be stored and by whom.*

[8.5] *The procedure should also define how many copies of the keys are to be made and how they are to be protected. Note that at least two copies of most keys will be needed – one for production use and one as an offsite backup copy. The backup copy should be protected as least as well as the production copy.*

[8.6] *The procedure should be largely self-documenting such that an auditor can use a copy of the procedure, original authorisations and audit records signed at the time by the participants – after the event – to conclude that the keys have been securely generated, stored and used.*

[8.7] *The integrity of keys must be ensured. Check digits within an HSM and MACs outside the HSM are suitable mechanisms for this purpose. If used, check digits must be calculated for the full component or for the actual key. Check digits calculated for components can be transferred with the component.*

[8.8] *If any secret or private cryptographic keys are found to exist outside of a physically secure device or the components of a cryptographic key are known or suspected to have been under the control of a single individual, then the keys are to be considered to have been compromised.*

8.1.1 RSA Key Transport and Storage

Public keys should be transported in a manner that allows for verification of their integrity and authenticity.

[8.9] *Public keys should be transported within a certificate, protected by a Message Authentication Code (MAC) created with a key used only for this purpose, or under dual control (i.e. by separate and independent transfer of a check value on the public key). Suitable MAC algorithms are described in ISO 16609.*

[8.10] *Recipients of a public key should have the appropriate means to verify its origin and integrity i.e., trusted root key, shared MAC key or check value.*

[8.11] *Private keys must be secured and transported in such a manner that guarantees their integrity and secrecy. Transportation mechanisms may include:*

- *a secure cryptographic device.*
- *encipherment using a symmetric algorithm of at least equal cryptographic strength to the key being protected.*
- *as key components using the principles of dual control and split knowledge.*

[8.12] *Business resumption strategies should require that copies of public/private keys be made and secured. These keys are necessary in the event of a catastrophic system failure. Back-up copies of system keys are to be secured using the same principles as described in the previous bullet.*[4]

8.1.2 DES Key Transport and Storage

It may be necessary to transport and store DES keys. Examples include the transport of DES keys from the issuer's site to that of a third party processor or card personalisation vendor. When DES keys are being transported or stored, the following measures will limit the potential for the compromise of the data.

[8.13] *Cleartext DES keys should be securely transferred and stored under the protection of a secure token or smart card.*

[8.14] *DES keys should only be transported or stored outside the protected memory of a secure token or smart card in one of the following ways:*

[4] Note that in case of an unintended deletion of an Issuer private key there need be no interruption for existing cards and certificates, but the issuer will need to generate and certify a new key.

- *In the form of two or more components using the principles of dual control and split knowledge, or*
- *As a cryptogram, created by a transport or storage key that has been securely established by the parties.*

8.2 Key Custodians - Practices and Responsibilities

8.2.1 Appointment of Key Management Personnel

[8.15] *Personnel responsible for the management of encryption keys and key components, secure tokens and other keying devices must be designated by the different parties involved; i.e., issuers, third party processors, and card personalisation vendors.*

[8.16] *When designating individuals to be responsible for the custodial control of keying material or secure tokens, sufficient controls should be implemented to assure that no single individual or unauthorised individual can obtain access to the data comprising a cryptographic key or secure token.*

[8.17] *Key custodians should be trusted employees and never temporary help or consultants.*

[8.18] *To assure service continuity alternate personnel should also be identified as "back-ups" to the primary key custodians. The criteria used to select "back-up" custodians should be the same as used to select the primary custodians.*

8.2.2 Functions of Key Custodians

Key custodial responsibilities are important and a fundamental part of an issuer's security protocol. The keying material that will be managed by these persons represent the most important keys in the cryptographic applications for an issuer's card program.

Each issuer is encouraged to review its internal key management procedures and the roles of those individuals relative to the following practices.

- The responsibilities of the key management personnel (key custodians) include the control of keying materials, verification of the materials, and their secure storage.
- Key custodians are responsible for the:
  - Receipt and secure storage of key components and secure tokens, including authentication of the keying material and confirmation of receipt.
  - Maintenance of records or logs tracking access to and usage of keying material; including times of access, date, purpose, and return to secure storage.
  - Verification of all transfers of keying material to other designated individuals outside the issuer's control.
  - Witnessing the destruction of old/obsolete key components.

- Entering of keying material into secure cryptographic modules as required from time to time.
- Directing and overseeing the destruction of obsolete keying materials.

8.2.3 Procedures for Shipping Keying Materials to Third Party Agents

[8.19] *A two-part form should accompany the forwarding of all keying material to a party other than the originator. This form should identify the sender and the materials being sent to the receiver. The originator should sign the form. Immediately upon receipt of the materials, the receiving custodian should verify the contents against the form and should sign and return one part of the form to the originator.*

[8.20] *When forwarding key components and other cryptographic data to third parties, each component should be sent to a different individual using different delivery services for each of the various components; i.e., registered mail, express mail, standard mail or air courier service. Where the keying material is secured using a secure token or smart card, registered mail is sufficient.*

[8.21] *The receiver of the keying material should be pre-notified of the forwarding of keying materials.*

8.2.4 Physical Procedures for Securing Keying Material at Third Party Agents

[8.22] *Upon receipt the responsible key custodian must immediately examine the shipping package for tampering and must verify the contents.*

[8.23] *If the receiving custodian has any uncertainty regarding the integrity of the keying material, the sender is to be immediately notified. The sender, in consultation with the receiving party, will decide the future status of the keying material. The basis for any decision regarding the continued use of the keying materials should be documented and retained by the two parties.*

[8.24] *If the hard copy keying material is to be retained for any period of time, the individual hardcopy components, secure token, or smart card should be secured in a serialised tamper evident envelopes stored in physically secure locations. The envelopes for individual hardcopy components should be stored in separate physically secure locations without shared access.*

[8.25] *The serialised tamper evident envelopes should be continually protected in a physically secured container, accessible only to the designated key custodian or alternate. Each access of the keying material should be logged, including time, date, envelope serial number, purpose, and signature. These logs are to be made available to any appropriate requesting authority.*

[8.26] *Keying materials should never be maintained outside of tamper evident envelopes and their physically secure environments longer than necessary for the task requiring the access.*

9 Hardware and Software Security Considerations

9.1 Secure Cryptographic Devices (SCD)

The term SCD is used in ISO standards to describe a broad array of equipment used to provide certain protections for cryptographic operations. These prevent the disclosure and/or modification of sensitive information. See ISO 13491. At the high end of the array of SCDs are the hardware security modules (HSMs) connected to host processors, personalisation devices and authorisation systems. At the opposite end of the spectrum are the secure tokens and integrated circuit cards (ICCs).

The materials contained in this part of the guidelines are to be used to enhance the reader's knowledge and understanding of the general security attributes required by the Payment Systems as a part of their risk management practices.

9.1.1 Tamper Resistance Requirements

Tamper resistance can be separated into two security domains, (1) physical, and (2) logical.

9.1.1.1 Physical Security Attributes

Physical security consists of the following attributes:

- Protection against penetration.
- Protection against unauthorised modification.
- Protections against the monitoring of side-channel emissions (e.g. electro-magnetic, power variation) resulting from operation of the device.

9.1.1.2 Logical Security Attributes

Logical security consists of the following attributes:

- Verification of software integrity.
- Function set design to assure that no single or combination of device functions will result in the disclosure of sensitive information.
- Mechanisms to assure that cryptographic keys are only used for their intended purpose.
- Protections against the monitoring of side-channel effects (e.g. timing variation, sequencing) resulting from operation of the device.
- Sensitive state operations to require dual control.
- Techniques for the authentication of data and software updates.

9.1.2 Key Storage – General Guidance

The following gives some general guidance on key storage issues that is applicable to both asymmetric and symmetric key storage:

9.1.2.1 Hardware versus Software

It is common to think of keys being stored in a 'hardware' location such as an HSM or a 'software' location such as on a host computer system. In reality, these are just shorthand notations for a set of assumptions about how a key may be protected from being compromised.

Keys are generally stored on magnetic media such as disks, volatile silicon memory or long term silicon memory such as is used on ICCs. Keys are protected by a variety of physical means such as the tamper resistance of an HSM or ICC and the logical protection of an operating system in HSMs (or SCDs), ICCs and host computers.

[9.1] *Key storage on magnetic media should be regarded as temporary. The lifetime of removable media should be assumed to be no more than a couple of years.*

[9.2] *Key storage in silicon non-volatile memory has a much longer potential life – typically at least 10 years.*

[9.3] *However keys are stored, they need protection from compromise. Devices that store keys should be physically secured – for example in a tamper evident package in a safe. Devices that use keys should check the key integrity before use. This is especially important for keys stored enciphered on a database. A non-cryptographic checksum may be used to check a key stored on a secure device such as an ICC, but a key stored in a manner that is open to alteration, such as in a database, should be protected by a cryptographic MAC.*

[9.4] *USB devices, such as memory sticks, used to store key material should be from a trustworthy source and used only for this purpose. In addition the recipient computer should have auto-run disabled.*

9.1.2.2 HSMs

In general, an HSM will contain separate storage and processing devices, and key material will pass across internal hardware busses. For this reason it is important that the HSM clear (or 'zeroise') its memory when compromise is detected. It is also important that the hardware design of the HSM addresses electromagnetic radiation.

Plug-in boards in a specialised PC used for cryptographic processing should be viewed as a form of HSM and similar levels of protection are required. Note however that the main reason for using a cryptographically secure device is to protect keys. If the host system that uses the HSM is itself insecure then it may be easier for an attacker to compromise that system's software functions and ignore the HSM.

It is important that HSMs used for Application Cryptogram processing cannot be mis-used, even by authorised personnel, to perform unauthorised cryptographic calculations (e.g. bogus ARQC verification or ARPC generation).

Card keys are derived and scripts constructed inside an HSM typically using calls across an API. Because the block formats can vary (e.g. padding), in some instances the API may allow flexibility by means of input data offsets. Such implementations

could in theory be abused by an inside attacker able to manipulate the software calling the API and to collect and collate the responses. A similar situation may also apply if the same process is used to encrypt keys for transport and management purposes.

[9.5] *A general purpose personal computer or similar insecure device should never be used to generate keying materials.*

[9.6] *Access controls should be enforced for HSM usage, with strong software management controls, logging and constant monitoring. APIs should only support those data formats that will actually be used.*

For further guidance on HSM security, see ISO 9564, ISO 13491 and PCI Security Standards Council www.pcisecuritystandards.org.

9.2 ICC and ICC Application Security

Issuers are responsible for selecting an appropriate IC platform and corresponding ICC. To protect the interests of the Payment System and those of the issuer, EMVCo provides Security Approvals for ICs and CPA ICCs. Payment Systems provide their own security approvals for other ICCs.

Unlike HSMs which are designed to be located within a secure environment, ICCs are designed for use in an unprotected environment. In general they rely on tamper resistance to protect sensitive information. EMVCo provides IC Security Guidelines to all IC vendors that submit ICs for the Security Evaluation process.

Management of the security for the card application includes the IC platform, application development security and post issuance application data security. Whilst it is beyond the scope of these guidelines to provide specific details, the following is a list of the more important considerations that should be used in the development and implementation of an EMV based application and that are addressed in the associated IC platform security evaluations.

- Application development will take place in a tightly managed and audited environment to eliminate undesirable features such as rogue code and unwanted test functionality.
- Cryptographic functions will be designed to minimize key leakage from timing measurements or power analysis.
- The use of individual keys will be restricted by means of session key processes and security counters.
- Post issuance functionality will be restricted to eliminate unauthorized data manipulation and cards will employ data integrity checks on a regular basis.

Cards need to be resilient to a variety of attacks on the RSA calculations, targeted at disclosing the card private key. These include timing, power analysis, fault analysis and perturbation attacks. From a security perspective, both the use of the Chinese Remainder Theorem (CRT) as a technique for improving calculation performance[5] and the choice of public exponent have an impact. For example when using e = 3, the most significant half of the private exponent can be calculated mathematically and thus cards need to have high resistance to attacks which can reveal the remaining bits of the key. Cards using ICs that are on the approved EMVCo security evaluation

[5] CRT accelerates RSA operations by performing the modular exponentiations based on the two primes that make up the RSA modulus, rather than the modulus itself.

product list will have been rated “high assurance” for all RSA implementations and exponents that they support. Therefore issuers only need to concern themselves from the point of view of performance, not security.

[9.7] *Issuers should only use EMVCo approved IC platforms.*

[9.8] *Issuers should pay extra attention to the expiry date of the IC to ensure the IC has an appropriate expiry date for the desired lifetime of the card (refer to Section 6.1.6.1)*

[9.9] *Application software should be developed in a controlled environment. This environment should not only be physically secure, but should be managed using procedures that ensure the integrity of the application and its source code.*

[9.10] *Data to be used for personalisation should be managed in accordance with existing data and IT security policies of the issuer. Secret application data, including cryptographic keys, reference PINs and other data designated as secret, should never be accessible in plaintext form.*

[9.11] *Secret application data should not be accessible outside the routines specified by the application. Consequently, no undocumented methods of updating, resetting or incrementing data should be permitted.*

Annex A Cryptographic Algorithms – Worked Examples

This chapter provides detailed examples of the implementations of CPA cryptographic algorithms.

A.1 - Introduction

This chapter provides examples of various cryptographic operations implemented in respect of CPA applications. They include:

- Purchase with Online Authorization:
  - Card key derivation
  - EMV common session key derivation
  - ARQC computation
  - ARPC computation
  - TC computation
- PIN Change:
  - Card key derivation for encryption
  - EMV common session key derivation for encryption
  - PIN block encryption
  - Card key derivation for script MAC
  - EMV common session key derivation for script MAC
  - Script MAC computation
- Static Data Authentication:
  - Verification of Signed Static Application Data (SSAD)
- Dynamic Data Authentication:
  - Verification of ICC Public Key Certificate
  - Derivation of Card key for IDN
  - IDN computation
  - Generation of Signed Dynamic Application Data (SDAD) for DDA
  - ICC Public Key Certificate verification
  - Verification of SDAD for DDA
- Combined DDA / AC Generation:
  - Encryption of offline counters
  - Generation of SDAD for CDA
- Offline PIN Encipherment:
  - Verification of ICC PIN Encipherment Public Key Certificate
  - Encryption and Decryption of PIN

For definitions of the cryptographic functions see EMV Book 2.

A.2 - Notation Used

In the following examples, data input and intermediate values are shown as a succession of byte values against a light gray background. For example:

```
13 33 90 00 00 61 65
```

Principal values, and the end results of calculations, are shown as a succession of byte values against a dark gray background, enclosed by a box. For example:

```
05 01 09 08 56 D3 96 58 A2 EE D9 B1 BB BB BB BB
BB BB BB BB BB BB BB BB BB BB BB BB BB BB BB BB
BB BB BB BB BB BB BB BB BB BB BB BB BB BB BB BB
BB BB BB BB BB BB BB BB BB BB A0 B1 C2 D3
```

DES3(K)[D] denotes the encryption of the data block D using key K and the Triple DES encryption algorithm (in the Electronic Code Book mode of operation).

CBC_DES3(K)[D] denotes the encryption of the data D (which must be a whole number of blocks) using key K and the Triple DES encryption algorithm (in the Cipher Block Chaining mode of operation).

MAC(K)[M} denotes the 8-byte MAC computed on the message M using key K and the MAC algorithm specified in [EMV2} Annex A1.2 and ISO/IEC 9797-1 Algorithm 3 with DES as the block cipher. Note that padding of M is performed as part of the MAC algorithm.

A.3 - Purchase with Online Authorization

This section illustrates the cryptographic computations for a purchase using online authorization, specifically:

- Card Master Key Derivation
- Session Key Derivation using EMV common session key derivation method
- ARQC computation
- ARPC computation
- TC computation

A.3.1 - Card Master Key Derivation

Key

The Issuer Master Key for AC computation: IMK_{AC}

Value:
9E 15 20 43 13 F7 31 8A CB 79 B9 0B D9 86 AD 29

Input

The concatenation of:

- The rightmost 7 bytes of the PAN (format n; if it is short, left-pad with 0-nibbles to obtain 7 bytes).

  For PAN 5413339000006165:

  ```
  13 33 90 00 00 61 65
  ```

- The PAN Sequence Number (1 byte, format n).

  For PAN Sequence Number 0:

  ```
  00
  ```

The input is:

```
13 33 90 00 00 61 65 00
```

Output

MK_{AC} = DES3(IMK_{AC})[Input] || DES3(IMK_{AC})[Input ⊕ 'FF FF FF FF FF FF FF FF']

Although this does not affect the computations using this key, the result is submitted to the DES parity forcing rule (each byte has an odd number of 1-bits, achieved by appropriately setting the least significant bit of each byte). Some implementations will refuse a DES key that does not conform to this parity forcing.

```
08 DF 34 25 32 20 A7 20 EF F2 C1 34 38 52 E6 3D
```

A.3.1.1 - Card Master Key Derivation with Long PAN

The following is an example of card key derivation according to Option B where the PAN is longer than 16 digits and so SHA-1 pre-processing is performed. Note that the card key so derived is not used elsewhere in these examples.

Key

The Issuer Master Key for AC computation: IMK_{AC}

Value:
9E 15 20 43 13 F7 31 8A CB 79 B9 0B D9 86 AD 29

Input

The concatenation of:

- The PAN.

  For PAN 541333900000006165:

  ```
  54 13 33 90 00 00 00 61 65
  ```

- The PAN Sequence Number (1 byte, format n).

  For PAN Sequence Number 0:

  ```
  00
  ```

The input to the SHA-1 hash algorithm is:

```
54 13 33 90 00 00 00 61 65 00
```

The output from the SHA-1 hash algorithm is:

```
8A A7 35 8F 06 B2 2A 83 11 8D BE 1D A5 EB 37 3D
5C BB 8D E1
```

The input to the DES process is:

```
87 35 80 62 28 31 18 15
```

Output

MK_{AC} = DES3(IMK_{AC})[Input] || DES3(IMK_{AC})[Input ⊕ 'FF FF FF FF FF FF FF FF']

Although this does not affect the computations using this key, the result is submitted to the DES parity forcing rule (each byte has an odd number of 1-bits, achieved by appropriately setting the least significant bit of each byte). Some implementations will refuse a DES key that does not conform to this parity forcing.

```
76 7C 58 7A 61 4C C7 29 97 2C 92 E3 92 EC A4 5B
```

A.3.2 - Session Key Derivation for AC

Key

The Card Master Key MK_{AC} for AC computation as derived above:

```
08 DF 34 25 32 20 A7 20 EF F2 C1 34 38 52 E6 3D
```

Input

For AC computation, the input is the concatenation of:

- The Application Transaction Counter (ATC):

```
34 56
```

- Six zero bytes:

```
00 00 00 00 00 00
```

Concatenation:

```
34 56 00 00 00 00 00 00
```

Output

The concatenation of:

- The Triple-DES Encryption of the input with the third byte replaced by ‘F0’
- The Triple-DES Encryption of the input with the third byte replaced by ‘0F’

This amounts to:

SK_{AC} = DES3(MK_{AC})[(ATC || ‘F0’ || ‘00’ || ‘00’|| ‘00’|| ‘00’|| ‘00’)] || DES3(MK_{AC})[(ATC || ‘0F’ || ‘00’ || ‘00’|| ‘00’|| ‘00’|| ‘00’)].

```
18 20 25 BA 4F AB 32 F5 A6 3A 1B A5 E6 84 5D 4E
```

Note **The result can be submitted to parity forcing, like the Card Master Keys. This is not shown here as this is entirely internal to the ICC, and the computation is not affected. This is not recommended for security reasons.**

A.3.3 - ARQC Generation

Key

The Session Key SK_{AC} as derived above:

```
18 20 25 BA 4F AB 32 F5 A6 3A 1B A5 E6 84 5D 4E
```

Input

The input is the concatenation of the values of:

- Amount (Authorized):

```
00 00 00 01 00 00
```

- Amount (Other):

```
00 00 00 00 10 00
```

- Terminal Country Code:

```
08 40
```

- Terminal Verification Results:

```
00 00 00 10 80
```

- Transaction Currency Code:

```
08 40
```

- Transaction Date:

```
98 07 04
```

- Transaction Type:

```
00
```

- Unpredictable Number:

```
11 11 11 11
```

- Application Interchange Profile (AIP):

```
58 00
```

- Application Transaction Counter:

```
34 56
```

- Issuer Application Data:

```
0F A5 00 A0 38 00 00 00 00 00 00 00 00 00 00 00
0F 01 00 00 00 00 00 00 00 00 00 00 00 00 00 00
```

Concatenation:

```
00 00 00 01 00 00 00 00 00 00 10 00 08 40 00 00
00 10 80 08 40 98 07 04 00 11 11 11 11 58 00 34
56 0F A5 00 A0 38 00 00 00 00 00 00 00 00 00 00
00 0F 01 00 00 00 00 00 00 00 00 00 00 00 00 00
00
```

Output

ARQC = MAC(SK_{AC})[Input]:

```
C2 00 39 27 0F E3 84 D5
```

A.3.4 - ARPC Generation

Key

The Session Key SK_{AC} as derived above for the ARQC generation:

```
18 20 25 BA 4F AB 32 F5 A6 3A 1B A5 E6 84 5D 4E
```

Input

The input is the concatenation of:

- The ARQC:

```
C2 00 39 27 0F E3 84 D5
```

- The Card Status Update (CSU):

```
00 82 00 00
```

Concatenation:

```
C2 00 39 27 0F E3 84 D5 00 82 00 00
```

Output

ARPC = MAC4(SK_{AC})[Input], where MAC4 denotes the first 4 bytes of the MAC

```
90 EF 47 7F
```

A.3.5 - TC Generation

Key

The Session Key SK_{AC} as derived above:

```
18 20 25 BA 4F AB 32 F5 A6 3A 1B A5 E6 84 5D 4E
```

Input

The input is the concatenation of the (current, refreshed) values of:

- Amount (Authorized):

```
00 00 00 01 00 00
```

- Amount (Other):

```
00 00 00 00 10 00
```

- Terminal Country Code:

```
08 40
```

- Terminal Verification Results:

```
00 00 00 10 80
```

- Transaction Currency Code:

```
08 40
```

- Transaction Date:

```
98 07 04
```

- Transaction Type:

```
00
```

- Unpredictable Number:

```
11 11 11 11
```

- Application Interchange Profile (AIP):

```
58 00
```

- Application Transaction Counter:

```
34 56
```

- Issuer Application Data:

```
0F A5 00 62 38 00 00 00 00 00 00 00 00 00 00 00
0F 01 00 00 00 00 00 00 00 00 00 00 00 00 00 00
```

The concatenation:

```
00 00 00 01 00 00 00 00 00 00 10 00 08 40 00 00
00 10 80 08 40 98 07 04 00 11 11 11 11 58 00 34
56 0F A5 00 62 18 00 20 00 00 00 00 00 00 00 00
00 0F 00 00 00 00 00 00 00 00 00 00 00 00 00 00
00
```

Output

TC = MAC(SK_{AC})[Input]:

```
60 2D 63 CC BF DE 58 FE
```

A.4 - PIN Change

This section illustrates the cryptographic computations performed by a CPA issuer for performing PIN change using secure messaging, specifically:

- Card Master Key derivation for encryption
- EMV common session Key derivation for encryption
- PIN block encryption
- Card Master Key derivation for Script MAC
- EMV common session Key derivation for script MAC
- Script MAC computation

A.4.1 - Card Master Key Derivation for Script Encryption

Key

The Issuer Master Key for Secure Messaging for Confidentiality IMK_{SMC}.

Value:

```
CE 29 3B 8C C1 2A 97 73 79 EF 25 6D 76 10 94 92
```

Input

The concatenation of:

- The rightmost 7 bytes of the PAN (format n; if it is short, left-pad with 0-nibbles to obtain 7 bytes).

  For PAN 5413339000006165:

  ```
  13 33 90 00 00 61 65
  ```

- The PAN Sequence Number (1 byte, format n).

  For PAN Sequence Number 0:

  ```
  00
  ```

The input is:

```
13 33 90 00 00 61 65 00
```

Output

MK_{SMC} = DES3(IMK_{SMC})[Input] || DES3(IMK_{SMC})[Input ⊕ 'FF FF FF FF FF FF FF FF']

Although this does not affect the computations, the result is submitted to the DES parity forcing rule (each byte has an odd number of 1-bits, achieved by appropriately setting the least significant bit of each byte). Some implementations will refuse to use a DES key that does not conform to this parity forcing.

```
DA 83 49 40 98 92 F2 31 61 52 BF 80 7F 46 B6 23
```

A.4.2 - Session Key Derivation for Script Encryption

Key

The Card Master Key MK_{SMC} for Secure Messaging Integrity as derived above:

```
DA 83 49 40 98 92 F2 31 61 52 BF 80 7F 46 B6 23
```

Input

The input is the last AC, in this case the ARQC:

```
14 1D 34 65 C6 85 7C 46
```

Output

The concatenation of:

- The Triple-DES Encryption of X = "the input with the third byte replaced by 'F0' "
- The Triple-DES Encryption of Y = "the input with the third byte replaced by '0F' "

This amounts to:

SK_{SMC} = DES3(MK_{SMC})[X] || DES3(MK_{SMC})[Y]:

```
F3 53 01 FF 7A CF 75 9C AC FF 35 56 01 D9 9E A4
```

Note **The result can be submitted to parity forcing, like the Card Master Keys. This is not shown here as this is entirely internal to the ICC, and the computation is not affected.**

Computation

The computation is described as a 2-block ECB (Electronic Code Book mode) Encryption of "X || Y".

A.4.3 - Script Encryption

Key

The Session Key SK_{SMC} for Secure Messaging Confidentiality as derived above:

```
F3 53 01 FF 7A CF 75 9C AC FF 35 56 01 D9 9E A4
```

Input

For Secure Messaging for Confidentiality (Script Encryption), the input consists of the concatenation of the plaintext command data, in this case the PIN block, padded according to ISO 9564-3. For PIN 12345:

```
25 12 34 5F FF FF FF FF 80 00 00 00 00 00 00 00
```

Output

The Triple-DES CBC encryption of the input:

EncData = CBC_DES3(SK_{SMC})[Input]:

```
DB 8D 1E 79 82 52 56 06 32 70 3D A7 2F B1 9B EA
```

A.4.4 - Card Master Key Derivation for Script MAC

Key

The Issuer Master Key for Secure Messaging Integrity: IMK_{SMI}.

Value:

```
46 64 94 2F E6 15 FB 02 E5 D5 7F 29 2A A2 B3 B6
```

Input

The concatenation of:

- The rightmost 7 bytes of the PAN (format n; if it is short, left-pad with 0-nibbles to obtain 7 bytes).

  For PAN 5413339000006165:

  ```
  13 33 90 00 00 61 65
  ```

- The PAN Sequence Number (1 byte, format n).

  For PAN Sequence Number 0:

  ```
  00
  ```

The input is:

```
13 33 90 00 00 61 65 00
```

Output

MK_{SMI} = DES3(IMK_{SMI})[Input] || DES3(IMK_{SMI})[Input ⊕ 'FF FF FF FF FF FF FF FF']

Although this does not affect the computations, the result is submitted to the DES parity forcing rule (each byte has an odd number of 1-bits, achieved by appropriately setting the least significant bit of each byte). Some implementations will refuse to use a DES key that does not conform to this parity forcing.

```
04 40 7F 0E 7F CD 4A 02 FD 7F 3B 75 EF 97 3E 52
```

A.4.5 - Session Key Derivation for Script MAC

Key

The Card Master Key MK_{SMI} for Secure Messaging Integrity as derived above:

```
04 40 7F 0E 7F CD 4A 02 FD 7F 3B 75 EF 97 3E 52
```

Input

The input is the last AC, in this case the ARQC:

```
14 1D 34 65 C6 85 7C 46
```

Output

The concatenation of:

- The Triple-DES Encryption of X = "the input with the third byte replaced by 'F0' "
- The Triple-DES Encryption of Y = "the input with the third byte replaced by '0F' "

This amounts to:

SK_{SMI} = DES3(MK_{SMI})[X] || DES3(MK_{SMI})[Y]:

```
04 D0 C2 A0 12 07 D8 62 40 3C BA C9 7D 74 C0 2B
```

Note **The result can be submitted to parity forcing, like the Card Master Keys. This is not shown here as this is entirely internal to the ICC, and the computation is not affected.**

A.4.6 - Script MAC for PIN Change

Key

The Session Key SK_{SMI} for Secure Messaging Integrity as derived above:

```
04 D0 C2 A0 12 07 D8 62 40 3C BA C9 7D 74 C0 2B
```

Input

For Secure Messaging Integrity (MAC computation), the input consists of the concatenation of:

- For the first script command in a session, the last Application Cryptogram; for each subsequent script command in the same session the latest full 8-byte MAC of the preceding script command. In this case, it is the ARQC:

```
14 1D 34 65 C6 85 7C 46
```

- The command header (CLA INS P1 P2), with padding. For PIN Change, this is:

```
8C 24 00 02 80 00 00 00
```

- The script data. In this case, the encrypted PIN block (EncData) as computed above embedded in a TLV-encoded data object:

```
87 11 01 DB 8D 1E 79 82 52 56 06 32 70 3D A7 2F
B1 9B EA
```

Note that the 5 padding bytes '80 00 00 00 00' for the above data object are added as the first step of the MAC computation.

The concatenation:

```
14 1D 34 65 C6 85 7C 46 8C 24 00 02 80 00 00 00
87 11 01 DB 8D 1E 79 82 52 56 06 32 70 3D A7 2F
B1 9B EA
```

Output

ScriptMAC = MAC4(SK_{SMI})[Input], where MAC4 denotes the first 4 bytes of the MAC:

```
21 9E 22 CD
```

The complete ICC command is now:

```
8C 24 00 02 19 87 11 01 DB 8D 1E 79 82 52 56 06
32 70 3D A7 2F B1 9B EA 8E 04 21 9E 22 CD
```

A.5 - Static Data Authentication

This section illustrates the cryptographic computations performed by a terminal when performing static data authentication.

Verification of Signed Static Application Data entails the following steps:

- Verification of the Issuer Public Key Certificate, using the Payment System Public Key
- Verification of the Signed Static Application Data, using the Issuer Public Key obtained in the previous step
- Retrieval of the Data Authentication Code

This section illustrates step two of the verification of the Signed Static Application Data, using the Issuer Public Key obtained in the previous step.

A.5.1 Verification of the Signed Static Application Data

Public Verification Key

The Issuer Public Key retrieved from the Issuer Public Key Certificate:

Modulus:

```
A0 DC F4 BD E1 9C 35 46 B4 B6 F0 41 4D 17 4D DE
29 4A AB BB 82 8C 5A 83 4D 73 AA E2 7C 99 B0 B0
53 A9 02 78 00 72 39 B6 45 9F F0 BB CD 7B 4B 9C
6C 50 AC 02 CE 91 36 8D A1 BD 21 AA EA DB C6 53
47 33 7D 89 B6 8F 5C 99 A0 9D 05 BE 02 DD 1F 8C
5B A2 0E 2F 13 FB 2A 27 C4 1D 3F 85 CA D5 CF 66
68 E7 58 51 EC 66 ED BF 98 85 1F D4 E4 2C 44 C1
D5 9F 59 84 70 3B 27 D5 B9 F2 1B 8F A0 D9 32 79
FB BF 69 E0 90 64 29 09 C9 EA 27 F8 98 95 95 41
AA 67 57 F5 F6 24 10 4F 6E 1D 3A 95 32 F2 A6 E5
15 15 AE AD 1B 43 B3 D7 83 50 88 A2 FA FA 7B E7
```

Exponent:

```
03
```

Signature Verification

The signature is verified. The first input for this recovery function consists of the Signed Static Application Data (SSAD, tag '93'):

```
2C 67 B5 70 4E FE 94 FF 95 F3 28 28 A5 5E 1C EC
7C E8 71 FF D1 82 F0 6A 9B 50 86 30 75 EA 7F 01
9D C2 52 5D 60 B6 FD B5 9D 4B 20 76 51 B0 C9 07
F5 B9 7E 65 B0 40 AB CD 92 46 06 CF 27 44 43 C3
6B D2 EF CE 8B 5C E0 2D 0C B9 9D D9 EC 4F F6 9B
45 FD DE 6A 4E 0E DC 3A 5D 08 DD 21 EC BC 08 72
E8 4E C0 DA F2 1C 2A B8 00 14 7F 60 F0 6E B5 09
86 27 A7 F9 E1 B9 60 CA AA 3D DD 61 25 A3 CC 62
F8 F6 1F 8A F4 2D 41 99 3D 0D BC 70 2E 98 CE 81
74 0F 6F 1C BA 33 69 FB ED 1C 9A 9C 82 0A 3F BD
25 C6 58 31 E8 A5 9B C0 B5 D0 E8 11 2D 9E 6C CF
```

The following steps are performed:

1. The signature as well as the Issuer Public Key Modulus consists of 176 bytes (1,408 bits).
2. The exponent is 3, so the Recover() function is:

   $X = (SSAD)^3 \bmod (\text{Issuer Public Key Modulus})$:

```
6A|03|01|00 00|BB BB BB BB BB BB BB BB BB BB BB
BB BB BB BB BB BB BB BB BB BB BB BB BB BB BB BB
BB BB BB BB BB BB BB BB BB BB BB BB BB BB BB BB
BB BB BB BB BB BB BB BB BB BB BB BB BB BB BB BB
BB BB BB BB BB BB BB BB BB BB BB BB BB BB BB BB
BB BB BB BB BB BB BB BB BB BB BB BB BB BB BB BB
BB BB BB BB BB BB BB BB BB BB BB BB BB BB BB BB
BB BB BB BB BB BB BB BB BB BB BB BB BB BB BB BB
BB BB BB BB BB BB BB BB BB BB BB BB BB BB BB BB
BB BB BB BB BB BB BB BB BB BB BB|47 26 7B 16 3C
87 53 5B 60 6C F7 76 A2 28 67 9A 66 73 F9 0B|BC
```

   (Separators '|' are inserted to ease parsing; the separators correspond to the data elements.)

3. The Recovered Data Header is the first byte:

```
6A
```

   which is correct.

4. The Certificate Format is 03:

```
03
```

5. The recovered part of the message (second through fifth data elements; that is all except header byte, hash result, and trailer byte) is:

```
03 01 00 00 BB BB BB BB BB BB BB BB BB BB BB BB
BB BB BB BB BB BB BB BB BB BB BB BB BB BB BB BB
BB BB BB BB BB BB BB BB BB BB BB BB BB BB BB BB
BB BB BB BB BB BB BB BB BB BB BB BB BB BB BB BB
BB BB BB BB BB BB BB BB BB BB BB BB BB BB BB BB
BB BB BB BB BB BB BB BB BB BB BB BB BB BB BB BB
BB BB BB BB BB BB BB BB BB BB BB BB BB BB BB BB
```

```
BB BB BB BB BB BB BB BB BB BB BB BB BB BB BB BB
BB BB BB BB BB BB BB BB BB BB BB BB BB BB BB BB
BB BB BB BB BB BB BB BB BB BB
```

Concatenate to this the Static Data to be authenticated as identified by the AFL, which consists of:

- Application Effective Date:
  ```
  5F 25 03 06 01 01
  ```
- Application Expiration Date:
  ```
  5F 24 03 10 12 31
  ```
- Application Usage Control:
  ```
  9F 07 02 FF 00
  ```
- Application PAN (5413339000006165):
  ```
  5A 08 54 13 33 90 00 00 61 65
  ```
- The PAN Sequence Number (0):
  ```
  5F 34 01 00
  ```
- Issuer Action Code Default:
  ```
  9F 0D 05 F0 40 64 10 00
  ```
- Issuer Action Code Denial:
  ```
  9F 0E 05 00 10 88 00 00
  ```
- Issuer Action Code Online:
  ```
  9F 0F 05 F0 E0 64 98 00
  ```
- CVM List:
  ```
  8E 10 00 00 00 00 00 00 00 00 41 03 42 03 5E 03
  1F 03
  ```
- Issuer Country Code:
  ```
  5F 28 02 09 78
  ```
- SDA Tag List:
  ```
  9F 4A 01 82
  ```
- The value (without tag and length) of the data element indicated in the SDA Tag List (tag '9F4A'), which is the Application Interchange Profile (tag '82'):
  ```
  58 00
  ```

The Static Data to be authenticated equals:

```
5F 25 03 06 01 01 5F 24 03 10 12 31 9F 07 02 FF
00 5A 08 54 13 33 90 00 00 61 65 5F 34 01 00 9F
0D 05 F0 40 64 10 00 9F 0E 05 00 10 88 00 00 9F
0F 05 F0 E0 64 98 00 8E 10 00 00 00 00 00 00 00
00 41 03 42 03 5E 03 1F 03 5F 28 02 09 78 9F 4A
01 82 58 00
```

This yields:

```
03 01 00 00 BB BB BB BB BB BB BB BB BB BB BB BB
BB BB BB BB BB BB BB BB BB BB BB BB BB BB BB BB
BB BB BB BB BB BB BB BB BB BB BB BB BB BB BB BB
BB BB BB BB BB BB BB BB BB BB BB BB BB BB BB BB
BB BB BB BB BB BB BB BB BB BB BB BB BB BB BB BB
BB BB BB BB BB BB BB BB BB BB BB BB BB BB BB BB
BB BB BB BB BB BB BB BB BB BB BB BB BB BB BB BB
BB BB BB BB BB BB BB BB BB BB BB BB BB BB BB BB
BB BB BB BB BB BB BB BB BB BB BB BB BB BB BB BB
BB BB BB BB BB BB BB BB BB BB 5F 25 03 06 01 01
5F 24 03 10 12 31 9F 07 02 FF 00 5A 08 54 13 33
90 00 00 61 65 5F 34 01 00 9F 0D 05 F0 40 64 10
00 9F 0E 05 00 10 88 00 00 9F 0F 05 F0 E0 64 98
```

```
00 8E 10 00 00 00 00 00 00 00 00 41 03 42 03 5E
03 1F 03 5F 28 02 09 78 9F 4A 01 82 58 00
```

6. The Hash Algorithm Indicator (the 2nd byte of the message, '01') indicates SHA-1 as hash function. Thus, apply SHA-1 to the complete message. The result is:

```
47 26 7B 16 3C 87 53 5B 60 6C F7 76 A2 28 67 9A
66 73 F9 0B
```

7. This is identical to the value obtained from X in step 2:

```
47 26 7B 16 3C 87 53 5B 60 6C F7 76 A2 28 67 9A
66 73 F9 0B
```

8. Static Data Authentication has succeeded.

A.6 - Dynamic Data Authentication (DDA)

Generation and verification of Signed Dynamic Application Data entails the following steps:

- Generation of the Signed Dynamic Application Data by the IC Card
- Verification of the Issuer Public Key Certificate, including retrieval of the Issuer Public Key
- Verification of the ICC Public Key Certificate, including retrieval of the ICC Public Key
- Verification of the Signed Dynamic Application Data, including retrieval of the IC Card Dynamic Number
- Retrieval of the IC Card Dynamic Number

This section illustrates all steps listed except the verification of the Issuer Public Key Certificate. Verification of an Issuer Public Key Certificate is similar to the verification of an ICC Public Key Certificate except that it does not include Static Data.

A.6.1 - Generation of the Signed Dynamic Application Data

ICC Private Key

The ICC Private Key, given in "standard" format:

Modulus:

```
A2 BC C5 CE BA C3 19 6B 7E 93 3B 1A 46 66 A6 7D
ED 64 2B A3 CE 89 5C 31 8A 3D FC 12 65 B3 B9 CF
FC 12 FD E5 16 8E 84 19 06 50 E1 74 17 EC 8B 04
A4 E9 85 E0 D4 E1 AF 4B 76 08 4F 44 A1 F2 7F E8
65 E5 FA B2 07 AA 71 52 37 A6 D1 F7 BA CF 1E 2D
DF FB 5A F2 40 00 93 58 4A FA F0 7D FC 73 F8 6C
94 E6 2C 2C FF 44 3D 90 87 EF C1 90 A8 68 24 5C
06 40 0E 06 A3 D2 0B 1C D5 D0 AB 86 CF 88 1B 81
39 DD 50 BD 06 35 12 41 12 A4 93 37 1C 88 9C 53
51 3C D5 CE 3F E3 7B B7 A2 0A AA 97 90 29 08 10
73 F2 31 1A 0C 0E 1D A9 AC 8E B3 45 AB 81 0D 8B
```

Private Exponent d:

```
0A D9 62 85 3F A6 9B 4B 6E D6 9D 8A 48 F5 C6 D5
31 F5 9C 82 63 1A 39 58 A2 D0 EE AB E4 A5 94 EB
BB 78 BB 97 CE 4D C4 8A 33 9E FD F6 AC 42 F8 33
82 75 F7 DB C9 EC E9 8D 90 66 F4 37 C6 87 A2 20
8F 53 99 3F 11 93 E5 6B E1 93 A7 99 0C 74 35 36
42 21 D2 DC F3 33 3D 05 C7 A4 65 30 4C 34 96 96
14 DD 4D C9 7B 2D 8B 70 63 7D A2 C3 DA CD 6F EE
FB 05 13 3B 7F E2 C3 54 FA 75 CF 1C 02 69 A4 73
E2 EB BC CE 4E 9F 4B 1A BF FC 57 75 E6 28 E4 C5
21 94 9C 1D 15 DC AB 95 FF C0 11 64 76 18 C4 6C
06 34 98 31 FD 11 60 8F A5 D8 DD DB 8F 56 0D B3
```

The same ICC Private Key, given in the commonly used "Chinese Remainder Theorem" format:

The prime p:

```
C7 0E 73 F0 30 3C C8 83 21 C3 60 AA 7E 57 13 A7
81 B8 80 72 16 16 AA D4 09 A5 5E 7C D8 77 75 F9
52 2C F6 03 51 8B 29 C8 D1 72 6C 2D 1E 00 20 54
98 AA 6C 77 A5 F6 84 70 5D 72 79 BB B4 A5 08 21
8C ED 4F 3D A5 CF 6F 78 AA EE 1D 0E 79 79 00 73
67 CC 99 02 23 A2 C8 25
```

The exponent used modulo p (this is the private exponent d reduced modulo p–1):

```
84 B4 4D 4A CA D3 30 57 6B D7 95 C6 FE E4 B7 C5
01 25 AA F6 B9 64 71 E2 B1 18 E9 A8 90 4F A3 FB
8C 1D F9 57 8B B2 1B DB 36 4C 48 1E 14 00 15 8D
BB 1C 48 4F C3 F9 AD A0 3E 4C 51 27 CD C3 5A C1
08 9E 34 D3 C3 DF 9F A5 C7 49 68 B4 50 FB 55 A2
45 33 10 AC 17 C1 DA C3
```

The prime q:

```
D1 4A 8E B9 55 22 5D 1E 3A 2B 3C B4 49 41 FE 53
31 DA B7 A4 C0 47 EA 87 47 4E 8F 0D 4D 11 23 28
D6 BC 92 DF 59 CC 4E EE 1C 9A D4 79 4C DF 8B 5B
3A 31 06 D6 FA 2C B0 55 FC 15 36 5E 43 50 65 CC
18 DC 56 76 FE E6 16 43 6B EF 29 1D BE 90 74 CB
8D 0D 1A 66 21 D3 77 EF
```

The exponent used modulo q (this is the private exponent d reduced modulo q–1):

```
8B 87 09 D0 E3 6C 3E 14 26 C7 7D CD 86 2B FE E2
21 3C 7A 6D D5 85 47 04 DA 34 5F 5E 33 60 C2 1B
39 D3 0C 94 E6 88 34 9E BD BC 8D A6 33 3F B2 3C
D1 76 04 8F 51 73 20 39 52 B8 CE E9 82 35 99 32
BB 3D 8E F9 FF 44 0E D7 9D 4A 1B 69 29 B5 A3 32
5E 08 BC 44 16 8C FA 9F
```

The inverse of q modulo p:

```
44 8B EF 26 9A AB 94 68 F9 C2 F5 C2 36 70 11 3E
C0 B7 1A 4D F3 F6 98 BF A4 81 BF 3E C2 C3 6F 1B
41 02 FD 42 B0 EB FD AF FC 29 E4 24 62 35 8B B8
3B A2 E9 73 C2 49 50 37 A7 C3 02 12 05 79 7E 49
9C 15 DF E4 11 91 53 5E D1 7E 23 DF 41 4C 71 C6
60 A5 07 B4 E2 1D 8D E3
```

The inverse of p modulo q:

```
89 38 5E C5 94 CE 7E 58 57 DE C3 A8 98 1D B8 B8
C4 FE 69 3F B4 5A 38 49 47 1B 01 EA 20 B1 DB CB
CD 70 D5 34 51 A4 C4 43 FC E1 AA 71 49 42 91 DA
D2 B8 09 59 A9 B9 63 AC B8 64 BF DF 69 31 15 5B
3C 81 F0 C9 72 2E E1 F6 D0 FB 43 FD F8 44 CC 08
85 34 03 1B 03 32 B5 73
```

Note **Typically, only one of the inverses given here is used.**

ICC Public Key

The ICC Public Key retrieved from the ICC Public Key Certificate:

Modulus:

```
A2 BC C5 CE BA C3 19 6B 7E 93 3B 1A 46 66 A6 7D
ED 64 2B A3 CE 89 5C 31 8A 3D FC 12 65 B3 B9 CF
FC 12 FD E5 16 8E 84 19 06 50 E1 74 17 EC 8B 04
A4 E9 85 E0 D4 E1 AF 4B 76 08 4F 44 A1 F2 7F E8
65 E5 FA B2 07 AA 71 52 37 A6 D1 F7 BA CF 1E 2D
DF FB 5A F2 40 00 93 58 4A FA F0 7D FC 73 F8 6C
94 E6 2C 2C FF 44 3D 90 87 EF C1 90 A8 68 24 5C
06 40 0E 06 A3 D2 0B 1C D5 D0 AB 86 CF 88 1B 81
39 DD 50 BD 06 35 12 41 12 A4 93 37 1C 88 9C 53
51 3C D5 CE 3F E3 7B B7 A2 0A AA 97 90 29 08 10
73 F2 31 1A 0C 0E 1D A9 AC 8E B3 45 AB 81 0D 8B
```

Exponent:

```
03
```

Note **The ICC Public Key is used only for DDA verification.**

Input

The Dynamic Application Data to be signed. It is the concatenation of:

- Signed Data Format:

```
05
```

- Hash Algorithm Indicator (SHA-1):

```
01
```

- ICC Dynamic Data Length:

```
09
```

- ICC Dynamic Data (the randomly generated ICC Dynamic Number IDN, preceded by its length on one byte ‘08’):

```
08 56 D3 96 58 A2 EE D9 B1
```

- Pad Pattern (176–34 = 142 bytes, as the ICC Public Key Modulus is 176 bytes long):

```
BB BB BB BB BB BB BB BB BB BB BB BB BB BB BB BB
BB BB BB BB BB BB BB BB BB BB BB BB BB BB BB BB
BB BB BB BB BB BB BB BB BB BB BB BB BB BB BB BB
BB BB BB BB BB BB BB BB BB BB BB BB BB BB BB BB
BB BB BB BB BB BB BB BB BB BB BB BB BB BB BB BB
BB BB BB BB BB BB BB BB BB BB BB BB BB BB BB BB
BB BB BB BB BB BB BB BB BB BB BB BB BB BB BB BB
BB BB BB BB BB BB BB BB BB BB BB BB BB BB BB BB
BB BB BB BB BB BB BB BB BB BB BB BB BB BB
```

- Terminal Dynamic Data, which is the concatenation of the values of the data elements indicated in the DDOL:

  Unpredictable Number:

```
A0 B1 C2 D3
```

Therefore, the Dynamic Application Data to be Signed (i.e. input to the SHA-1 hash algorithm) equals:

```
05 01 09 08 56 D3 96 58 A2 EE D9 B1 BB BB BB BB
```

```
BB BB BB BB BB BB BB BB BB BB BB BB BB BB BB
BB BB BB BB BB BB BB BB BB BB BB BB BB BB BB
BB BB BB BB BB BB BB BB BB BB BB BB BB BB BB
BB BB BB BB BB BB BB BB BB BB BB BB BB BB BB
BB BB BB BB BB BB BB BB BB BB BB BB BB BB BB
BB BB BB BB BB BB BB BB BB BB BB BB BB BB BB
BB BB BB BB BB BB BB BB BB BB BB BB BB BB BB
BB BB BB BB BB BB BB BB BB BB A0 B1 C2 D3
```

the SHA-1 result of which is:

```
8E 8A 2B F6 0D E6 B9 A3 19 38 FA 8E F4 A4 11 B4
AB 1A 53 18
```

Signature Generation

First, concatenate:

- The header byte:

```
6A
```

- The leftmost 80–22 = 58 bytes of the message (the Dynamic Application Data to be Signed):

```
05 01 09 08 56 D3 96 58 A2 EE D9 B1 BB BB BB BB
BB BB BB BB BB BB BB BB BB BB BB BB BB BB BB BB
BB BB BB BB BB BB BB BB BB BB BB BB BB BB BB BB
BB BB BB BB BB BB BB BB BB BB BB BB BB BB BB BB
BB BB BB BB BB BB BB BB BB BB BB BB BB BB BB BB
BB BB BB BB BB BB BB BB BB BB BB BB BB BB BB BB
BB BB BB BB BB BB BB BB BB BB BB BB BB BB BB BB
BB BB BB BB BB BB BB BB BB BB BB BB BB BB BB BB
BB BB BB BB BB BB BB BB BB BB
```

(we need 176–22 bytes, as the ICC Public Key Modulus is 176 bytes long).

- The result of application of the Hash Algorithm (SHA-1) to the complete message:

```
8E 8A 2B F6 0D E6 B9 A3 19 38 FA 8E F4 A4 11 B4
AB 1A 53 18
```

- And the trailer byte:

```
BC
```

This results in the Input:

```
6A 05 01 09 08 56 D3 96 58 A2 EE D9 B1 BB BB BB
BB BB BB BB BB BB BB BB BB BB BB BB BB BB BB BB
BB BB BB BB BB BB BB BB BB BB BB BB BB BB BB BB
BB BB BB BB BB BB BB BB BB BB BB BB BB BB BB BB
BB BB BB BB BB BB BB BB BB BB BB BB BB BB BB BB
BB BB BB BB BB BB BB BB BB BB BB BB BB BB BB BB
BB BB BB BB BB BB BB BB BB BB BB BB BB BB BB BB
BB BB BB BB BB BB BB BB BB BB BB BB BB BB BB BB
BB BB BB BB BB BB BB BB BB BB BB BB BB BB BB BB
BB BB BB BB BB BB BB BB BB BB 8E 8A 2B F6 0D
```

```
E6 B9 A3 19 38 FA 8E F4 A4 11 B4 AB 1A 53 18 BC
```

Apply the RSA private transformation to this, i.e. raise it to the power 'd' modulo the ICC Public Key Modulus. This can be done using the standard representation or CRT representation. This is outside the scope of this document. The Signed Dynamic Application Data equals:

SDAD = $(\text{Input})^d$ mod (ICC Public Key Modulus):

```
0E 3E 5F 59 62 F0 04 74 85 CE 92 B9 6F 52 C9 8B
39 A1 DC F7 28 AE E8 BA D4 6C E5 2E 91 03 FB A7
2D 63 01 50 5F 2D 66 0F E8 78 5B 68 72 02 92 B6
8C 92 0B 50 8C 8D A9 DA 0B E3 52 01 CD BB C9 FA
DF 65 7D E8 98 89 70 2C 87 44 8D 27 2F 66 2B D6
33 75 E1 5D 4A F1 89 99 0B 81 FB E3 8B 40 44 92
99 BB 1C 2E DD 8C E5 C3 EA 18 BC 03 E6 C8 66 41
71 96 1C 0E 75 47 CB 3B 55 3C 66 81 2A 54 4B AA
43 92 C6 AE 23 16 F2 69 1C 7F 6D D3 B6 8E BC A4
FC 77 AB F8 E7 2B CF 85 69 45 7B 83 4C 52 3D A3
EC 56 2E 7B DD 57 8E 21 88 7F D1 72 DB 7D 12 9B
```

This concludes the ICC computations for Dynamic Data Authentication. The following sections deal with the terminal verification of the SDAD.

A.6.2 - Verification of the ICC Public Key Certificate

Public Verification Key

The Issuer Public Key retrieved from the Issuer Public Key Certificate:

Modulus:

```
A0 DC F4 BD E1 9C 35 46 B4 B6 F0 41 4D 17 4D DE
29 4A AB BB 82 8C 5A 83 4D 73 AA E2 7C 99 B0 B0
53 A9 02 78 00 72 39 B6 45 9F F0 BB CD 7B 4B 9C
6C 50 AC 02 CE 91 36 8D A1 BD 21 AA EA DB C6 53
47 33 7D 89 B6 8F 5C 99 A0 9D 05 BE 02 DD 1F 8C
5B A2 0E 2F 13 FB 2A 27 C4 1D 3F 85 CA D5 CF 66
68 E7 58 51 EC 66 ED BF 98 85 1F D4 E4 2C 44 C1
D5 9F 59 84 70 3B 27 D5 B9 F2 1B 8F A0 D9 32 79
FB BF 69 E0 90 64 29 09 C9 EA 27 F8 98 95 95 41
AA 67 57 F5 F6 24 10 4F 6E 1D 3A 95 32 F2 A6 E5
15 15 AE AD 1B 43 B3 D7 83 50 88 A2 FA FA 7B E7
```

Exponent:

```
03
```

Signature Verification

The first input for this recovery function consists of the ICC Public Key Certificate (ICCCert, tag '9F46'):

```
7D D1 FF 5A 89 27 04 B0 69 25 20 77 17 E0 60 9D
C4 EE DC 9E D4 77 0C EC 8F CE D0 1D 0D AF 25 82
94 C3 42 62 40 DC E9 6B DA F0 85 6B 1B CE 44 42
FE 36 15 DB 0F 86 78 F9 4C 5F 4B 77 88 9D 2C C9
21 F6 F2 CB AD 0D 25 EB 32 3A D8 B1 46 DB F3 C9
EC 92 41 33 94 6D E7 EA 52 DF EF 79 21 87 E3 90
58 D4 D0 E2 8F EC 37 27 B2 04 C8 9D BB 70 4E D8
F5 E3 F7 E5 27 22 80 C9 6A 0E AA DB 51 5E 90 61
81 56 16 85 AB E9 6B 56 22 35 EA 83 29 72 38 74
12 8A 21 64 BD B2 5D B6 D8 34 8B 15 EE E8 98 7E
FC 0A F0 86 EB DD 00 2A A2 02 A0 C8 53 B4 AF 8A
```

The following steps are performed:

1. The signature as well as the Issuer Public Key Modulus consists of 176 bytes (1,408 bits).
2. The exponent is 3, so the Recover() function is:

$X = (ICCCert)^3 \bmod (Issuer\ Public\ Key\ Modulus)$:

```
6A|04|54 13 33 90 00 00 61 73 FF FF|12 10|00 00
01|01|01|B0|01|A2 BC C5 CE BA C3 19 6B 7E 93 3B
1A 46 66 A6 7D ED 64 2B A3 CE 89 5C 31 8A 3D FC
12 65 B3 B9 CF FC 12 FD E5 16 8E 84 19 06 50 E1
74 17 EC 8B 04 A4 E9 85 E0 D4 E1 AF 4B 76 08 4F
44 A1 F2 7F E8 65 E5 FA B2 07 AA 71 52 37 A6 D1
F7 BA CF 1E 2D DF FB 5A F2 40 00 93 58 4A FA F0
7D FC 73 F8 6C 94 E6 2C 2C FF 44 3D 90 87 EF C1
```

```
90 A8 68 24 5C 06 40 0E 06 A3 D2 0B 1C D5 D0 AB
86 CF 88 1B 81 39 DD 50 BD 06 35|07 31 88 9D 56
06 65 89 CD AD 1D A9 45 B3 5C CD BD 81 19 A7|BC
```

(Separators ' | ' are inserted to ease parsing; the separators correspond to the data elements.)

3. The Recovered Data Header is the first byte:

```
6A
```

which is correct..

4. The Certificate Format is '04':

```
04
```

5. The recovered part of the message (second through tenth data elements; that is all except header byte, hash result and trailer byte) is:

```
04 54 13 33 90 00 00 61 73 FF FF 12 10 00 00 01
01 01 B0 01 A2 BC C5 CE BA C3 19 6B 7E 93 3B 1A
46 66 A6 7D ED 64 2B A3 CE 89 5C 31 8A 3D FC 12
65 B3 B9 CF FC 12 FD E5 16 8E 84 19 06 50 E1 74
17 EC 8B 04 A4 E9 85 E0 D4 E1 AF 4B 76 08 4F 44
A1 F2 7F E8 65 E5 FA B2 07 AA 71 52 37 A6 D1 F7
BA CF 1E 2D DF FB 5A F2 40 00 93 58 4A FA F0 7D
FC 73 F8 6C 94 E6 2C 2C FF 44 3D 90 87 EF C1 90
A8 68 24 5C 06 40 0E 06 A3 D2 0B 1C D5 D0 AB 86
CF 88 1B 81 39 DD 50 BD 06 35
```

Concatenate to this:

- The ICC Public Key Remainder:

```
12 41 12 A4 93 37 1C 88 9C 53 51 3C D5 CE 3F E3
7B B7 A2 0A AA 97 90 29 08 10 73 F2 31 1A 0C 0E
1D A9 AC 8E B3 45 AB 81 0D 8B
```

- The ICC Public Key Exponent:

```
03
```

- The Static Data to be Authenticated as identified by the AFL , which consists of:
  - Application Effective Date (TLV coded):

```
5F 25 03 06 01 01
```

  - Application Expiration Date (TLV coded):

```
5F 24 03 10 12 31
```

  - Application Usage Control (TLV coded):

```
9F 07 02 FF 00
```

  - Application PAN (5413339000006173) (TLV coded):

```
5A 08 54 13 33 90 00 00 61 73
```

  - The PAN Sequence Number (0) (TLV coded):

```
5F 34 01 00
```

  - Issuer Action Code Default (TLV coded):

```
9F 0D 05 F8 40 64 10 00
```

- Issuer Action Code Denial (TLV coded):

```
9F 0E 05 00 10 88 00 00
```

- Issuer Action Code Online (TLV coded):

```
9F 0F 05 F8 E0 64 98 00
```

- CVM List:

```
8E 12 00 00 00 00 00 00 00 00 44 03 41 03 42 03
5E 03 1F 03
```

- Issuer Country Code:

```
5F 28 02 02 76
```

- SDA Tag List:

```
9F 4A 01 82
```

- CDOL1 consisting of the following:

  Amount Authorized:

```
9F 02 06
```

  Amount Other:

```
9F 03 06
```

  Terminal Country Code:

```
9F 1A 02
```

  Terminal Verification Result:

```
95 05
```

  Transaction Currency Code:

```
5F 2A 02
```

  Transaction Date:

```
9A 03
```

  Transaction Type:

```
9C 01
```

  Unpredictable Number:

```
9F 37 04
```

  Terminal Type:

```
9F 35 01
```

  Data Authentication Code:

```
9F 45 02
```

  CVM Results:

```
9F 34 03
```

  The complete CDOL1 is then:

```
9F 02 06 9F 03 06 9F 1A 02 95 05 5F 2A 02 9A 03
9C 01 9F 37 04 9F 35 01 9F 45 02 9F 34 03
```

- CDOL2 consisting of the following for CPA:

  Issuer Authentication Data:

```
91 0A
```

  Authorization Response Code:

```
8A 02
```

  Terminal Verification Result:

```
95 05
```

Unpredictable Number:

```
9F 37 04
```

The complete CDOL2 is then:

```
91 0A 8A 02 95 05 9F 37 04
```

– The value (without tag and length) of the data element indicated in the SDA Tag List (tag '9F4A'), which is the Application Interchange Profile (tag '82'):

```
79 00
```

The Static Data to be authenticated equals:

```
5F 25 03 08 01 01 5F 24 03 10 12 31 9F 07 02 FF
00 5A 08 54 13 33 90 00 00 61 73 5F 34 01 00 9F
0D 05 F8 40 64 10 00 9F 0E 05 00 10 88 00 00 9F
0F 05 F8 E0 64 98 00 8E 12 00 00 00 00 00 00 00
00 44 03 41 03 42 03 5E 03 1F 03 5F 28 02 02 76
9F 4A 01 82 9F 02 06 9F 03 06 9F 1A 02 95 05 5F
2A 02 9A 03 9C 01 9F 37 04 9F 35 01 9F 45 02 9F
34 03 91 0A 8A 02 95 05 9F 37 04 79 00
```

This yields:

```
04 54 13 33 90 00 00 61 73 FF FF 12 10 00 00 01
01 01 50 01 A2 BC C5 CE BA C3 19 6B 7E 93 3B 1A
46 66 A6 7D ED 64 2B A3 CE 89 5C 31 8A 3D FC 12
65 B3 B9 CF FC 12 FD E5 16 8E 84 19 06 50 E1 74
17 EC 8B 04 A4 E9 85 E0 D4 E1 AF 4B 76 08 4F 44
A1 F2 7F E8 65 E5 FA B2 07 AA 71 52 37 A6 D1 F7
BA CF 1E 2D DF FB 5A F2 40 00 93 58 4A FA F0 7D
FC 73 F8 6C 94 E6 2C 2C FF 44 3D 90 87 EF C1 90
A8 68 24 5C 06 40 0E 06 A3 D2 0B 1C D5 D0 AB 86
CF 88 1B 81 39 DD 50 BD 06 35 12 41 12 A4 93 37
1C 88 9C 53 51 3C D5 CE 3F E3 7B B7 A2 0A AA 97
90 29 08 10 73 F2 31 1A 0C 0E 1D A9 AC 8E B3 45
AB 81 0D 8B 03 5F 25 03 06 01 01 5F 24 03 10 12
31 9F 07 02 FF 00 5A 08 54 13 33 90 00 00 61 73
5F 34 01 00 9F 0D 05 F8 40 64 10 00 9F 0E 05 00
10 88 00 00 9F 0F 05 F8 E0 64 98 00 8E 12 00 00
00 00 00 00 00 00 44 03 41 03 42 03 5E 03 1F 03
5F 28 02 02 76 9F 4A 01 82 9F 02 06 9F 03 06 9F
1A 02 95 05 5F 2A 02 9A 03 9C 01 9F 37 04 9F 35
01 9F 45 02 9F 34 03 91 0A 8A 02 95 05 9F 37 04
79 00
```

6. The Hash Algorithm Indicator (obtained from X, '01') indicates SHA-1 as hash function. Thus, apply SHA-1 to the complete message. The result is:

```
07 31 88 9D 56 06 65 89 CD AD 1D A9 45 B3 5C CD
BD 81 19 A7
```

7. This is identical to the value obtained from X in step 2:

```
07 31 88 9D 56 06 65 89 CD AD 1D A9 45 B3 5C CD
BD 81 19 A7
```

8. Check that the PAN as obtained from the certificate:

```
54 13 33 90 00 00 61 73 FF FF
```

corresponds to the Application PAN 5413339000006173 read from the ICC.

9. The expiration month is (obtained from X above) '12 10', indicating December 2010.

10. The ICC Public Key Algorithm Indicator (obtained from X above) is '01', which is recognized as "RSA".

11. Concatenate the Leftmost Digits of the ICC Public Key (obtained from X above) and the ICC Public Key Remainder to find the ICC Public Key Modulus:

```
A2 BC C5 CE BA C3 19 6B 7E 93 3B 1A 46 66 A6 7D
ED 64 2B A3 CE 89 5C 31 8A 3D FC 12 65 B3 B9 CF
FC 12 FD E5 16 8E 84 19 06 50 E1 74 17 EC 8B 04
A4 E9 85 E0 D4 E1 AF 4B 76 08 4F 44 A1 F2 7F E8
65 E5 FA B2 07 AA 71 52 37 A6 D1 F7 BA CF 1E 2D
DF FB 5A F2 40 00 93 58 4A FA F0 7D FC 73 F8 6C
94 E6 2C 2C FF 44 3D 90 87 EF C1 90 A8 68 24 5C
06 40 0E 06 A3 D2 0B 1C D5 D0 AB 86 CF 88 1B 81
39 DD 50 BD 06 35 12 41 12 A4 93 37 1C 88 9C 53
51 3C D5 CE 3F E3 7B B7 A2 0A AA 97 90 29 08 10
73 F2 31 1A 0C 0E 1D A9 AC 8E B3 45 AB 81 0D 8B
```

A.6.3 - Verification of the Signed Dynamic Application Data

Public Verification Key

The ICC Public Key retrieved from the ICC Public Key Certificate:

Modulus:

```
A2 BC C5 CE BA C3 19 6B 7E 93 3B 1A 46 66 A6 7D
ED 64 2B A3 CE 89 5C 31 8A 3D FC 12 65 B3 B9 CF
FC 12 FD E5 16 8E 84 19 06 50 E1 74 17 EC 8B 04
A4 E9 85 E0 D4 E1 AF 4B 76 08 4F 44 A1 F2 7F E8
65 E5 FA B2 07 AA 71 52 37 A6 D1 F7 BA CF 1E 2D
DF FB 5A F2 40 00 93 58 4A FA F0 7D FC 73 F8 6C
94 E6 2C 2C FF 44 3D 90 87 EF C1 90 A8 68 24 5C
06 40 0E 06 A3 D2 0B 1C D5 D0 AB 86 CF 88 1B 81
39 DD 50 BD 06 35 12 41 12 A4 93 37 1C 88 9C 53
51 3C D5 CE 3F E3 7B B7 A2 0A AA 97 90 29 08 10
73 F2 31 1A 0C 0E 1D A9 AC 8E B3 45 AB 81 0D 8B
```

Exponent:

```
03
```

Signature Verification

The first input for this recovery function consists of the Signed Dynamic Application Data SDAD as computed above:

```
0E 3E 5F 59 62 F0 04 74 85 CE 92 B9 6F 52 C9 8B
39 A1 DC F7 28 AE E8 BA D4 6C E5 2E 91 03 FB A7
2D 63 01 50 5F 2D 66 0F E8 78 5B 68 72 02 92 B6
8C 92 0B 50 8C 8D A9 DA 0B E3 52 01 CD BB C9 FA
DF 65 7D E8 98 89 70 2C 87 44 8D 27 2F 66 2B D6
33 75 E1 5D 4A F1 89 99 0B 81 FB E3 8B 40 44 92
99 BB 1C 2E DD 8C E5 C3 EA 18 BC 03 E6 C8 66 41
71 96 1C 0E 75 47 CB 3B 55 3C 66 81 2A 54 4B AA
43 92 C6 AE 23 16 F2 69 1C 7F 6D D3 B6 8E BC A4
FC 77 AB F8 E7 2B CF 85 69 45 7B 83 4C 52 3D A3
EC 56 2E 7B DD 57 8E 21 88 7F D1 72 DB 7D 12 9B
```

The following steps are performed:

1. The signature as well as the ICC Public Key Modulus consists of 176 bytes (1,408 bits).
2. The exponent is 3, so the Recover() function is:

$X = (SDAD)^3 \bmod (\text{ICC Public Key Modulus})$:

```
6A|05|01|09|08 56 D3 96 58 A2 EE D9 B1 BB BB BB
BB BB BB BB BB BB BB BB BB BB BB BB BB BB BB BB
BB BB BB BB BB BB BB BB BB BB BB BB BB BB BB BB
BB BB BB BB BB BB BB BB BB BB BB BB BB BB BB BB
BB BB BB BB BB BB BB BB BB BB BB BB BB BB BB BB
BB BB BB BB BB BB BB BB BB BB BB BB BB BB BB BB
BB BB BB BB BB BB BB BB BB BB BB BB BB BB BB BB
BB BB BB BB BB BB BB BB BB BB BB BB BB BB BB BB
BB BB BB BB BB BB BB BB BB BB BB BB BB BB BB BB
BB BB BB BB BB BB BB BB BB BB BB|8E 8A 2B F6 0D
E6 B9 A3 19 38 FA 8E F4 A4 11 B4 AB 1A 53 18|BC
```

(Separators '|' are inserted to ease parsing; the separators correspond to the data elements.)

3. The Recovered Data Header is the first byte:

```
6A
```

which is correct.

4. The Signed Data Format is '05':

```
05
```

5. The recovered part of the message (second through tenth data elements; that is all except header byte, hash result and trailer byte) is:

```
05 01 09 08 56 D3 96 58 A2 EE D9 B1 BB BB BB BB
BB BB BB BB BB BB BB BB BB BB BB BB BB BB BB BB
BB BB BB BB BB BB BB BB BB BB BB BB BB BB BB BB
BB BB BB BB BB BB BB BB BB BB BB BB BB BB BB BB
BB BB BB BB BB BB BB BB BB BB BB BB BB BB BB BB
BB BB BB BB BB BB BB BB BB BB BB BB BB BB BB BB
BB BB BB BB BB BB BB BB BB BB BB BB BB BB BB BB
BB BB BB BB BB BB BB BB BB BB BB BB BB BB BB BB
BB BB BB BB BB BB BB BB BB BB BB BB BB BB BB BB
BB BB BB BB BB BB BB BB BB BB
```

Concatenate to this the Terminal Dynamic Data, which is the concatenation of the values of the data elements indicated in the DDOL, in this case just the:

Unpredictable Number:

```
A0 B1 C2 D3
```

The Dynamic Data to be authenticated equals:

```
05 01 09 08 56 D3 96 58 A2 EE D9 B1 BB BB BB BB
BB BB BB BB BB BB BB BB BB BB BB BB BB BB BB BB
BB BB BB BB BB BB BB BB BB BB BB BB BB BB BB BB
BB BB BB BB BB BB BB BB BB BB BB BB BB BB BB BB
BB BB BB BB BB BB BB BB BB BB BB BB BB BB BB BB
BB BB BB BB BB BB BB BB BB BB BB BB BB BB BB BB
BB BB BB BB BB BB BB BB BB BB BB BB BB BB BB BB
BB BB BB BB BB BB BB BB BB BB BB BB BB BB BB BB
BB BB BB BB BB BB BB BB BB BB A0 B1 C2 D3
```

6. The Hash Algorithm Indicator (obtained from X, '01') indicates SHA-1 as hash function. Thus, apply SHA-1 to the complete message. The result is:

```
8E 8A 2B F6 0D E6 B9 A3 19 38 FA 8E F4 A4 11 B4
AB 1A 53 18
```

7. This is identical to the value obtained from X in step 2:

```
8E 8A 2B F6 0D E6 B9 A3 19 38 FA 8E F4 A4 11 B4
AB 1A 53 18
```

Dynamic Data Authentication has succeeded.

A.7 - Combined DDA / AC Generation (CDA)

This section provides an example of Combined DDA/AC generation, where the Application Cryptogram is a TC. An overview of the command and response is given followed by the steps for constructing the Signed Dynamic Application Data.

GENERATE AC Command

The content of the first GENERATE AC command is follows:

CLA	INS	P1	P2	Lc	Data	Le
80	AE	50	00	2B	0000000002990000000000000560000 0000009780604010011223344220000 00000000000000010002	00

The CDOL1 Related Data is:

- Amount Authorized is '000000000299'
- Amount Other is '000000000000' (no cashback)
- Terminal Country Code is '0056' (Belgium)
- Terminal Verification Results is '0000000000'
- Transaction Currency Code is '0978' (Euro)
- Transaction Date is '060401' (4 April, 2006)
- Transaction Type is '00' (Goods and Services)
- Unpredictable Number is '11223344'
- Terminal Type is '22' (Merchant terminal, offline with online capability)
- CVM result is '010002' (offline plaintext PIN verification successfully performed)

GENERATE AC Response

The content of the GENERATE AC response is follows:

T	L	← Data →												SW1-SW2
		T	L	CID	T	L	ATC	T	L	SDAD	T	L	IAD	
77	81DF	9F27	01	40	9F36	02	0002	9F4B	50	....	9F10	12	....	9000

where the:

CID is a TC:

```
40
```

ATC is:

```
00 02
```

SDAD is:

```
79 ED 26 F1 30 58 3C C2 86 63 FD 68 AD 8D F8 D9
CE 36 78 C5 FB AF BE C4 F7 BF A6 63 58 84 B2 E3
6B 72 38 B3 51 D4 95 D2 ED BD D2 F3 EB 00 29 3B
```

```
A1 56 86 B2 07 F5 BE 84 56 3F 48 79 2B 25 7B 4C
F9 94 48 92 8A C8 03 16 F7 5D 7A 78 21 46 28 2B
BF B7 20 2C 36 6D 2D ED 1E 48 6A B5 AF E7 D7 E8
43 E4 0E A8 5C 51 5D DB 61 16 34 41 17 10 20 5F
EC 02 E0 11 3F A4 13 98 DC 4E 09 EC 82 BD 38 A0
4A 7E 99 23 9C B1 76 94 6E A8 C6 9C C7 C2 1E 75
B6 E3 CC AA 2F 76 E2 E3 57 CE AF E0 3C CB 2F B0
04 CF EC 97 34 67 4D 6D 3A 22 F5 1F B4 9B 6E 4D
```

IAD is:

```
0F A5 01 9A 38 00 00 00 00 00 00 00 00 00 00 00
0F 01 00 00 00 00 00 00 00 00 00 00 00 00 00 00
```

which it is constructed as:

- Length:

```
0F
```

- Common Core Indicator (CCI):

```
A5
```

- Derivation Key Index: (determined by issuer):

```
01
```

- Card Verification Result (CVR):

```
9A 38 00 00 00
```

- Counters:

```
00 00 00 00 00 00 00 00
```

- Length:

```
0F
```

- profile ID:

```
01
```

- Issuer Discretionary:

```
00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
```

SDAD Computation – TC, Encrypted Counters, and Transaction Data Hash Code

In order for the ICC to generate the SDAD, the TC and the Transaction Data Hash Code must first be computed. The following data is needed:

The TC is computed as (see earlier example):

```
39 65 68 89 AB C1 AF FC
```

The Transaction Data Hash Code is computed by applying the SHA1 hash algorithm to input data consisting of all the values identified in CDOL1 and the TLV data contained in the response to the GENERATE AC Command (with the exception of the SDAD). Thus the input is equal to:

```
00 00 00 00 02 99 00 00 00 00 00 00 00 56 00 00
00 00 00 09 78 06 04 01 00 11 22 33 44 22 00 00
00 00 00 00 00 00 00 00 01 00 02|9F 27 01 40|9F
36 02 00 02|9F 10 20 0F A5 01 9A 38 00 00 00 00
00 00 00 00 00 00 00 0F 01 00 00 00 00 00 00 00
00 00 00 00 00 00 00
```

and the Transaction Data Hash Code is equal to SHA1[input] =

```
D2 A4 65 FE 33 2B 10 99 98 AD D8 96 BD BA D8 CB
7E C9 02 60
```

SDAD Computation – Computation of IDN

The ICC randomly generates the ICC Dynamic Number (IDN) as:

```
E7 3A C4 64 CA 63 9D 58
```

SDAD Computation – Computation of SDAD Hash

The Dynamic Application Data to be Signed (i.e. input to the SDAD hash) is the concatenation of:

- Signed Data Format:

```
05
```

- Hash Algorithm Indicator (SHA-1):

```
01
```

- ICC Dynamic Data Length:

```
26
```

- ICC Dynamic Data:

  The ICC Dynamic Number IDN as computed above, preceded by its length on one byte '08'):

```
08 E7 3A C4 64 CA 63 9D 58
```

- CID:

```
40
```

- TC:

```
39 65 68 89 AB C1 AF FC
```

- Transaction Data Hash Code:

```
D2 A4 65 FE 33 2B 10 99 98 AD D8 96 BD BA D8 CB
7E C9 02 60
```

- Pad Pattern (176–63 = 113 bytes, as the example ICC Public Key Modulus is 176 bytes long):

```
BB BB BB BB BB BB BB BB BB BB BB BB BB BB BB BB
BB BB BB BB BB BB BB BB BB BB BB BB BB BB BB BB
BB BB BB BB BB BB BB BB BB BB BB BB BB BB BB BB
BB BB BB BB BB BB BB BB BB BB BB BB BB BB BB BB
BB BB BB BB BB BB BB BB BB BB BB BB BB BB BB BB
BB BB BB BB BB BB BB BB BB BB BB BB BB BB BB BB
BB BB BB BB BB BB BB BB BB BB BB BB BB BB BB BB
BB
```

- Terminal Dynamic Data, which is the concatenation of the values of the data elements indicated in the DDOL:
- Unpredictable Number:

```
11 22 33 44
```

Thus the Dynamic Data to be signed (i.e. input to hash algorithm) is equal to:

```
05 01 26|08 E7 3A C4 64 CA 63 9D 58|40|39 65 68
89 AB C1 AF FC|D2 A4 65 FE 33 2B 10 99 98 AD D8
96 BD BA D8 CB 7E C9 02 60|BB BB BB BB BB BB BB
BB BB BB BB BB BB BB BB BB BB BB BB BB BB BB BB
BB BB BB BB BB BB BB BB BB BB BB BB BB BB BB BB
BB BB BB BB BB BB BB BB BB BB BB BB BB BB BB BB
```

```
BB BB BB BB BB BB BB BB BB BB BB BB BB BB BB BB
BB BB BB BB BB BB BB BB BB BB BB BB BB BB BB BB
BB BB BB BB BB BB BB BB BB BB BB BB BB BB BB BB
BB BB BB BB BB BB BB BB BB BB|11 22 33 44
```

and the SDAD_Hash is equal to SHA1[input] =

```
22 34 7B 69 A1 A0 81 7D 91 39 16 9F 9B 98 46 37
4F B3 40 23
```

SDAD Computation - Signature Generation

All but the Terminal Dynamic Data is recoverable from the signature thus the input to the Sign() function is constructed as follows.

First, concatenate:

- The header byte:

```
6A
```

- The leftmost 176–22 = 154 bytes of the message (the Dynamic Application Data to be Signed):

```
05 01 26 08 E7 3A C4 64 CA 63 9D 58 40 39 65 68
89 AB C1 AF FC D2 A4 65 FE 33 2B 10 99 98 AD D8
96 BD BA D8 CB 7E C9 02 60 BB BB BB BB BB BB BB
BB BB BB BB BB BB BB BB BB BB BB BB BB BB BB BB
BB BB BB BB BB BB BB BB BB BB BB BB BB BB BB BB
BB BB BB BB BB BB BB BB BB BB BB BB BB BB BB BB
BB BB BB BB BB BB BB BB BB BB BB BB BB BB BB BB
BB BB BB BB BB BB BB BB BB BB BB BB BB BB BB BB
BB BB BB BB BB BB BB BB BB BB
```

(we need 176–22 bytes, as the ICC Public Key Modulus is 176 bytes long).

- The result of application of the Hash Algorithm (SHA-1) to the complete message:

```
22 34 7B 69 A1 A0 81 7D 91 39 16 9F 9B 98 46 37
4F B3 40 23
```

- and the trailer byte:

```
BC
```

This results in the input:

```
6A 05 01 26 08 E7 3A C4 64 CA 63 9D 58 40 39 65
68 89 AB C1 AF FC D2 A4 65 FE 33 2B 10 99 98 AD
D8 96 BD BA D8 CB 7E C9 02 60 BB BB BB BB BB BB
BB BB BB BB BB BB BB BB BB BB BB BB BB BB BB BB
BB BB BB BB BB BB BB BB BB BB BB BB BB BB BB BB
BB BB BB BB BB BB BB BB BB BB BB BB BB BB BB BB
BB BB BB BB BB BB BB BB BB BB BB BB BB BB BB BB
BB BB BB BB BB BB BB BB BB BB BB BB BB BB BB BB
BB BB BB BB BB BB BB BB BB BB 22 34 7B 69 A1
A0 81 7D 91 39 16 9F 9B 98 46 37 4F B3 40 23 BC
```

Apply the RSA private transformation Sign() to this, i.e. raise it to the power ‘d’ modulo the ICC Public Key Modulus. This can be done using the standard representation or CRT representation. This is outside the scope

of this manual. Using the ICC Private Key d and ICC Public Key Modulus defined in the previous section, the Signed Dynamic Application Data equals:

SDAD = (Input)^{d} mod (ICC Public Key Modulus):

```
86 43 98 92 29 81 89 2B 5A 34 51 EC 2E CF 41 5F
3E 83 09 C9 5F C0 D6 64 1B 88 18 F2 C2 7E 03 11
EF EF 54 89 E8 AB E7 12 79 B9 E1 78 C9 25 3C 76
29 00 6F 99 C0 E8 3B 0E B7 DE DD 24 7A 1A 4A 99
28 1D 5A D9 A3 34 7F A5 88 80 B2 80 9E 55 B4 35
13 EB 46 96 8C 5D 8C 2D CB DF C4 8D FA 6F F6 8E
6E CB BF 24 40 35 2C B0 C1 2A 33 D2 FA 90 69 90
8D 97 4D 36 74 30 6D F8 05 64 F9 E2 4B 9A EF B9
3A 52 EA 62 4B 40 98 B4 44 71 35 E0 9A E6 57 6A
C1 33 20 3F 0F 35 9F CC BF E3 FC 61 20 DC E4 F8
E5 74 68 72 E7 58 8B B8 2A CC 08 29 6F 55 B9 5F
```

A.8 - Offline PIN Encipherment

Offline PIN Encipherment includes the following steps:

- Verification of the Issuer Public Key Certificate, including retrieval of the Issuer Public Key by the terminal
- Verification of the ICC (PIN Encipherment) Public Key Certificate, including retrieval of the ICC (PIN Encipherment) Public Key by the terminal
- Encryption of the PIN block by the terminal
- Decryption of the PIN block by the ICC

There are two possibilities for the ICC PIN Encipherment Public Key:

1 It is the same key as the ICC Public Key used for Dynamic Data Authentication. In this case, the same ICC Public Key Certificate is used as well.

2 It is a specific key, uniquely used for PIN Encipherment. The ICC PIN Encipherment Public Key Certificate has a slightly different format than the ICC Public Key Certificate. The Static Data to be authenticated is omitted.

For illustrative purposes, the second option is chosen here. Verification of the Issuer Public Key Certificate is not shown.

A.8.1 - Verification of the ICC PIN Encipherment Certificate

This step can be identical to the Verification of the ICC Public Key Certificate as detailed in the section on DDA, in case the same ICC Public Key is used for both PIN Encipherment and Dynamic Data Authentication.

It is assumed that the ICC PIN Encipherment Public Key is a separate key, only used for PIN Encipherment. Therefore a separate ICC PIN Encipherment Public Key Certificate is given. For simplicity, the example uses an ICC PIN Encipherment Public Key with the same value as that used for DDA.

Public Verification Key

The Issuer Public Key retrieved from the Issuer Public Key Certificate:

Modulus:

```
A0 DC F4 BD E1 9C 35 46 B4 B6 F0 41 4D 17 4D DE
29 4A AB BB 82 8C 5A 83 4D 73 AA E2 7C 99 B0 B0
53 A9 02 78 00 72 39 B6 45 9F F0 BB CD 7B 4B 9C
6C 50 AC 02 CE 91 36 8D A1 BD 21 AA EA DB C6 53
47 33 7D 89 B6 8F 5C 99 A0 9D 05 BE 02 DD 1F 8C
5B A2 0E 2F 13 FB 2A 27 C4 1D 3F 85 CA D5 CF 66
68 E7 58 51 EC 66 ED BF 98 85 1F D4 E4 2C 44 C1
D5 9F 59 84 70 3B 27 D5 B9 F2 1B 8F A0 D9 32 79
FB BF 69 E0 90 64 29 09 C9 EA 27 F8 98 95 95 41
AA 67 57 F5 F6 24 10 4F 6E 1D 3A 95 32 F2 A6 E5
15 15 AE AD 1B 43 B3 D7 83 50 88 A2 FA FA 7B E7
```

Exponent:

```
03
```

Signature Verification

The first input for this recovery function consists of the ICC PIN Encipherment Public Key Certificate (PECert, tag '9F2D'):

```
29 31 40 8D 10 41 AB 5B AA 2F D3 56 86 C3 6B E4
B0 40 C9 01 07 E5 AA 25 4D 74 4B 2B FF 40 FF F9
08 0D 07 B6 45 03 A3 45 B9 7A 72 11 50 8D 91 E0
D4 DB 9A 4D EB 71 F7 98 85 12 FD 2B 3A A4 F4 42
41 61 55 4A F1 51 2C 81 F0 B1 80 06 4F D0 8C E8
9E 08 90 78 E3 F0 89 61 1A CE 5D 9B B1 A0 3E 88
77 30 E0 39 46 75 E6 5A 0E 09 00 96 74 6C 87 CB
70 6A DA 9A F7 1A 8C 2D 0E 03 8F 20 30 F5 13 CE
6D 13 78 AB 50 66 A7 5C 6E 61 0A 2A 25 A5 08 5D
88 00 E1 52 F8 09 1D 23 2E 8D 84 34 AD B2 41 8E
C4 8D EB 26 20 99 56 BC D2 B3 35 6E 9C E4 5D 10
```

The description below follows that for DDA but does not involve Static Data to be authenticated:

1. The certificate and the Issuer Public Key Modulus both consist of 176 bytes (1,408 bits).
2. The exponent is 3 so that the Recover() function gives:

   $X = (PECert)^3 \bmod (Issuer\ Public\ Key\ Modulus)$:

   ```
   6A|04|54 13 33 90 00 00 61 73 FF FF|12 10|00 00
   02|01|01|B0|01|A2 BC C5 CE BA C3 19 6B 7E 93 3B
   1A 46 66 A6 7D ED 64 2B A3 CE 89 5C 31 8A 3D FC
   12 65 B3 B9 CF FC 12 FD E5 16 8E 84 19 06 50 E1
   74 17 EC 8B 04 A4 E9 85 E0 D4 E1 AF 4B 76 08 4F
   44 A1 F2 7F E8 65 E5 FA B2 07 AA 71 52 37 A6 D1
   F7 BA CF 1E 2D DF FB 5A F2 40 00 93 58 4A FA F0
   7D FC 73 F8 6C 94 E6 2C 2C FF 44 3D 90 87 EF C1
   90 A8 68 24 5C 06 40 0E 06 A3 D2 0B 1C D5 D0 AB
   86 CF 88 1B 81 39 DD 50 BD 06 35|4C 54 2E 7C 9B
   9C 53 9E 6A BC 72 98 44 14 CB E9 9E EC D4 C6|BC
   ```

   (Separators '|' are inserted to ease parsing.)

3. The Recovered Data Header is the first byte:

   ```
   6A
   ```

   which is correct.

4. The Certificate Format is '04':

   ```
   04
   ```

5. The recovered part of the message (second through tenth data elements; that is all except header byte, hash result and trailer byte) is:

   ```
   04 54 13 33 90 00 00 61 73 FF FF 12 10 00 00 02
   01 01 B0 01 A2 BC C5 CE BA C3 19 6B 7E 93 3B 1A
   46 66 A6 7D ED 64 2B A3 CE 89 5C 31 8A 3D FC 12
   65 B3 B9 CF FC 12 FD E5 16 8E 84 19 06 50 E1 74
   17 EC 8B 04 A4 E9 85 E0 D4 E1 AF 4B 76 08 4F 44
   A1 F2 7F E8 65 E5 FA B2 07 AA 71 52 37 A6 D1 F7
   ```

```
BA CF 1E 2D DF FB 5A F2 40 00 93 58 4A FA F0 7D
FC 73 F8 6C 94 E6 2C 2C FF 44 3D 90 87 EF C1 90
A8 68 24 5C 06 40 0E 06 A3 D2 0B 1C D5 D0 AB 86
CF 88 1B 81 39 DD 50 BD 06 35
```

Concatenate to this:

- The ICC PIN Encipherment Public Key Remainder:

```
12 41 12 A4 93 37 1C 88 9C 53 51 3C D5 CE 3F E3
7B B7 A2 0A AA 97 90 29 08 10 73 F2 31 1A 0C 0E
1D A9 AC 8E B3 45 AB 81 0D 8B
```

- The ICC PIN Encipherment Public Key Exponent:

```
03
```

This yields:

```
04 54 13 33 90 00 00 61 73 FF FF 12 10 00 00 02
01 01 B0 01 A2 BC C5 CE BA C3 19 6B 7E 93 3B 1A
46 66 A6 7D ED 64 2B A3 CE 89 5C 31 8A 3D FC 12
65 B3 B9 CF FC 12 FD E5 16 8E 84 19 06 50 E1 74
17 EC 8B 04 A4 E9 85 E0 D4 E1 AF 4B 76 08 4F 44
A1 F2 7F E8 65 E5 FA B2 07 AA 71 52 37 A6 D1 F7
BA CF 1E 2D DF FB 5A F2 40 00 93 58 4A FA F0 7D
FC 73 F8 6C 94 E6 2C 2C FF 44 3D 90 87 EF C1 90
A8 68 24 5C 06 40 0E 06 A3 D2 0B 1C D5 D0 AB 86
CF 88 1B 81 39 DD 50 BD 06 35 12 41 12 A4 93 37
1C 88 9C 53 51 3C D5 CE 3F E3 7B B7 A2 0A AA 97
90 29 08 10 73 F2 31 1A 0C 0E 1D A9 AC 8E B3 45
AB 81 0D 8B 03
```

6. The Hash Algorithm Indicator (obtained from X, '01') indicates SHA-1 as hash function. Thus, apply SHA-1 to the complete message. The result is:

```
4C 54 2E 7C 9B 9C 53 9E 6A BC 72 98 44 14 CB E9
9E EC D4 C6
```

7. This is identical to the value obtained from X in step 2:

```
4C 54 2E 7C 9B 9C 53 9E 6A BC 72 98 44 14 CB E9
9E EC D4 C6
```

8. Check that the PAN as obtained from the certificate:

```
54 13 33 90 00 00 61 73 FF FF
```

corresponds to the Application PAN 5413339000006173 read from the ICC.

9. The expiration month is (obtained from X above) '12 10', indicating December 2010.

10. The ICC PIN Encipherment Public Key Algorithm Indicator (obtained from X above) is '01', which is recognized as "RSA".

11. Concatenate the leftmost digits of the ICC PIN Encipherment Public Key (obtained from X above) and the ICC PIN Encipherment Public Key Remainder to find the ICC PIN Encipherment Public Key Modulus:

```
A2 BC C5 CE BA C3 19 6B 7E 93 3B 1A 46 66 A6 7D
ED 64 2B A3 CE 89 5C 31 8A 3D FC 12 65 B3 B9 CF
```

```
FC 12 FD E5 16 8E 84 19 06 50 E1 74 17 EC 8B 04
A4 E9 85 E0 D4 E1 AF 4B 76 08 4F 44 A1 F2 7F E8
65 E5 FA B2 07 AA 71 52 37 A6 D1 F7 BA CF 1E 2D
DF FB 5A F2 40 00 93 58 4A FA F0 7D FC 73 F8 6C
94 E6 2C 2C FF 44 3D 90 87 EF C1 90 A8 68 24 5C
06 40 0E 06 A3 D2 0B 1C D5 D0 AB 86 CF 88 1B 81
39 DD 50 BD 06 35 12 41 12 A4 93 37 1C 88 9C 53
51 3C D5 CE 3F E3 7B B7 A2 0A AA 97 90 29 08 10
73 F2 31 1A 0C 0E 1D A9 AC 8E B3 45 AB 81 0D 8B
```

A.8.2 - PIN Encipherment

Public Encryption Key

The ICC PIN Encipherment Public Key retrieved from the ICC PIN Encipherment Public Key Certificate (or ICC Public Key Certificate):

Modulus:

```
A2 BC C5 CE BA C3 19 6B 7E 93 3B 1A 46 66 A6 7D
ED 64 2B A3 CE 89 5C 31 8A 3D FC 12 65 B3 B9 CF
FC 12 FD E5 16 8E 84 19 06 50 E1 74 17 EC 8B 04
A4 E9 85 E0 D4 E1 AF 4B 76 08 4F 44 A1 F2 7F E8
65 E5 FA B2 07 AA 71 52 37 A6 D1 F7 BA CF 1E 2D
DF FB 5A F2 40 00 93 58 4A FA F0 7D FC 73 F8 6C
94 E6 2C 2C FF 44 3D 90 87 EF C1 90 A8 68 24 5C
06 40 0E 06 A3 D2 0B 1C D5 D0 AB 86 CF 88 1B 81
39 DD 50 BD 06 35 12 41 12 A4 93 37 1C 88 9C 53
51 3C D5 CE 3F E3 7B B7 A2 0A AA 97 90 29 08 10
73 F2 31 1A 0C 0E 1D A9 AC 8E B3 45 AB 81 0D 8B
```

Exponent:

```
03
```

PIN Encryption

The input to PIN Encryption consists of:

A Data Header:

```
7F
```

A PIN block. For PIN 12345 this is:

```
25 12 34 5F FF FF FF FF
```

The ICC Unpredictable Number (obtained via a GET CHALLENGE command) is required:

```
1A 2B 3C 4D 5E 6F 70 81
```

A terminal generated random pad pattern of length 176–17 = 159 bytes, as the ICC PIN Encipherment Public Key Modulus is 176 bytes long:

```
FC 02 7D 70 F1 28 4A 48 41 08 48 28 80 20 B8 25
F5 3D F1 54 B7 32 9C B0 24 3A 6E 5A D2 16 F2 E1
02 B5 1D 41 2F 2D 44 43 6C 18 04 7B BC AD CF C1
96 D9 A8 BC AC 80 9F 1E BF 5C 3F 1A 1D DC E4 6D
37 07 F8 4D F6 76 BE A4 0A FD CE 50 E9 9A 6B 03
```

```
F7 7B 32 28 48 3B 25 1A A0 61 54 A6 2A A1 77 03
6B 27 80 AC 95 BC 05 B2 D8 7B A9 81 07 76 93 A9
30 9B D0 06 47 44 8A 77 94 F2 E5 96 39 D6 6C 06
0E B2 3C F8 96 41 F9 40 43 24 87 CD EA 18 B7 10
34 75 75 C4 28 00 8B 52 BF 9F 61 FF C9 F2 49
```

The concatenation X of these:

```
7F 25 12 34 5F FF FF FF FF 1A 2B 3C 4D 5E 6F 70
81 FC 02 7D 70 F1 28 4A 48 41 08 48 28 80 20 B8
25 F5 3D F1 54 B7 32 9C B0 24 3A 6E 5A D2 16 F2
E1 02 B5 1D 41 2F 2D 44 43 6C 18 04 7B BC AD CF
C1 96 D9 A8 BC AC 80 9F 1E BF 5C 3F 1A 1D DC E4
6D 37 07 F8 4D F6 76 BE A4 0A FD CE 50 E9 9A 6B
03 F7 7B 32 28 48 3B 25 1A A0 61 54 A6 2A A1 77
03 6B 27 80 AC 95 BC 05 B2 D8 7B A9 81 07 76 93
A9 30 9B D0 06 47 44 8A 77 94 F2 E5 96 39 D6 6C
06 0E B2 3C F8 96 41 F9 40 43 24 87 CD EA 18 B7
10 34 75 75 C4 28 00 8B 52 BF 9F 61 FF C9 F2 49
```

is input to the RSA Recover() function:

EncPINData = X^3 mod (ICC PIN Encipherment Public Key Modulus):

```
39 97 F6 15 E6 05 20 41 D4 C7 00 96 3B 78 A4 B3
B2 68 53 29 7F A9 D4 93 81 38 86 A9 42 76 12 2C
13 41 DB 19 95 76 32 C7 AA 0A B2 8B D5 BF 89 F7
C9 80 65 BD 7C E0 30 7A 64 1F 7D D5 13 FC 52 ED
1D FB 9D 82 0A B4 F4 F2 43 B9 D5 AF 26 24 61 16
1D 8A 1B 79 CA B8 AC 99 1A F9 09 D8 28 DA 6A CC
A2 40 C3 2A 38 B0 E6 35 83 35 1F 5C A5 0F 39 27
BE 6A ED 9E A0 D5 3F 2C 4A F2 D9 A6 38 B8 DD 28
AF A8 02 7F BD 36 63 26 56 87 81 6A DD 82 C7 81
EF 81 CF B1 2B 30 35 7D FD 31 E5 A0 6C A3 7C FA
29 E2 75 74 AB E2 DC ED 46 9B 87 F0 53 3F 8B B8
```

A.8.3- PIN Decipherment

ICC Private Key

The ICC PIN Encipherment Private Key, given in "standard" format:

Modulus:

```
A2 BC C5 CE BA C3 19 6B 7E 93 3B 1A 46 66 A6 7D
ED 64 2B A3 CE 89 5C 31 8A 3D FC 12 65 B3 B9 CF
FC 12 FD E5 16 8E 84 19 06 50 E1 74 17 EC 8B 04
A4 E9 85 E0 D4 E1 AF 4B 76 08 4F 44 A1 F2 7F E8
65 E5 FA B2 07 AA 71 52 37 A6 D1 F7 BA CF 1E 2D
DF FB 5A F2 40 00 93 58 4A FA F0 7D FC 73 F8 6C
94 E6 2C 2C FF 44 3D 90 87 EF C1 90 A8 68 24 5C
06 40 0E 06 A3 D2 0B 1C D5 D0 AB 86 CF 88 1B 81
39 DD 50 BD 06 35 12 41 12 A4 93 37 1C 88 9C 53
51 3C D5 CE 3F E3 7B B7 A2 0A AA 97 90 29 08 10
```

```
73 F2 31 1A 0C 0E 1D A9 AC 8E B3 45 AB 81 0D 8B
```

Private Exponent d:

```
0A D9 62 85 3F A6 9B 4B 6E D6 9D 8A 48 F5 C6 D5
31 F5 9C 82 63 1A 39 58 A2 D0 EE AB E4 A5 94 EB
BB 78 BB 97 CE 4D C4 8A 33 9E FD F6 AC 42 F8 33
82 75 F7 DB C9 EC E9 8D 90 66 F4 37 C6 87 A2 20
8F 53 99 3F 11 93 E5 6B E1 93 A7 99 0C 74 35 36
42 21 D2 DC F3 33 3D 05 C7 A4 65 30 4C 34 96 96
14 DD 4D C9 7B 2D 8B 70 63 7D A2 C3 DA CD 6F EE
FB 05 13 3B 7F E2 C3 54 FA 75 CF 1C 02 69 A4 73
E2 EB BC CE 4E 9F 4B 1A BF FC 57 75 E6 28 E4 C5
21 94 9C 1D 15 DC AB 95 FF C0 11 64 76 18 C4 6C
06 34 98 31 FD 11 60 8F A5 D8 DD DB 8F 56 0D B3
```

PIN Decryption

The Encipher PIN Data, as computed above:

```
39 97 F6 15 E6 05 20 41 D4 C7 00 96 3B 78 A4 B3
B2 68 53 29 7F A9 D4 93 81 38 86 A9 42 76 12 2C
13 41 DB 19 95 76 32 C7 AA 0A B2 8B D5 BF 89 F7
C9 80 65 BD 7C E0 30 7A 64 1F 7D D5 13 FC 52 ED
1D FB 9D 82 0A B4 F4 F2 43 B9 D5 AF 26 24 61 16
1D 8A 1B 79 CA B8 AC 99 1A F9 09 D8 28 DA 6A CC
A2 40 C3 2A 38 B0 E6 35 83 35 1F 5C A5 0F 39 27
BE 6A ED 9E A0 D5 3F 2C 4A F2 D9 A6 38 B8 DD 28
AF A8 02 7F BD 36 63 26 56 87 81 6A DD 82 C7 81
EF 81 CF B1 2B 30 35 7D FD 31 E5 A0 6C A3 7C FA
29 E2 75 74 AB E2 DC ED 46 9B 87 F0 53 3F 8B B8
```

is submitted to the RSA Sign() function:

$X = (\text{EncPINData})^d \bmod (\text{ICC PIN Encipherment Public Key Modulus})$.

This results in:

```
7F|25 12 34 5F FF FF FF FF|1A 2B 3C 4D 5E 6F 70
81|FC 02 7D 70 F1 28 4A 48 41 08 48 28 80 20 B8
25 F5 3D F1 54 B7 32 9C B0 24 3A 6E 5A D2 16 F2
E1 02 B5 1D 41 2F 2D 44 43 6C 18 04 7B BC AD CF
C1 96 D9 A8 BC AC 80 9F 1E BF 5C 3F 1A 1D DC E4
6D 37 07 F8 4D F6 76 BE A4 0A FD CE 50 E9 9A 6B
03 F7 7B 32 28 48 3B 25 1A A0 61 54 A6 2A A1 77
03 6B 27 80 AC 95 BC 05 B2 D8 7B A9 81 07 76 93
A9 30 9B D0 06 47 44 8A 77 94 F2 E5 96 39 D6 6C
06 0E B2 3C F8 96 41 F9 40 43 24 87 CD EA 18 B7
10 34 75 75 C4 28 00 8B 52 BF 9F 61 FF C9 F2 49
```

(Separators ‘|’ are inserted to ease parsing).

The recovered data header is the first field:

```
7F
```

The ICC Unpredictable Number is the second field (checked before the header):

```
1A 2B 3C 4D 5E 6F 70 81
```

and is indeed equal to the value provided in answer to the GET CHALLENGE above.

The PIN is retrieved from the PIN block, which is the third field:

```
25 12 34 5F FF FF FF FF
```

which encodes a PIN:

```
12345
```

This PIN is then compared by the ICC to the Reference PIN value, stored in the ICC.

«««§ END OF DOCUMENT §»»»