Comercia 2.0

Security Overview v0.9

Table of contents

List of figures

List of tables

References

[POS_PROTOCOL]	“POI-SWITCH Protocol” Version 0.13 06/02/2018
[POS_CRYPTOGRAHY]	“C20_POI_CRYPTOGRAPHY_V101”
[REDSYS_KEYS]	RS.RI.SEG.PRO.0001_GUÍA PARA LA GESTIÓN DE CLAVES CRIPTOGRÁFICAS CON ENTIDADES 2.1.pdf
[REDSYS_KEYS_H2H]	“Cifrado de PIN (host to host) y cifrado de pista, PAN y CVV2 en PUC (host to host) v2.5”
[ANSI X9.24-1:2009]	ANSI: Retail Financial Services Symmetric Key Management Part 1: Using Symmetric Techniques.
[ANSI X9.24-3:2017]	ANSI: Retail Financial Services Symmetric Key Management - Part 3: Derived Unique Key Per Transaction
[PCI-P2PE]	Payment Card Industry (PCI) Point-to-Point Encryption Version 2.0 June 2015

List of acronyms

POS	Point Of Sales
CGP	Comercia Global Payments
KCV	Key Check Value
KSN	Key Serial Number
PAN	Primary Account Number
PIN	Personal Identification Number
RE	Runtime Environment
POI	Point of Interaction
ZCMK	Zone Control Master Key

Document history

Version	Date	Description	Author
V0.1	02/06/17	• Initial draft	J.Molina
V0.2	13/06/17	• Modified architecture scheme • Key type and length for operational keys and BDK's • Added annexes DUKPT and P2PE	J.Molina
V0.3	21/06/17	• TLS with client certificate for POS/POI • Type and length of keys • POS/POI data elements (KSN, Key Id and KCV) • CryptoMiddleware authentication with HTTPS	J.Molina
V0.4	18/08/17	• Modified architecture scheme • PIN-Pad starter usage • BDK diversification and versioning • New concept for non-operational keys (K_{AUTH} and K_{EK}) • Introduction of random data in DUKPT scheme • Removed usage of Key Vault • Added Redsys key requirements	J.Molina
V0.5	20/09/17	• Removed HMAC • Removed random data • Introduction of K_{SIG} and BDK_{SIG}	J.Molina
V0.6	21/12/17	• Detail keys initially injected • Figure "Cryptographic flow for PAN and PIN" updated • Reference section updated • Clarification about KSIG usage included in 2.1.1 • Conciliation file generation mentioned as case of PAN/CARD data use in 2.3.7 • KSN structure modified in 3.2.3 • KSN will be different per key. KSN description modified in: ○ 3.1.2 ○ 3.2.3 • KSN online loading is mentioned in: ○ 3.2.1 ○ 3.2.3 • KPANbatch_n added in 3.1.3 as the key to be used for encrypting card data in the conciliation file. • Atalla AT1000 as the HSM model to be used in 3.2.1 • SCA3 as the tool for LMK & ZMK management in 3.2.2 • Table 3 updated to include different KSNs for operational keys and derivation data. • Table 4 listing CryptoMiddleware updated with: Key validation, Authentication, Key generation, BDKs generation • Table 6 updated with BDKsig usage details • Atalla AT1000 mentioned in 4.1 • ZMK as a second level key in 3.2.2.	J.Molina J.F.Zapata

		• Different KSN per operative key mentioned in 3.2.3 instead of same KSN root with different counter • KSN per key type included in 3.3.2 • KSN replaced per derivation data in 3.3.3 • Several KSNs mentioned in 2.2.3	
V0.7	22/03/18	• Tokenizer architecture changed so it is seen as application • Master keys renamed in table 2, In general original BDK_{AUT}, BDK_{EK}, BDK_{SIG} are replaced by MK_{AUT}, MK_{KEK}, MK_{SIG} along the document • KSNs description and Ids for non-operational keys updated in table 3 • Derivation data length set to 24 bytes in table 3 • MKs/BDKs generation as part of the functionality detailed in table 4 • Figure 5 updated to use MK instead of BDK to name non-operational master keys • [REDSYS_KEYS_H2H] reference document added • [POS_CRYPTOGRAPHY] reference document changed to C20_POI_CRYPTOGRAPHY_V100 • [POS_PROTOCOL] reference updated • Table 6 updated with new MKAUT, MKKEK, MKSIG • Figure 6 updated to include key management details • Table 7 updated with Segment Id structure • Detail of initial double use of KAUTH included in Table 2	J.Molina J.F.Zapata
V0.8	29/03/18	• Modification of figures • Correction of key lengths	J.Molina
V0.9	21/05/18	• Clarification on exit-point tasks in 1. Overview and goals. MAC won't come from the Core, it is generated at the exit-point. • Figure 2 modified to update MAC key names. • Table 1 changed to set Kauth as the key used for MAC and to include key lengths. • Point 2.Transactional security details the current ANSI DUKPT standard adopted by POS as well as the AES option for the future. • 2.2.1 Initial authentication removed as Kauth is used for MAC. Specific mechanism for authentication. • Kauth is the key used for authentication in 2.2.1 authentication. • 2.2.2 Operational keys rewritten to explain new role of Kauth. • Kauth removed from 2.2.3 Non-operational keys • Kek size updated in table 2. • MAC DUKPT related data removed from table 3. • Derivation data size changed to 32 bytes in table 3 • MAC generation description changed in table 4 • 2.3.2 Operational keys modified to include new role of KAUTH • 2.3.3 Non-operational keys modified to remove KAUTH • Figure 4 name changed to specify DUKPT operationl keys • 3.1.1 Operationl keys updated to distinguish between DUPKT and static derivation operational keys • 3.1.2 Non-operational keys modified to exclude KAUTH • BDKMAC removed from table 6	J.F.Zapata

		• Key lengths specified in table 6, setting MKKek as TDES 192 bits • BDKMac removed from 3.2.2 Master keys transport • Figure 6 updated to remove BDKMAC and use new non-DUPKT master keys acronym: MK • 3.3.2 title changed to "Loading/updating DUPKT operational keys (IPEK, IPAN)" • Authentication mechanism changed in 3.3.2 to specify that KAUTH is used at frame level • 3.3.3 title changed to "3.3.3. Loading of non-operational keys (KEK) and static operational keys (KAuth)" • Authentication mechanism changed in 3.3.3 to specify that KAUTH is used at frame level	

1. Overview and goals

Comercia 2.0 is an extensive program with several pieces aiming to provide to CGP (Comercia Global Payments) control and information about all the transactions processed in a timely manner, offering extended capabilities to its customers (i.e. merchants) and generating added value applications at the Point Of Sales (POS). Notice that we will be referring to POS and Point of Interaction (POI) indistinctly.

This program is very disruptive as it bases its operational model in the Cloud. This has several benefits, but from a security perspective, it opens several issues which would be less severe working with on premises systems. The solution must be certified by the payment industry standards, known as PCI-DSS and PCI-PIN, also targeting a further level of security known as P2PE.

There are three big pieces to take into account:

- ***POS/POI***

The POS/POI handles the extraction of the card information needed to complete the transaction in a secure way and transfers it to the transaction processor entity. It includes payment and added value applications. The POS/POI PIN-Pad must be PCI-PTS certified.

- ***Transactional Switch***

This is on the server side. It can be splitted into three main components, the Entry-Point, the Core and the Exit-Point. Entry/Exit-Points are in charge of the verification of the messages and the reception and delivery from the POS/POI or the transaction processors. The Core takes care of business operations as data extraction or applications of rules.

- ***Merchant Portal***

This component provides the merchants capabilities to contract products and perform queries about their operations It is fed by the data extracted from the transactional processes, basically from the Switch (and other legacy sources). It also interacts with an ERP/CRM, but this is not in the scope of this document.

This document will be mainly focused on the POS/POI and Transactional Switch. The interaction with the Merchant Portal and ERP/CRM will be the data exportation model, which is covered as part of the Switch security.

The image below (Figure 1 - Highlevel architecture) provides a high level view about the components in the system. Let’s briefly describe how a transaction process would pass through the system.

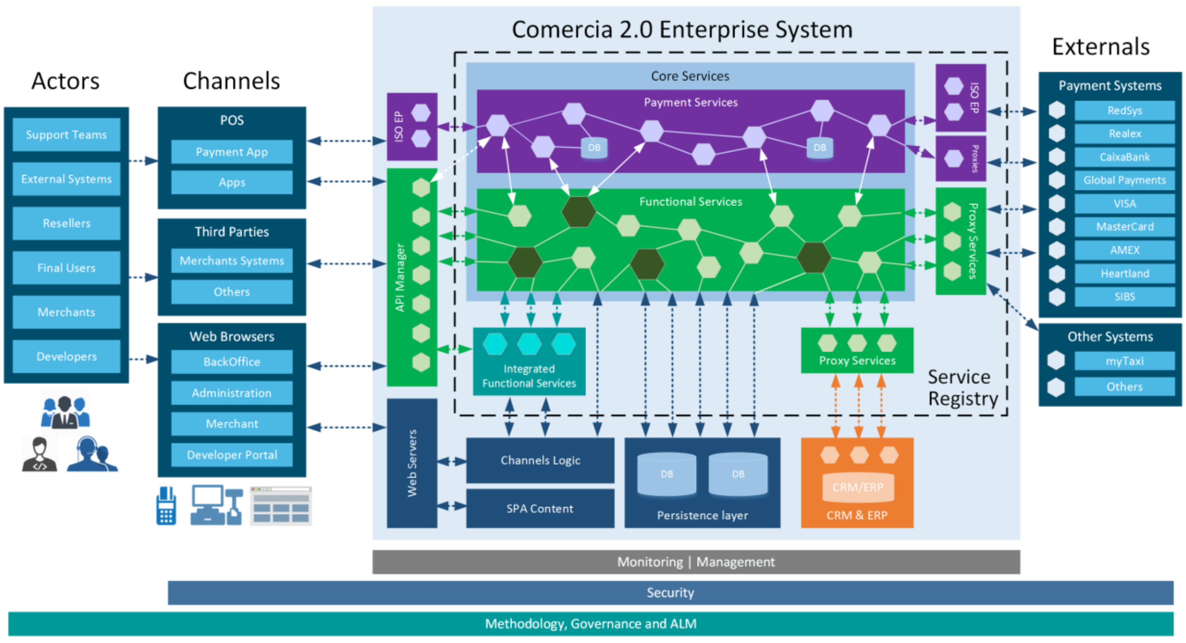


Figure 1 - Highlevel architecture

From a merchant location a purchaser initiates a payment transaction by introducing its card information through a POS/POI (excluding e-commerce here) where the card track data, which includes the PAN, is transmitted to the transaction acquirer. For some transactions, a verification of the user’s Personal Indentification Number (PIN) will be also required and therefore transmitted to the transaction acquirer. These two elements, the card track data (including the PAN) and the PIN, are the basic elements to be handled in a secure way.

As per PCI-DSS, the PAN can be only handled in clear text within the systems complying with PCI-DSS certification and in no case can be persisted in clear. In addition, PAN as part of card track data is subjected also to a highly exigent PCI certification known as P2PE. As per PCI-PIN, the PIN cannot be handled in clear text, but can be translated within a very secure and controlled environment (i.e. HSM compliant with FIPS 140-2 level 3).

Track data, and PIN if required, is transmitted from the POS/POI to the Switch Entry-Point using an adaptation of the ISO8583 protocol, which includes a MAC field, over a secure TLS v1.2 (or higher) connection. The Entry-Point verifies the integrity and authenticity of the message with its MAC. Then transfers the message via HTTPS to the Core, formatting it as a JSON object.

The Core then checks where has to send this transaction query and translates the track data, and PIN if required, with the corresponding keys. It also generates a token of the PAN in order to export it to its data base, which is to be consumed by Merchant Portal (among other services).

The Exit-Point receives the track data (and the PIN if required) already ciphered with the corresponding keys and formats the message according to the receiver processor.

2. Transactional security

The transactional data to be protected when performing a transaction are the track data, which includes the PAN, and the PIN, so see below a diagram showing how these two components are handled in Figure 2 - Cryptographic flow for PAN and PIN. Note that we refer to PAN for simplification, but the actual concept is track data.

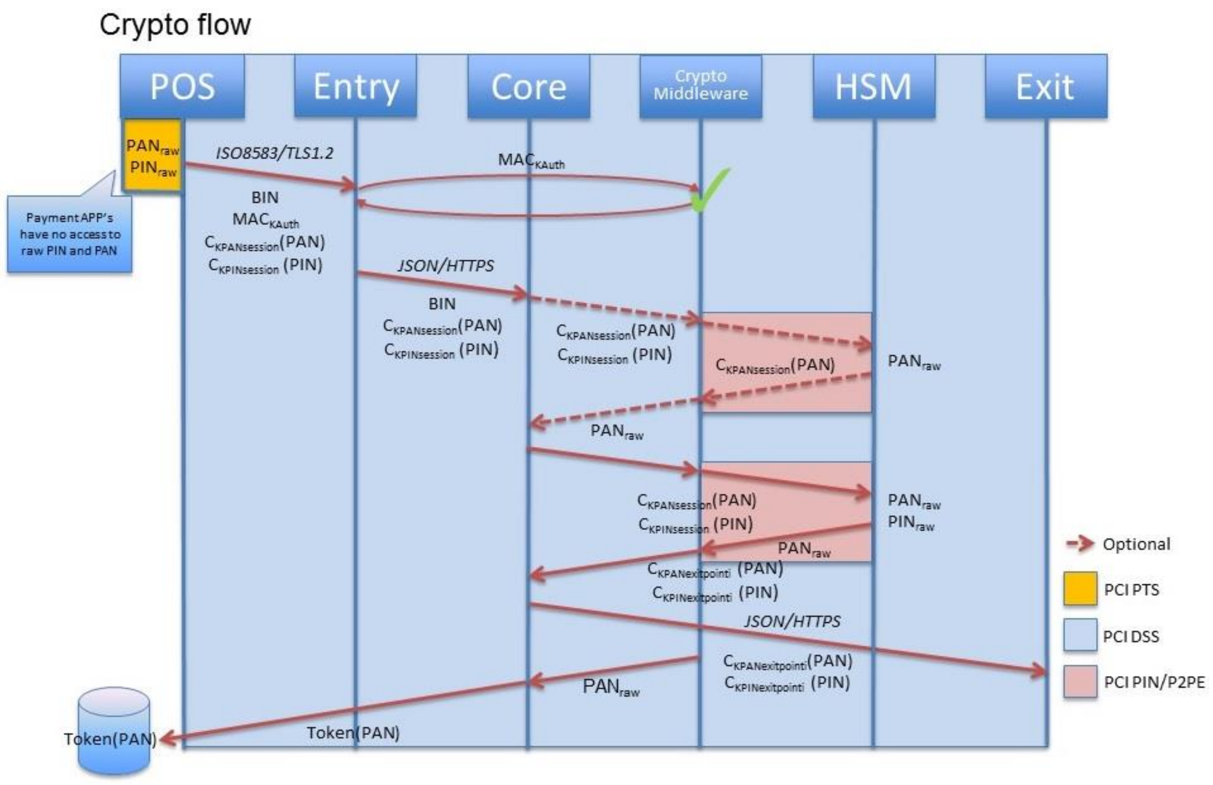


Figure 2 - Cryptographic flow for PAN and PIN

See in the table below information about the keys taking part on the process. It is detailed who is the owner or responsible of these keys management and also from which master key has been derived as part of the DUKPT (Derived Unique Key Per Transaction) process that will be described later on (see DUKPT).

Key	Owner	Description	Key Type	Master Key
$KPIN_{session}$	CGP	Key used for PIN encryption between POS/POI and Core. This key is unique per terminal and transaction. PIN ciphering uses PIN ISO-0.	TDES 128 bits	BDK_{PIN}
$KPAN_{session}$	CGP	Key used for track data encryption between POS/POI and Core. This key is unique per terminal and transaction.	TDES 128 bits	BDK_{PAN}

K_{Auth}	CGP	Key used for MAC verification between POS/POI and Entry-Point or POS/POI and API Manager by means of credentials provided by the Core. This key is unique per terminal and transaction by including random data in the frame. MAC algorithm is ANSI X9.19.	TDES 128 bits	MK_{Auth}
$KPIN_{exitpoint\_n}$	Processor_n / CGP	Key used for PIN encryption between Core and Processor_i. The management of this key is defined by Processor_i.	Processor dependant	Processor dependant
$KPAN_{exitpoint\_n}$	Processor_n / CGP	Key used for track data encryption between Core and Processor_i. The management of this key is defined by Processor_i.	Processor dependant	Processor dependant
$KMAC_{exitpoint\_n}$	Processor_n / CGP	Key used for MAC verification between Exit-Point and Processor_i. The management of this key is defined by Processor_i.	Processor dependant	Processor dependant

Table 1 - Operational keys

The Key Type for $KPIN_{session}$, $KPAN_{session}$ are stated as TDES double length keys as current specification of DUKPT [ANSI X9.24-1:2009] does not admit AES. In the future the plan is to migrate to AES keys as defined in [ANSI X9.24-3:2017] for DUKPT.

2.1. Applications

We must distinguish between the secure payment functionality (native Payment APP) offered by the platform, known as Runtime Environment (RE), and the Payment APP's. The Payment APP's are those which can trigger the payment functionality (e.g. sales, refund, etc.)

Additionally, there are other added value applications, which have no access to the payment APIs offered by the Runtime Environment (RE) provided by the POS/POI, but can trigger the Payment APP's. It is important to emphasize that added value applications have no access to neither track data nor PIN.

2.1.1.Certifications

In terms of P2PE certification, the native Payment APP is to be assessed with special requirements under the Domain 2. The Payment APP's and added value applications are evaluated as per Domain 1, as they don't have access to card track data (see P2PE).

2.1.1.Verification signature

In order to enforce that only authorized applications run on the platform, all applications shall be signed when distributed and verified before executed. Preferably the validation mechanism provided by the manufacturer will be used for simplicity, but if this were not present or could be considered that doesn't offer the required level of security, a proprietary method will be used instead. Based on a BDK_{SIG} a K_{SIG} will be derived for every device and all APP's manifest must be signed with K_{SIG} in order to be loaded an executed in the POS/POI.

2.1.2.Native Payment APP

A Payment APP can initiate a transaction triggering the native Payment APP. The following scheme (Figure 3 – Native Payment APP operation scheme) conceptualizes how the sensitive data is originated and protected and how interacts with a Payment APP.

The Payment APP exposes services in a secure way to the Payment APP's, which in turn exposes a way to trigger payment to the added value applications.

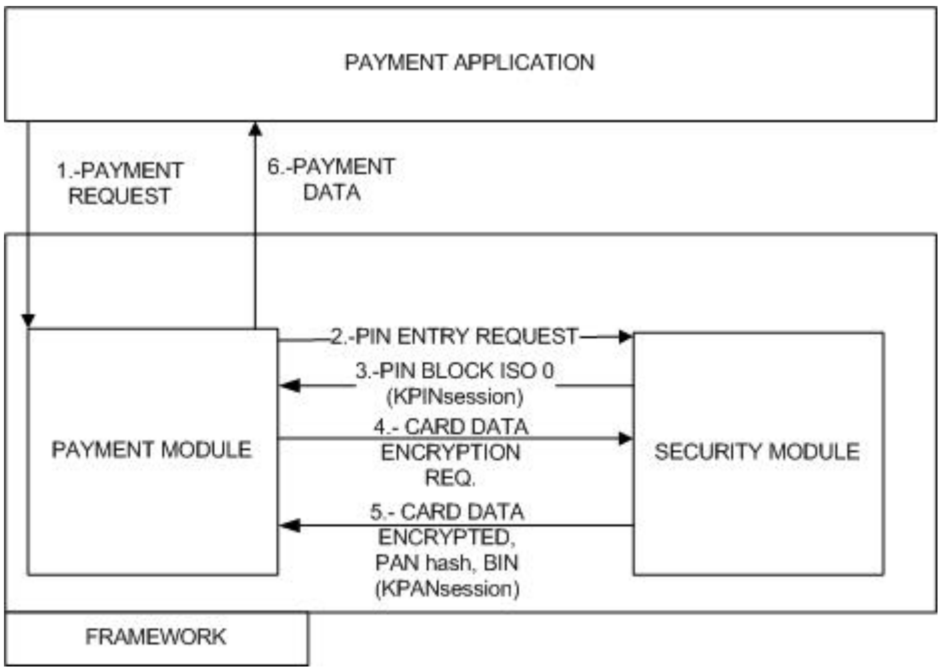

PAYMENT APPLICATION
1.-PAYMENT REQUEST
6.-PAYMENT DATA
2.-PIN ENTRY REQUEST
3.-PIN BLOCK ISO 0 (KPINsession)
4.- CARD DATA ENCRYPTION REQ.
5.- CARD DATA ENCRYPTED, PAN hash, BIN (KPANsession)
PAYMENT MODULE
SECURITY MODULE
FRAMEWORK

Figure 3 – Native Payment APP operation scheme

The description of the process is as follows:

1) A Payment APP triggers a financial transaction through the Payment Module (native Payment APP) in the POS/POI.
2) If required, depending on the type of operation, the Payment Module asks for the PIN to the user through the Security Module.
3) The Security Module generates a frame including the PIN formatted according to PIN ISO-0 and ciphered with $KPIN_{session}$.
4) The Payment Module requests to the Security Module to cipher the track data (in whatever format it is) and also to retrieve the BIN in clear text.
5) The Security Module returns the track data frame ciphered with the $KPAN_{session}$ key and also the BIN in clear text.
6) The Payment APP has triggered the generation of all the necessary data to generate the ISO8583 adapted frame with the MAC calculated with the K_{Auth} key.

2.2. POS/POI

Comercia2.0 POS/POI provides a RE platform, so several applications will be able to run over it. It contains all the necessary functionality to be provided to the Payment APP's (and others). The Payment APP's rely on it for the cryptographic and communications related tasks.

2.2.1.Authentication

The POS/POI gets authenticated due to the inclusion of a MAC generated with K_{Auth}, not only when handling with payment operations but also when using other services (e.g. API Manager authentication). Additionally a specific mechanism for authentication, independent of the financial activity will be included in the POS .

2.2.2.Operational keys

The POS/POI Security Module is responsible of managing the different keys taking part of a transaction process (i.e. $KPIN_{session}$, $KPAN_{session}$, K_{Auth}):

- For PIN and PAN encryption It does so based on a DUKPT model (see DUKPT). This scheme assures that a new key is used for every new transaction by using a counter as diversification element for key generation. In the unlikely situation where the counter reaches the maximum it starts again setting the counter to 1.
  The POS/POI shall inform about Key Serial Numbers (KSNs), the version of the Base Derivation Key (BDK) in use and the Key Check Values (KCV) of the initial keys in use (IPEK, IPAN, K_{AUTH} and K_{EK}) to verify proper key sets are being generated. The size of the KSN is 10 bytes, the size of the key identifiers is 2 bytes and the KCV is 3 bytes.

- For MAC generation and validation static derivation is used to generate a unique KAuth per device from the corresponding master key MKAuth.
  The POS/POI will inform about derivation data and the version of the MKAuth and KCV as well.

2.2.3.Non-operational keys

The POS handles K_{EK} and K_{SIG} for different purposes, as shown in 3.1.2.

Key	Owner	Description	Key Type	Master Key
K_{EK}	CGP	Key used for the encryption of the operational keys loaded into the POS/POI.	TDES 192 bits	MK_{KEK}
K_{SIG}	CGP	Key used for the signature authentication of APP's and software pieces being loaded into the POS/POI. Based on CBC ANSI X.9.19 MAC.	TDES 128 bits	MK_{SIG}

Table 2 - Initial POS/POI keys

2.2.4.Certifications

The POS/POI is composed with a PIN-Pad that shall comply with PCI-PTS (PIN Transaction Security) certification. Additionally, in terms of P2PE, it is affected by the Domain 1 (see P2PE).

2.2.5.Communication

The communication with the Entry-Point is based on a secure TLS v1.2 (or higher) session, and can be performed either from private or public networks. The preferred option for the TLS session will be based on PKI and the client (i.e. POS/POI) shall present a certificate, performing a dual authentication session. The native Payment APP can send over this channel the generated ISO8583 adapted frame.

Element	Length	Description
KSN_{PIN}	10 bytes	Key Serial number for the POS/POI. Provides information about the terminal and the BDK id. See 5.1.
KSN_{PAN}	10 bytes	
Key Id BDK_{PIN}	2 bytes	Key identifier for BDK_{PIN} versioning.
Key Id BDK_{PAN}	2 bytes	Key identifier for BDK_{PAN} versioning.
Key Id MK_{Auth}	2 bytes	Key identifier for MK_{Auth} versioning.
Key Id MK_{KEK}	2 bytes	Key identifier for MK_{KEK} versioning.
Key Id MK_{SIG}	2 bytes	Key identifier for MK_{SIG} versioning.
KCV_{PIN}	3 bytes	Checksum of the IPEK.
KCV_{PAN}	3 bytes	Checksum of the IPAN.
KCV_{AUTH}	3 bytes	Checksum of the K_{AUTH}.
KCV_{KEK}	3 bytes	Checksum of the K_{EK}.
KCV_{SIG}	3 bytes	Checksum of the K_{SIG}.
DerivData	32 bytes	Derivation data used to generated non-operative and initial unique keys per POS

Table 3 - POS/POI data elements for operation keys generation

2.2.6. Key generation for PKI

In future the POS/POI shall be able to internally generate a RSA 2048 key pair and a certificate to expose its public key. As the certificate might need to be signed with a public CA, a certificate chain mechanism shall be supported.

Current use cases for this request are:

- TLS with client certificate (see 2.2.5).
- K_{EK} exchange (see 3.1.2).

2.3. Switch

The Switch is responsible for identifying the processor where it has to send the incoming transaction and prepare and cipher data accordingly. It is composed of Entry-Point, Core an Exit-Point. To process a transaction it needs information from the card, normally just the BIN, but there might be other scenarios where the complete PAN is needed. That’s not an issue as the PAN can be handled in clear text inside PCI-DSS area.

As a module centralizing all the cryptographic operations, we introduce here the CryptoMiddleware, the module which centralizes all cryptographic operations in the Switch.

2.3.1.CryptoMiddleware

All the cryptographic functions are part of the CryptoMiddleware component, so it provides a level of abstraction to all the services that need its functionality (i.e. Entry-Point, Core, Exit-Point and potentially others). The CryptoMiddleware exposes the functionality listed in Table 4 – CryptoMiddleware functionality.

Functionality	User	Description
PIN translation	Core	Translation of the PIN, from $KPIN_{session}$ to $KPIN_{exitpoint\_n}$.
PAN decryption	Core	Decryption of the track data (normally with incoming $KPAN_{session}$).
PAN encryption	Core	Encryption of the track data (normally with outgoing $KPAN_{exitpoint\_n}$).
PAN translation	Core	Translation of the track data, from $KPIN_{session}$ to $KPIN_{exitpoint\_n}$.
MAC generation	Entry/Exit -Point	MAC generation for both entry and exit points realtime message exchange using KAuth and $KMAC_{exitpoint\_n}$.
PAN tokenization	Core	Generates a token from a given PAN.
PAN detokenization	Core	Retrieves a PAN from a given token.
Signature	Core	Generates a signature from a binary input.
Key validation	Core	Validation of the diversified keys details per POS according to versioning info reported.
Authentication	Core	Challenge generation and verification for POI/CORE two way mutual authentication.
Key generation	Core	New both operational and no operational diversified keys per POI generation according to BDKs versioning assigned.
MKs/BDKs generation	Core	New BDKs/ MKs generation to renew or initially load operative/non operative keys.

Table 4 – CryptoMiddleware functionality

For details about access and deployment of the CryptoMiddleware refer to section 4.2.

2.3.2.Operational keys

The Switch has to be able to recover for each transaction and for each POS/POI the keys that are being used for sensitive data protection and authentication. That's handled based on a derivation key model where the current key in use can be recovered by:

- Having the KSN of the ongoing transaction and the BDK that was used as a master key (see DUKPT), for DUPKT based keys such as $KPIN_{session}$, $KPAN_{session}$.
- Having the derivation data and versioning info used to generate KAUTH as a unique key per device from MKAUTH.

Basically, the Switch needs to have access to different BDK's (i.e. BDK_{PIN}, $BDK_{PAN,}$) and MKs (MKAUTH) from which originates the different operational keys needed in a transaction (i.e. $KPIN_{session}$, $KPAN_{session}$, KAUTH). For each kind of key and based on its specific BDK and the KSN or MK and derivation data of the transaction, the Switch can always recover the current session/device key.

2.3.3. Non-operational keys

For the exchange of the rest of operation keys based on the DUKPT approach, the Core has to also store an additional MK_{KEK} from which will derive K_{EK}. The same way K_{SIG} will be derived from MK_{SIG} and used in contents download verification.

2.3.4. Certifications

All the components in the Switch having access to the PAN in clear text, or having the ability to recover it, must conform to the PCI-DSS certification.

As the PIN key is the most critical, PCI-PIN establishes that it must be treated with special requirements (FIPS 140-2 level 3). As per P2PE, the requirements for managing the track data keys are equivalent to the ones for the PIN. In terms of P2PE, the Switch is under the scope of Domain 5 and Domain 6 evaluation (see P2PE).

2.3.5. Communication

The communication among the different components of the Switch will be based on HTTPS with at least TLS v1.2 (or higher).

2.3.6. Tokenization

In order to record all the transactions passing through the Switch outside the PCI-DSS zone a representation of the PAN is needed. In addition to this fact, other services could require tokenization (not only for the PAN). For that reason, a tokenizer will be part of the Comercia2.0 solution.

Whatever the tokenization solution is, it shall consist on non-reversible tokens. According to this use case, the tokens in use are known as acquiring tokens, meaning they are no used for transactional purposes.

2.3.7. Operational data (PAN)

For some operational purposes (fraud, blacklists, conciliation file generation, etc.) there might be a need to have a PAN database inside the PCI zone. As already mentioned, PAN cannot be persisted in clear text, so such database shall consider that fact.

The current options are:

a) The PAN database is loaded on volatile memory
b) The PAN database consists on PAN hashes
c) The PAN database consists on tokens of the PAN

3. Key Management

The Key Management system must be reliable and flexible enough to handle hundreds of thousands of POS/POI, each with different keys among them and for each operation. The P2PE certification establishes that the operational scheme to follow is DUKPT (see DUKPT) for sensitive data protection.

3.1. Device keys generation and injection

The device is loaded with operational and non-operational keys, and different schemes are used in each case.

3.1.1. Operational keys (IPEK, IPAN and KAUTH)

DUKPT defines a mechanism where from a given BDK and a KSN, an initial key is uniquely derived for each POS/POI. The KSN is an 80 bit number which identifies each terminal and each transaction.

The process goes as follows (example for PIN). The initial key (IPEK for PIN) is the key loaded in the POS/POI, and from this one and the KSN, the operational (or session) keys are derived. This allows the server side storing just the BDK, but being able to handle many POS/POI. The scheme below (Figure 4 - Key generation scheme) synthesizes how this process is handled.

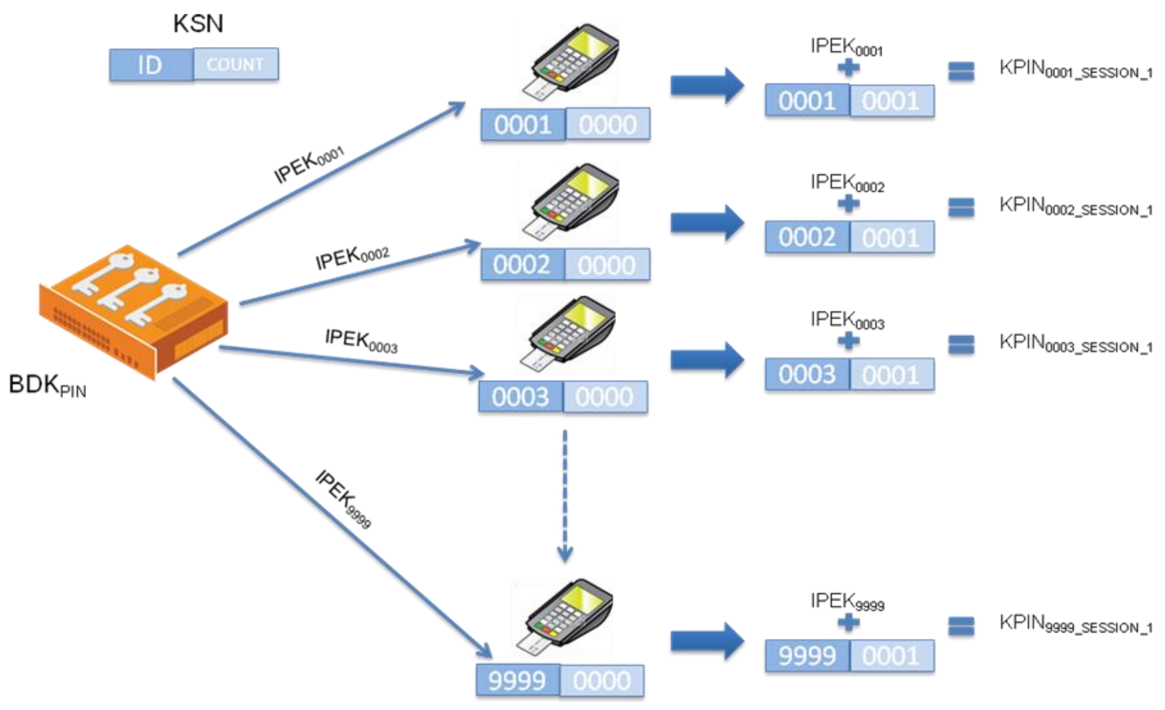


Figure 4 - Key generation scheme for DUKPT operational keys

This process is used for PIN and track data, the initial keys derived are named:

- IPEK (PIN)
- IPAN (PAN)

Initially and for simplicity reasons the KSN root will be shared for the derivation of the different key types, but it could be possible to change this approach in order to use actual different KSNs. This means that there will be a different counter per key stored in the POS/POI that will increase by one with every new transaction. When the counter reaches the maximum it sets starts again setting the counter to 1.

The injection of DUKPT operational keys is performed in the field following the procedures described in section 3.3.2.

MAC key generation (K_{AUTH} from MK_{AUTH}) has been defined using static derivation scheme according to NIST SP800-108 as described in [POS_CRYPTOGRAHY] and graphically in Figure 5.

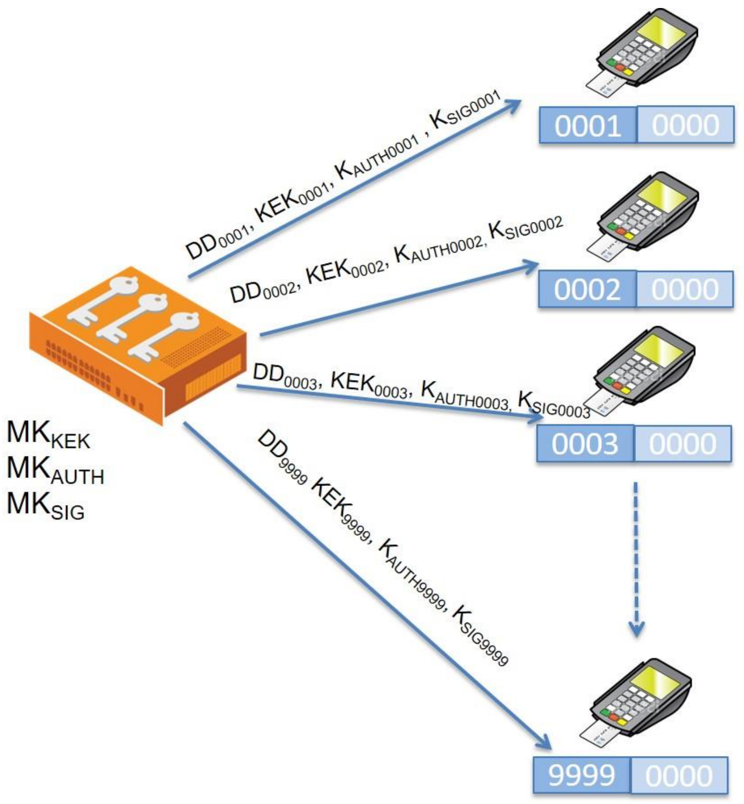
3.1.2.Non-operational keys (K_{EK} and K_{SIG})

A standard derivation scheme is used for the initialization and reload of transport keys of the POS/POI, that is, when loading K_{EK} and K_{SIG}. The derivation of these keys will be based in different master keys: MK_{KEK} and MK_{SIG}, respectively.

The derivation scheme for K_{AUTH}, K_{EK} and K_{SIG} is according to NIST SP800-108.

The first initialization of the POS/POI will be done in a secure environment, normally in specific facilities[1]or vendor secure injection rooms. The requirements for these rooms are defined by PCI-PIN and P2PE. The KSNs will be later injected to the POS/POI during the initial key load process where the DUKPT operational keys will be sent as well.

[1] Diusframi, Servicio10, Intaremit, etc.



DD0001, KEK0001, KAUTH0001 , KSIG0001
DD0002, KEK0002, KAUTH0002, KSIG0002
DD0003, KEK0003, KAUTH0003, KSIG0003
DD9999 KEK9999, KAUTH9999, KSIG9999
0001 0000
0002 0000
0003 0000
9999 0000
MKKEK
MKAUTH
MKSIG

Figure 5 – Key generation scheme for non-operational keys

3.1.3.Redsys

The generation and transport of the keys to communicate with third parties (i.e. $KPIN_{exitpoint\_n}$, $KPAN_{exitpoint\_n}$ and $KMAC_{exitpoint\_n}$) depends on the policies defined by each processor or acquirer entity. The keys to be used are defined in [REDSYS_KEYS_H2H]. Redsys defines in [REDSYS_KEYS] a process to generate and exchange the ZCMK, which is used to cipher before transporting all operational keys defined in Table 5 - Redsys operational keys.

Key	Owner	Mapping to Redsys	Key Type	Master Key
$KPIN_{exitpoint\_n}$	CGP	CGP generates two keys for each Price application, namely CPA1 (principal key) and CPA2 (alternative keys). PIN ciphering uses PIN ISO-0.	TDES 128	N/A
$KPAN_{exitpoint\_n}$	CGP	CGP generates two keys for each Price application, a principal key and an alternative one. Ciphering is done in CBC mode with IV set to null.	TDES 128	N/A
$KMAC_{exitpoint\_n}$	CGP	CGP generates two keys for each Price application, a principal key and an alternative one. The MAC algorithm in use is ANSI X9.19.	TDES 128	N/A
$KPAN_{batch\_n}$	GCP	CGP generates this key to be used to encrypt card data in conciliation file generation.	¿?	N/A
ZCMK	CGP	CGP generates this key by means of three components. This key is	TDES 192	N/A

		used to cipher operational keys.	/ 128	

Table 5 - Redsys operational keys

3.2. Master keys generation and injection

The cryptographic scheme defined is based on the secure storage of several master keys. The master keys, also known in this context as BDK's or MK's, must be generated inside secure modules (HSM's) and never exposed.

3.2.1. Master keys generation

Management and generation of PIN and track data keys have to conform to PCI-PIN and P2PE certifications. This sets a strict level of security in order to manage these keys. That means that a specific HSM has to be installed in a secure area complying with several security checks. Interaction with the HSM will consist only on the usage of its primitives, exposed through the manufacturer's API and some specific management tools.

As the operational keys are transported and by means of K_{AUTH} and K_{EK}, the MK's for those two keys (i.e. MK_{AUT} and MK_{KEK}) are also stored with the same security requirements. This also affects the MK_{SIG} keys.

Comercia2.0 HSM for key management are to be stored in a host provider which is strategically located to provide services for Azure. The hosting provider guarantees the physical and logical level of security needed to comply with PCI-PIN and P2PE.

The specific HSM to be used is to be decided, but it has to comply with:

- PCI-PTS HSM
- FIPS 140-2 level 3

The HSM model that will be used is:

- Atalla AT1000

Generation of keys will be triggered in the master HSM's.

Key	Owner	Description	Derived key	Key Type	Location
BDK_{PIN}	CGP	BDK for PIN keys generation.	IPEK	TDES 128 bits	HSM
BDK_{PAN}	CGP	BDK for track data keys generation.	IPAN	TDES 128 bits	HSM
MK_{AUT}	CGP	MK for key injection authentication generation.	K_{AUTH}	TDES 128 bits	HSM
MK_{KEK}	CGP	MK for key injection encryption generation.	K_{EK}	TDES 192 bits	HSM
MK_{SIG}	CGP	MK for software update signature generation.	K_{SIG}	TDES 128 bits	HSM

Table 6 - Master keys in the Switch

3.2.2.Master keys transport

Transport and protection of master keys (MK's & BDK's) is assured by the HSM's. Depending on the HSM the keys will actually be stored in an external database using a two level keys hierarchy:

- Level 1: Protection key of the system generated internally by the HSM AES 256, namely LMK (or MFK).
- Level 2: Operational keys stored in a database, protected with LMK (BDK_{PIN}, BDK_{PAN}, MK_{AUT}, MK_{KEK} and MK_{SIG}). Transport key ZMK used to keys exchange protected with LMK.

Synchronization of LMK or MFK among HSM shall be performed:

a) by means of automatic procedures (see Figure 6 - HSM)
b) by physical custodians, according to the procedure to be defined by CGP

Synchronization of master keys to third party HSM's shall be performed using:

a) an encryption key previously shared with the third party HSM
b) by means of automatic procedures, if possible
c) by physical custodians, according to the procedure to be defined by CGP

LMK and ZMK management: generation/import/export will be achieved using proprietary vendor tools (Secure configuration assistant → SCA-3).

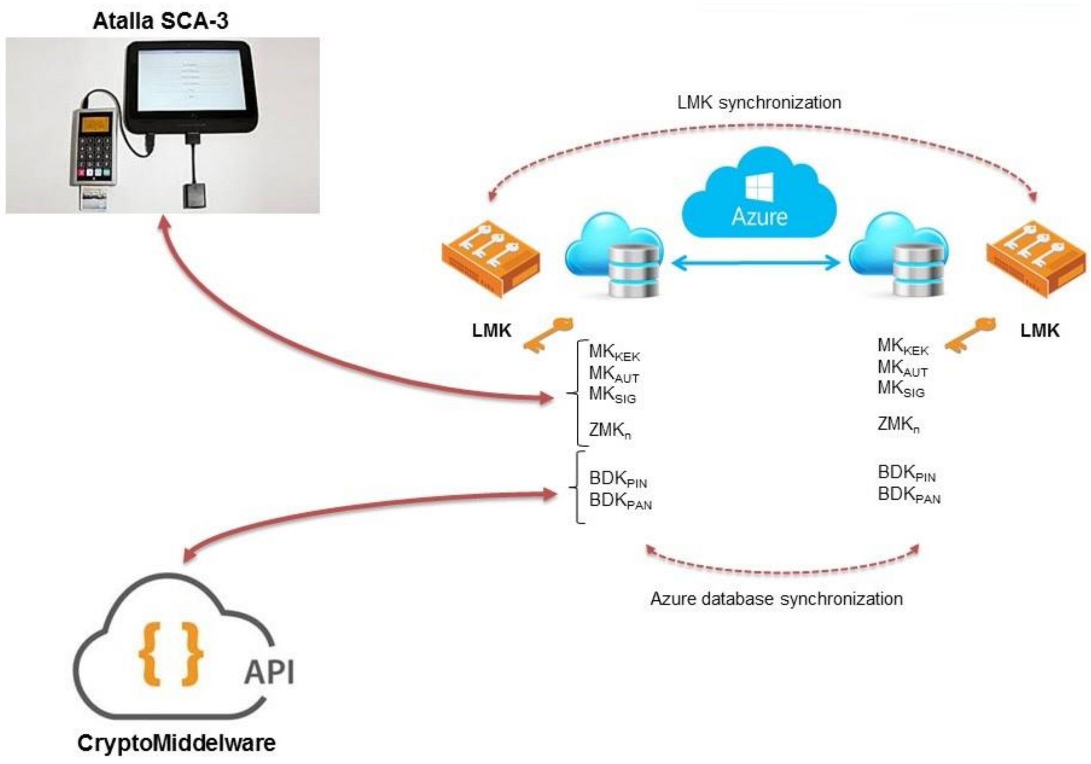


Figure 6 - HSM key generation

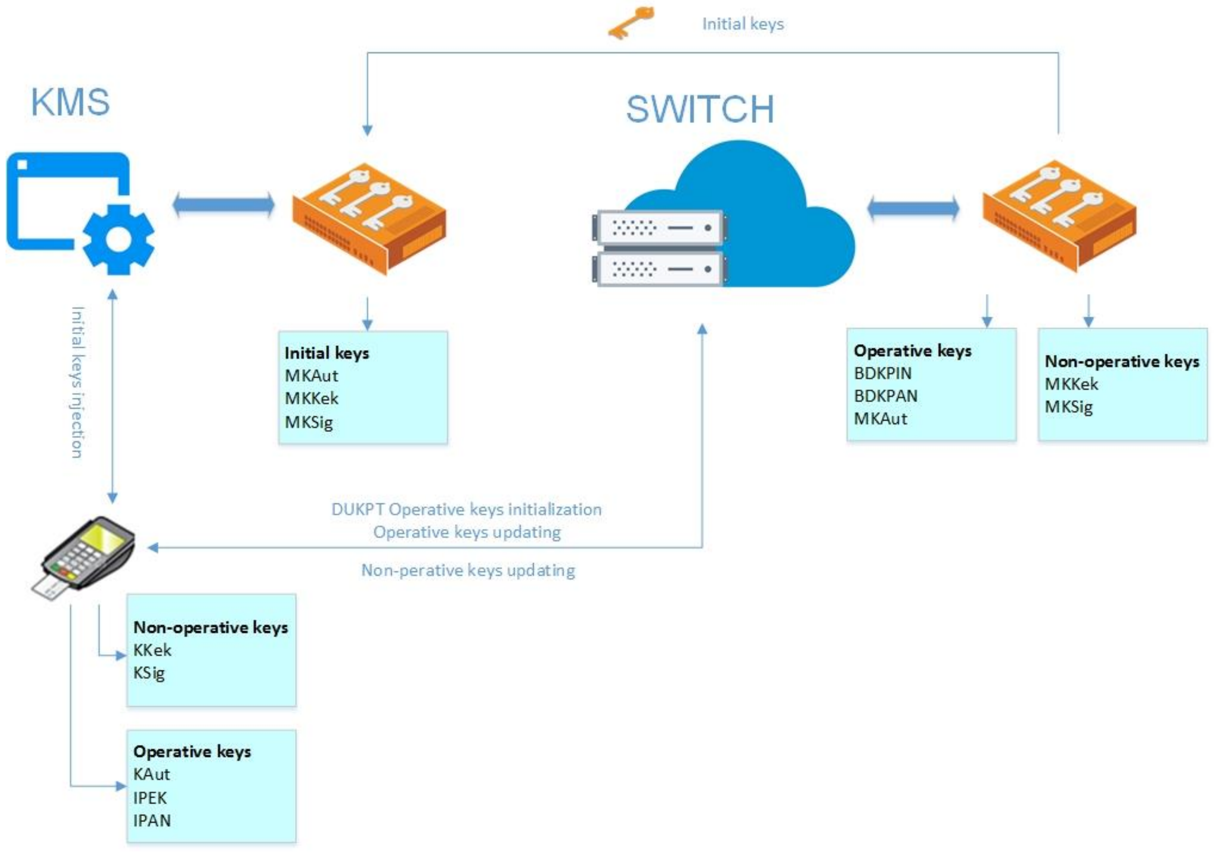


Figure 7 - HSM key ecosystem

3.2.3.Master keys diversification and segmentation

As per control 20-4 on PCI-PIN the DUKPT strategy must incorporate segmentation (i.e. different BDK's) based on one or more techniques (financial institutions, vendors, geography).

According to DUKPT usage of KSN (see DUKPT), CGP will apply segmentation on BDK's using the KSN's Segment Id as defined below (see Table 7 - KSN usage) as described also in [POS_CRYPTOGRAHY].

bits	Name	Usage for CGP	Maximum
16	Not used		
	Segment Id	Key identifier of BDK used for a specific device. Segmentation will be done by generating different BDK's.	
8	IIN	Rightmost byte of the IIN, Issuer identification number	256
8	Customer Id	Value used to differentiate customers	256
8	Group Id	Value use to separate by device type, vendor or model	256
19	TRSM Id	Device identifier. Given a Key Id this field is generated for each device using the same BDK.	524.288
21	Counter	Transaction counter. This counter is incremented for every transaction.	2.097.152

Table 7 - KSN usage

Thus, a device will be loaded with a set of initial KSNs (one per operational key), which actually consists on a Segment Id + TRSM Id + Counter set to zero. That KSN (without considering counters) will identify the POS/POI uniquely.

When a new BDK is generated (see 3.3.1), the Segment Id will remain the same, but the BDK Key Id will change (see Table 3 - POS/POI data elements for operation keys generation).

Note: Although the segmentation strategy exposed here is based on the DUKPT scheme, the same will be valid in terms of key segmentation for the non-operational keys (i.e. MK_{AUT}, MK_{KEK} and MK_{SIG}).

3.3. Key renewal

Although using a cryptographic scheme that generates different keys for every transaction, the process to obtain those keys is based on a common master key. Therefore, it is recommended to temporary renew those keys.

3.3.1. Renewal policy

PCI does not specify the period of live of each key. However, it recommends the following industry best practices (e.g. *NIST Special Publication 800-57*[2]). In a nutshell, the recommendation of key renewal depends on the key type. For symmetric master keys the recommendation is one year, so that's the cryptoperiod that Comercia2.0 will enforce. However, different cryptoperiods can be defined or applied for every kind of key (e.g. BDK_{PIN} might require a more frequent synchronization).

The track of the cryptoperiod of the keys will be done in the server (i.e. the POS/POI is agnostic on the cryptoperiod).

The renewal procedure shall be similar to an atomic operation; however, if the keys are not compromised, current keys shall be valid until the renewal has been completed. In other words, the POS/POI shall be operative at every point of time.

3.3.2. Loading/updating DUPKT operational keys (IPEK, IPAN)

Upon a request from the server to renew the key(s) when the cryptoperiod has elapsed, the POS/POI will receive a key renewal procedure message on the ISO8583 channel.

The following process takes place:

1. A mutual authentication at frame level using K_{AUTH} is performed as explained in section **¡Error! No se encuentra el origen de la referencia.**.
2. A new version of the different BDK's is generated in the HSM (see Figure 4 - Key generation scheme).
3. With the KSN of the POS/POI corresponding to the given key type assigned and the BDK_X the new initial keys are derived in the HSM (i.e. IPEK, IPAN).
4. The newly generated key is ciphered with K_{EK} key. The frame includes a MAC generated with the K_{AUTH} key. The frame is sent to the POS/POI. KSN for this key is also included in the frame.
5. The POS/POI verifies the MAC and stores the new initial key and KSN.

[2] http://nvlpubs.nist.gov/nistpubs/SpecialPublications/NIST.SP.800-57pt1r4.pdf

3.3.3. Loading of non-operational keys (K_{EK}) and static operational keys (K_{Auth})

Upon a request from the server to renew the key(s) when the cryptoperiod has elapsed, the POS/POI will receive a key renewal procedure message on the ISO8583 channel.

The following process takes place:

1. A mutual authentication at frame level using K_{AUTH_i} is performed as explained in section **¡Error! No se encuentra el origen de la referencia.**.
2. A new version of the MK_{AUT_i+1} and MK_{KEK_i+1} are generated in the HSM.
3. With the derivation data of the POS/POI and the MK_{AUT_i+1} and MK_{KEK_i+1}, new K_{AUTH_i+1} and K_{EK_i+1} are derived in the HSM.
4. The newly generated keys K_{AUTH_i+1} and K_{EK_i+1} are ciphered with the previous K_{EK_i} key. The frame includes a MAC generated with the previous K_{AUTH_i} key. The frame is sent to the POS/POI.
5. The POS/POI verifies the MAC and stores the new K_{AUTH_i+1} and K_{EK_i+1} keys.

3.3.4. Loading of signature key (K_{SIG})

Upon a request from the server to renew the K_{SIG} when the cryptoperiod has elapsed, the POS/POI will receive a key renewal procedure message on the ISO8583 channel.

The following process takes place:

1. A mutual authentication at frame level using K_{AUTH} is performed as explained in section **¡Error! No se encuentra el origen de la referencia.**.
2. A new version of the MK_{SIG} is generated in the HSM.
3. With the KSN of the POS/POI and the MK_{SIG} a new K_{SIG} is derived in the HSM.
4. The newly generated key K_{SiG} is ciphered with the K_{EK} key. The frame includes a MAC generated with the K_{AUTH} key. The frame is sent to the POS/POI.
5. The POS/POI verifies the MAC and stores the new K_{SIG} key.

3.3.5. Key recovery in the field

In future there shall be a PKI mechanism in place, to be used just in case the non-operational keys get compromised. In this situation, the POS/POI would generate a key pair and will send its public key to the key manager in the Switch, so a new key set can be injected.

3.3.6. Redsys

The renewal of keys with Redsys is to be defined by CGP, so the policy to apply is the same than for all other keys (see 3.3.1). The process to renew the keys is specified in [REDSYS_KEYS].

4. Architecture

The whole system has to be designed taking a few considerations into account. The main goals are to achieve a low latency with a solution compliant to PCI and with a design as much reusable as possible.

The scheme below (Figure 8 - Arquitecture defined for the cryptography) is a representation of the different components and how they interact. As already introduced in section 2.3, we have some relevant components.

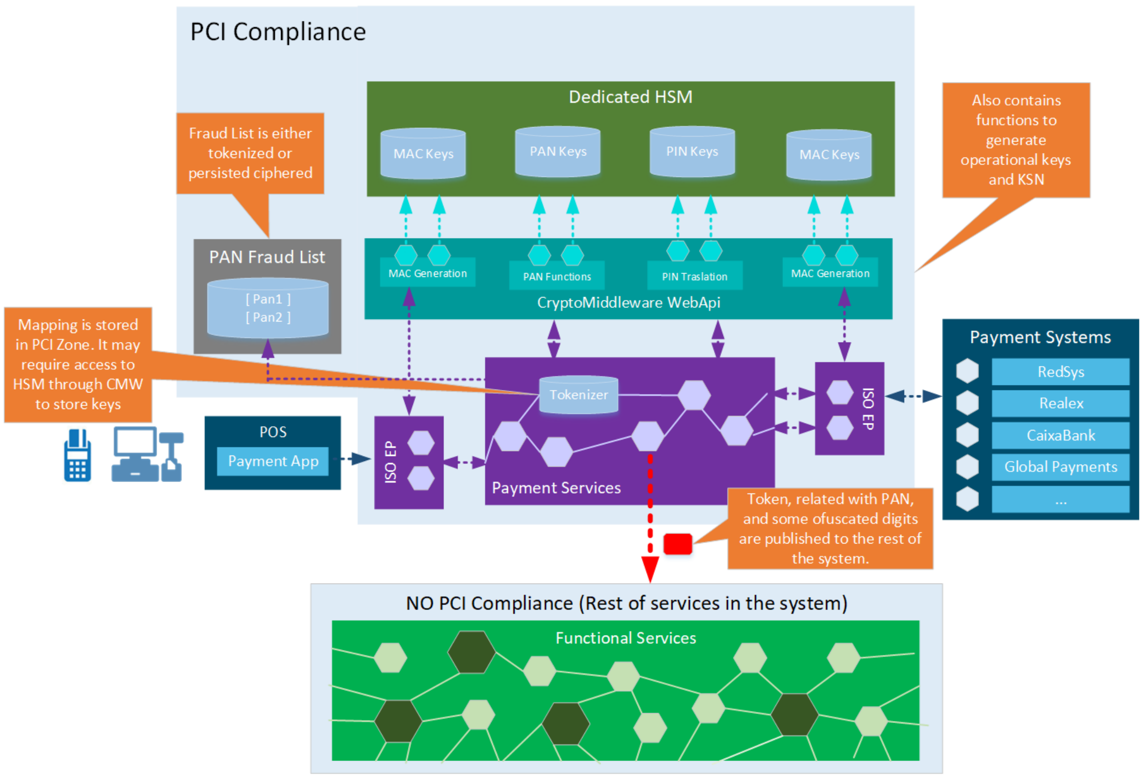

PCI Compliance
Dedicated HSM
MAC Keys
PAN Keys
PIN Keys
MAC Keys
Fraud List is either tokenized or persisted ciphered
Also contains functions to generate operational keys and KSN
MAC Generation
PAN Functions
PIN Traslation
MAC Generation
CryptoMiddleware WebApi
PAN Fraud List
[Pan1]
[Pan2]
Mapping is stored in PCI Zone. It may require access to HSM through CMW to store keys
Tokenizer
ISO EP
Payment Systems
RedSys
Realex
CaixaBank
Global Payments
...
POS
Payment App
ISO EP
Payment Services
Token, related with PAN, and some ofuscated digits are published to the rest of the system.
NO PCI Compliance (Rest of services in the system)
Functional Services

Figure 8 - Arquitecture defined for the cryptography

4.1. HSM's

The rationale for having a dedicated storage component for key management is due to the need to be compliant with PCI-PIN and P2PE certifications. This component is most probably adding latency to the system, besides on the complexities of its management.

The model for PIN dedicated HSM is:

- Atalla AT1000

4.2. CryptoMiddleware

The CryptoMiddleware has been designed in order to centralize all the cryptographic functionality, providing a level of abstraction to the Core. This will allow in the future the transition to other HSM's without affecting the Core implementation.

The architectural solution for deploying this module is to have DLL's in a PaaS, exposing its methods via WebApi. In such a way, the CryptoMiddleware component can be easily isolated from being called by all the services in the Core.

For both client authentication and requests authorization it is desired to use HTTPS with client certificate, based on TLS v1.2 (or higher). The minimum length of the certificates shall be 2048 bits. Data ciphering groups are limited to AES256 with SHA256 or higher.

As the PaaS model in Azure does not verify client certificate[3], it shall be responsibility of the application to verify it.

4.3. Tokenization

The tokenization solution is under analysis, as it can be implemented by software or making use of an appliance. The preferred solution is to have it implemented by software to avid maintenance and storage of dedicated hardware.

Anyhow, the tokenization will provide service to the Switch, as the PAN cannot be exported to the external databases that will be consumed from the non-PCI components, but probably also to external services (not necessarily tokenizing a PAN).

[3] https://docs.microsoft.com/es-es/azure/app-service-web/app-service-web-configure-tls-mutual-auth?toc=%2fazure%2fapp-service-api%2ftoc.json

5. Annexes

5.1. DUKPT

DUKPT (Derived Unique Key Per Transaction) is a key management scheme which generates a unique key per transaction. It is specified in [ANSI X9.24-1:2009], so refer to that specification for further information.

Based on a BDK it can handle several POS/POI with different keys each. It does so deriving from BDK as many IPEK (Initial PIN Encryption Keys) as POS/POI. This derivation is done taking the KSN (Key Serial Number) as input. Obviously, each POS/POI has a different KSN.

The KSN is an 80 bit (10 byte) string which consists of:

- 40 bits: Key Set ID
- 19 bits: TRSM ID
- 21 bits: Transaction counter

It is a common practice to actually use a 64 bit (8 byte) model, ignoring the two most significant bytes and using 6 nibbles for (A) Key Set ID, (B) 5 nibbles for TRSM ID and (C) 5 nibbles for the transaction counter (i.e. a KSN would look like 0xFFFFAAAAAABBBBBCCCCC).

5.2. P2PE

Point to Point Encryption is a standard promoted by PCI, defined in [PCI-P2PE], oriented to data encryption extreme to extreme (i.e. from POI to acquirer) where the merchant has no availability of encryption keys. This standard complements other PCI ones (i.e. PCI-PTS, PA-DSS and PCI-DSS).

By conforming to this solution, merchants have a reduced set of checks to be passed compared with a solution not compliant with P2PE.

P2PE defines six domains to be analyzed:

1) Encryption Device and Application Management
2) Application Security
3) P2PE Solution Management
4) Merchant-managed Solutions
5) Decryption Environment
6) P2PE Cryptography Key Operations and Device Management

For further information and details please refer to official PCI P2PE documentation (https://www.pcisecuritystandards.org/).

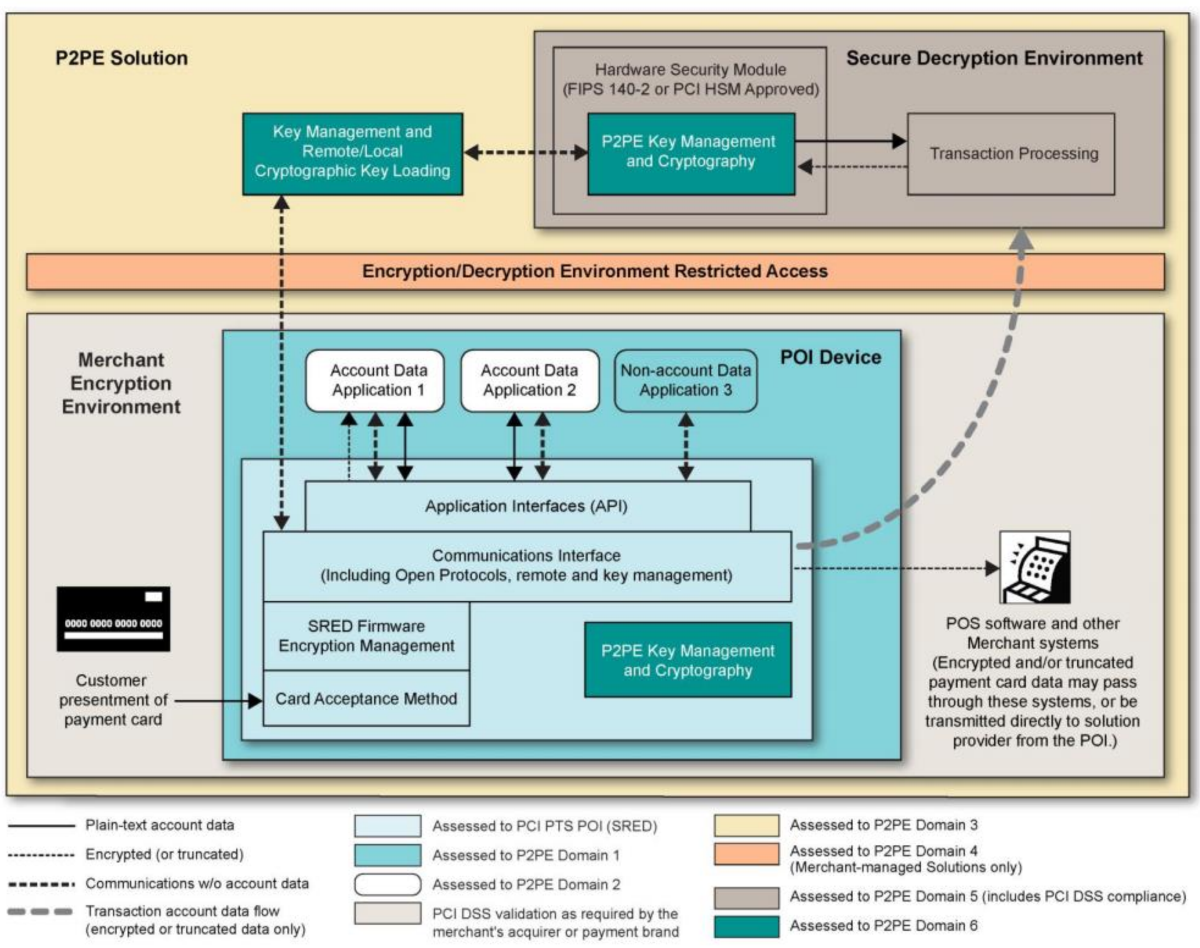

P2PE Solution
Secure Decryption Environment
Hardware Security Module (FIPS 140-2 or PCI HSM Approved)
Key Management and Remote/Local Cryptographic Key Loading
P2PE Key Management and Cryptography
Transaction Processing
Encryption/Decryption Environment Restricted Access
Merchant Encryption Environment
POI Device
Account Data Application 1
Account Data Application 2
Non-account Data Application 3
Application Interfaces (API)
Communications Interface (Including Open Protocols, remote and key management)
SRED Firmware Encryption Management
Card Acceptance Method
P2PE Key Management and Cryptography
0000 0000 0000 0000
Customer presentment of payment card
POS software and other Merchant systems (Encrypted and/or truncated payment card data may pass through these systems, or be transmitted directly to solution provider from the POI.)
Plain-text account data
Encrypted (or truncated)
Communications w/o account data
Transaction account data flow (encrypted or truncated data only)
Assessed to PCI PTS POI (SRED)
Assessed to P2PE Domain 1
Assessed to P2PE Domain 2
PCI DSS validation as required by the merchant's acquirer or payment brand
Assessed to P2PE Domain 3
Assessed to P2PE Domain 4 (Merchant-managed Solutions only)
Assessed to P2PE Domain 5 (includes PCI DSS compliance)
Assessed to P2PE Domain 6

Figure 9 - P2PE impledmentation