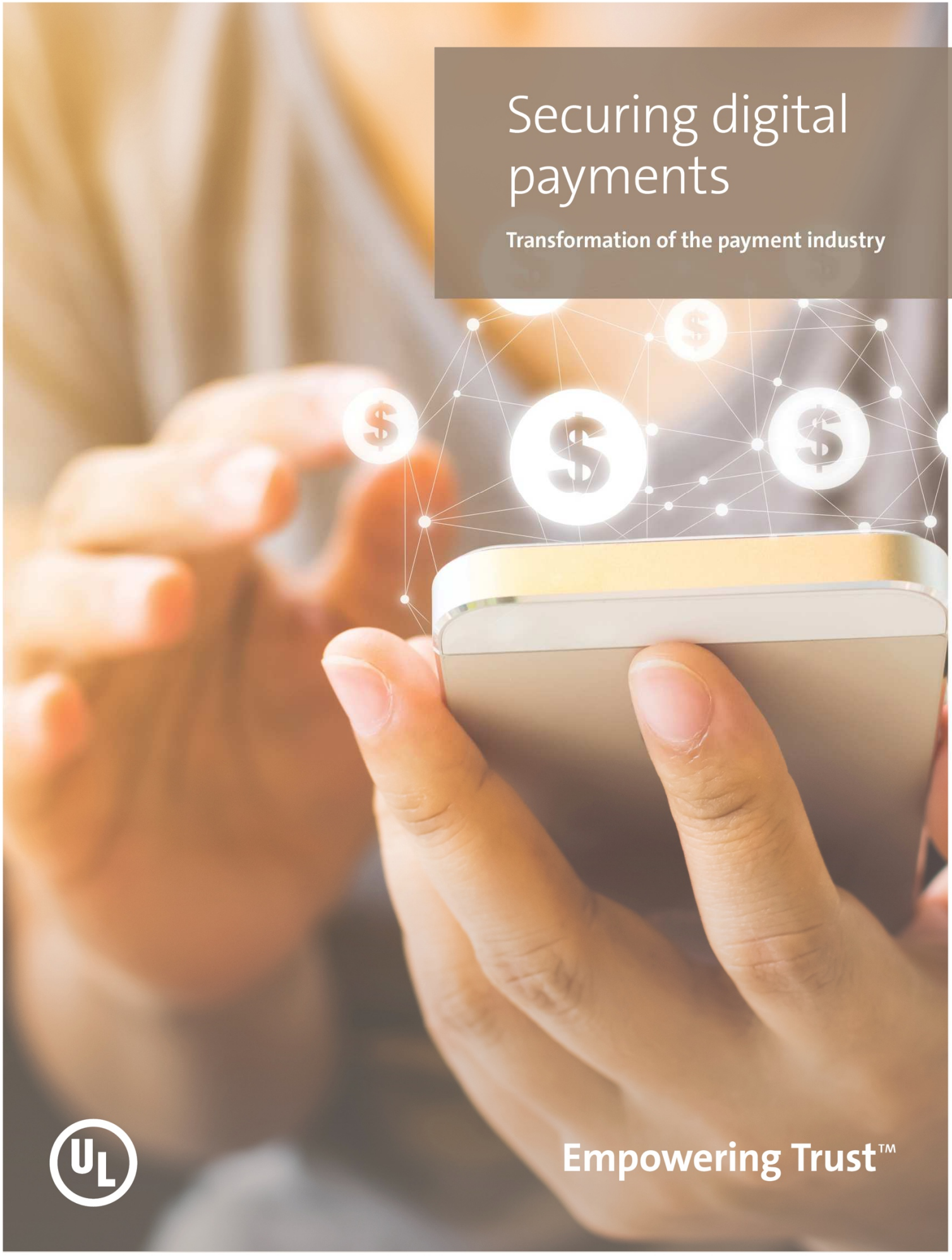
Securing digital payments

Transformation of the payment industry



Empowering Trust™

This book is primarily aimed at people who are currently working in the payment industry, work with the payment industry, or who are looking to move into this space. After many decades of things being mostly the same, the payment industry is undergoing a massive transformation as processes and data are digitized, Fintechs challenge the incumbent methods of operation, banks open up their interfaces for 3rd parties, and the market drives a need for secure, real-time commerce from anywhere to anywhere.

For those of you already in the payment industry, I'd be willing to bet you got here by accident. Our industry is often one where people 'end up' – not in a bad way, but not necessarily through deliberate action. This is changing somewhat with the push for Fintechs to shake things up, but rare is the child who you meet that proudly tells you they want to work in payments when they grow up. Because of this, it's often hard to find good information on the exact how, why, and what's of payments. You can't go to University and major in Payments, and you certainly don't learn anything about the underlying mechanisms of how payments work at school, or during a Comp Sci or Engineering degree.

If no formal education path exists, how do people get this knowledge? Generally, through hard work, sweat, learning on the job, and independent research (and no small amount of tears, if my own experience is anything to go by). There is no one source of truth that covers the breadth of the payment industry, and even for those who have been working in payments for decades there is often gaps in their knowledge based on their own personal experiences.

So, this book is my attempt to help solve that problem. I'm not quite arrogant enough to believe it's a one and done thing, that with this simple document I have solved the problems I have listed above, but it's intended to be a start. Indeed, I expect to create a v2 of this book soon, adding in items that are currently missing – if you have feedback or things you'd like to see, let me know.

For those of you starting in the payment industry, this will give you an overview of how payments works, the various standards that exist and how they interact, and the main stakeholders you may come across.

For those of you who have been working in payments for many years, hopefully this book will help fill in some of the gaps in your knowledge, or act as a reference for the more esoteric details so you don't need to memorize the difference between a format 0 and format 3 PIN block. The content herein is absolutely not comprehensive; if you're an EMV guru, you're not going to learn anything new from the section that talks about that, but you may just find some interesting bits you didn't know about terminals that accept EMV cards, the key management used in payments, or how mobile payments works.

For those of you embarking on the Fintech journey looking to disrupt the status-quo, this book will help you understand that status-quo, hopefully helping you spot the niches into which you can drive your own personal wedge.

It's a bit of a cliché to say that I wrote this book to be the introduction to my field I missed when I started my career, but it's certainly true. I hope you find it useful. Please do provide feedback and stay tuned for the updated edition!

Andrew Jamieson
@andrewrjamieson

Table of Contents

It's fair to say that the last few years have been a busy time for those of us working in the payment industry, and especially so for those who have been working on payment standards. Many standards, from the way we perform key management to the way we encrypt customer PINs, have been updated to deprecate older cryptographic methods, to address changes in the threat landscape, or to adapt to new ways that customers expect to be able to perform payments. Software has continued to eat the payment industry as well, with many of the new standards allowing for use of commoditized hardware to perform payment processes previously reserved for dedicated systems. New payment channels and mechanisms have emerged, using QR codes and biometric authentication, to challenge the incumbent processes.

Trying to capture the full details of each aspect of the payment industry across all geographies, into both the past and future, is difficult. This book attempts to focus on the aspects of history which are required to understand the how of where we are now, and the why of where we may be going. So, this book does not attempt to provide comprehensive details on the entire payment industry and all of its many facets – if you want comprehensive details on things like EMV (for example), there are many academic and industry documents that will give that to you, or you can seek the 1000+ page truth of the specifications themselves.

The book is divided into three main parts – an overview section which gives some background on payments and their history, a section that details each of the standards of import which impact or influence the payment process (this is the longest section), and a series of Appendices which provide some technical overviews and references which are useful throughout reading of this book.

Lions, and Tigers, and Bears – Oh my!

In addition to the plethora of different standards that exist, you can also categorize different types of standards, and different methods of assessment to these. Standards range from the very detailed and prescriptive, through to very high level without much implementation detail. As a general rule, standards that assess the functional aspects of a device tend to be quite proscriptive, both because the details of exactly how each datagram is formatted and transmitted is important, and also because as a functional standard it is possible to define exactly how an implementation should operate.

Non-functional security standards tend to be different, being less about how something should be done, and more about what the implementation should achieve – the assets it must protect, the types of attack it should resist, how much effort may be applied to extract secret data. Here details are deliberately left out to foster innovation in the way in which security is provided to the device or system, as well as to avoid proscribing a specific type of security, a specific implementation, that may at some future point become subject to compromise.

However, even with security there are different levels of detail applied to standardization. Some standards define high level goals – "Cryptographic keys must be protected from disclosure and misuse" for example – and leave it at that. Others will take the high level requirement and go into further detail on how that requirement should be validated, creating what is often referred to as "Derived Test Requirements" or DTRs.

In addition to the standards themselves, there is the way in which those standards are assessed, usually being one of the following three:

- Audit
- Evaluation
- Penetration test

An audit consists of a point in time check of implementation, where review of documentation, logs, records, interviews, and sampling of systems configurations, etc, are used to confirm that the target of the audit is compliant. An audit does not involve actual attempts to exploit vulnerabilities, although it may include requirements to have this done and the records/output of that would be validated during the audit.

Audits are generally time bound, and requires full details of the implementation to be provided to the auditor to review. PCI DSS and PCI PIN are examples of audits.

An evaluation is a review of a very specific implementation of a system or process, such as a particular device or computer system. Like an audit, evaluations require that full details are provided of the implementation, but often the level of detail required goes further than an audit, requiring in-depth information such as source code, physical design files, etc. An evaluation will usually involve some types of testing, including attempts to bypass aspects of security controls as part of a security evaluation. However, the evaluator may use the details they have been given to choose the best type of attack or test to perform – which can sound like 'cheating', but the intent is to remove the uncertainty and increased time required to reverse engineer this information, focusing solely on testing of the potential issues.

Evaluations are results bound, with a set of requirements that must be met before the process is complete. PCI PTS and EMV ICC testing are examples of evaluations.

Finally, a penetration test is an attempt to extract or gain access to specific assets, with little or even no help from the designer or implementer of the system under test. In penetration testing, the tester must reverse engineer, socially engineer, or find other sources to gain the information they need to perform the test and gain access to the assets. Penetration testing is usually time bound, and in fact one of the problems with a penetration test is that the findings can often be very binary – either the test was successful or it was not, and if not successful it is very difficult to say if another hour or another month of testing would have been required to succeed.

Many of the payment security standards require penetration testing to be performed, but are not solely penetration testing standards themselves.

The Payment Players

When it comes to standards for payments, the stakeholders can usually be sorted into four main groups:

- Global payment industry bodies, such as PCI SSC and EMVCo
- Global standards setting bodies, such as ANSI, ISO, or NIST
- Global payment brands, such as Visa and Mastercard
- Domestic and regional payment brands, such as EFTPOS in Australia or Interac in Canada
- New players (Fintechs) such as WeChat, AliPay, or AfterPay

Each of these bodies has a different role in the payment eco-system, and often understanding their role and history helps to frame an understanding of the standards that they produce. One body can often influence another, such that with some experience you can start to see what may be coming into the future with one body by looking at what another is doing right now.

With this in mind, let's take a bit of time to have a look at the history and role of each of these groups.

The PCI SSC, or the Payment Card Industry Security Standards Council, is responsible for setting and managing security standards for the global payment card transaction market. The PCI SSC body was created by five of the card brands (Visa, Mastercard, JCB, Discover, and American Express) in 2006 for this specific purpose – to drive a globally consistent minimum standard for security in all important aspects of card payments.

PCI still boasts members from the five founding card brands as part of their 'Executive Committee' who are responsible for approving the security standards that are produced and published. In addition to the Executive Committee, PCI also have a number of industry out-reach mechanisms to involve other payment brands and industry stakeholders. This includes over 750 Participating Organisations (POs), who are provided with early access and feedback rights to new standards, a Board of Advisors which is sourced from the POs who help steer the content of the standards, various working groups and task forces which provide input to the standards as they are produced, affiliate members from other (often local) payment brands who help with the inception of standards, as well as other mechanisms.

Most often, when people think of "PCI" they think of one particular aspect of the entirety that is the PCI SSC. The reality is that it is a large and multi-faceted organization that produces a large range of content and standards.

EMVCo was also formed by the card brands (Europay, Mastercard and Visa at the time – the E, M, and V), but this time to help standardize the method for storing and communicating payment data from a 'chip' on a payment card. Since then, the purview of EMVCo has expanded to cover more than 'just' chip transactions, such as online commerce and software based mobile payments issuance, but the purview remains focused on interoperability for transactions and security on the Issuance and use of payment data on the customer side.

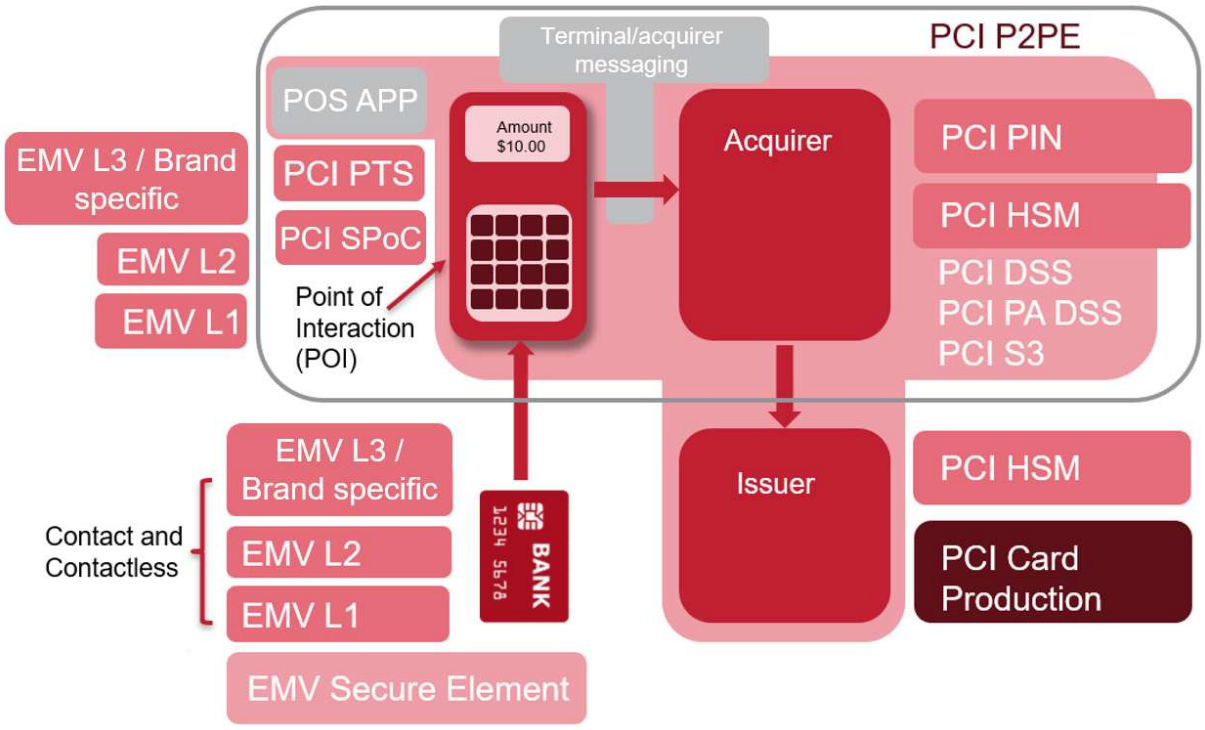
EMVCo also has various methods for outreach and development of standards, similar in many ways as it also involves various task forces and working groups, as well as advisory boards and an executive committee. However, where PCI have 'Participating Organisations' and 'Affiliate Members', EMVCo instead have 'Subscribers', 'Technical Associates', and 'Business Associates'.

Without adding too much needless complexity it's generally safe to say that EMVCo is responsible for making sure that payment card based transactions work, and the PCI SSC is responsible for making sure these can be conducted securely. There is some cross-over in some areas where this simple rule fails, but it's generally correct in most cases.

The various payment brands obviously feature heavily in both PCI SSC and EMVCo, but also have their own role to play in standardization. The structures of the payment brands is less complex than that of PCI SSC or EMVCo, and obviously they do not involve multiple competitive parties, so often the payment brands find it quicker and easier to publish their own standards – at least initially. Indeed, there can be an observable trend that the brands will publish 'best practice' guidelines, then a standard, and then have that standard consolidated with other brands documents by PCI SCC or EMVCo to become an industry-wide standard.

The global standards bodies, such as ISO and ANSI, play a supporting role to many of the other bodies discussed here. They set over-arching standards, such as those for key management, encryption, PIN Block formats, etc, and as such support the more specific standards implemented through the other bodies.

And then, of course, we have the fintechs who are driving innovation in the payment space. Often this is done without 'standardisation' as it is generally understood, but to grow market share there will often be open APIs or interface rules that allow for other players to leverage the same technology, or to provide interfaces for acceptance or issuance. The pressure applied by these parties can often have a strong influence on the new or changing standards that are output from the bodies discussed above.



The 'four corner' model of payments, with associated standards

Although it is facing increasing pressure, the traditional view of payments as the 'four corner' model (where four separate parties are involved in each transaction; the customer, the merchant, the merchants Acquiring Financial Institution, and the customers Issuing Financial Institution) remains valid for most, at least in the card-present world.

However, even here we see new players and new standards changing the way that a payment can be processed.

There are two main things underlying many of the changes in payment standards over the last few years:

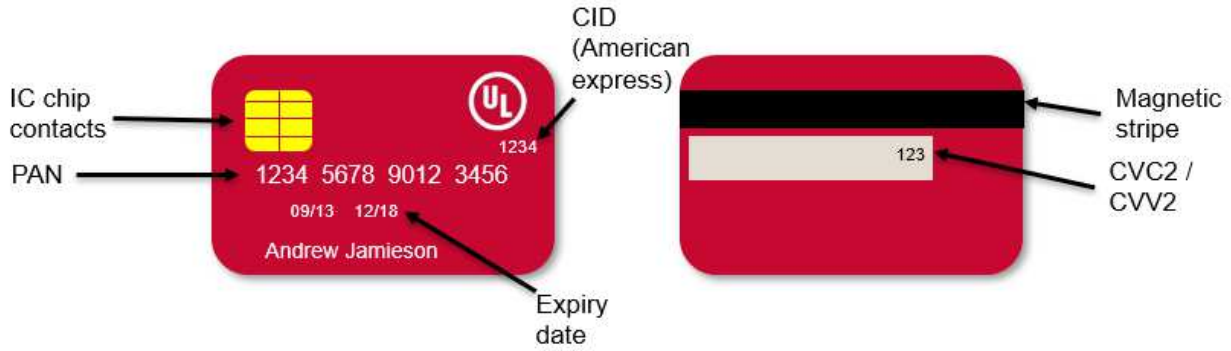
1) A move from static data to dynamic data – changing what was once a global fixed value into something that is unique each time, or can more easily be changed when compromised.
2) The 'commoditization' of payment systems – increasingly both of the 'horizontals' of the four corner model of payment, representing the customer and merchant relationships (Customer to Issuer, Merchant to Acquirer), are facing increased pressure to replace the bespoke mechanisms used, such as POS terminals and payment cards, with software running on commoditized hardware, such as tablets and mobile phones.

These two items combined are driving a rapid change in the payment landscape, both to reduce fraud and to compete against new payment methods being introduced.

The payment infrastructure we use today has been around for some few decades now, but often people don't think too much about how it works or how we've managed to get so many different countries to all agree on interoperable ways to make payments. Some history on the basics of payments, and historic ways of paying that still persist today, is useful to understand the context of how we got here, why current standards exist, and what may be coming into the future to continue to secure payments.

Working Card, or Card-ly Working?

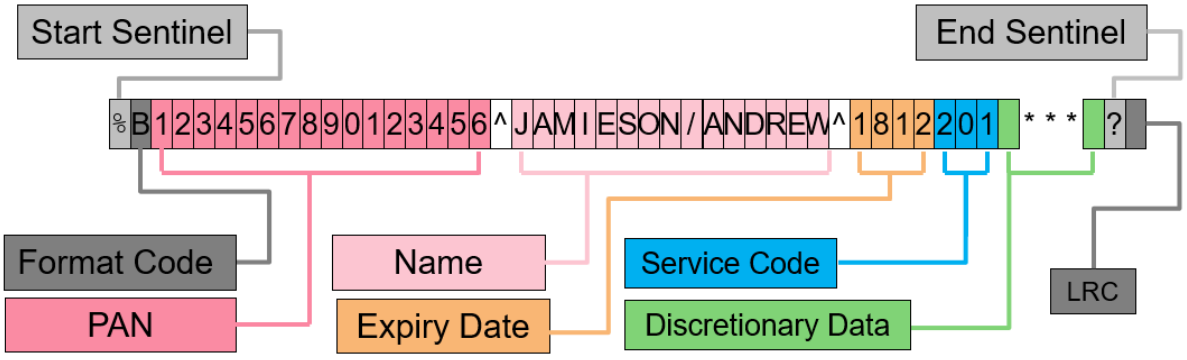
A physical payment card can be made up of a number of different elements, as shown below.



The elements of a traditional payment card

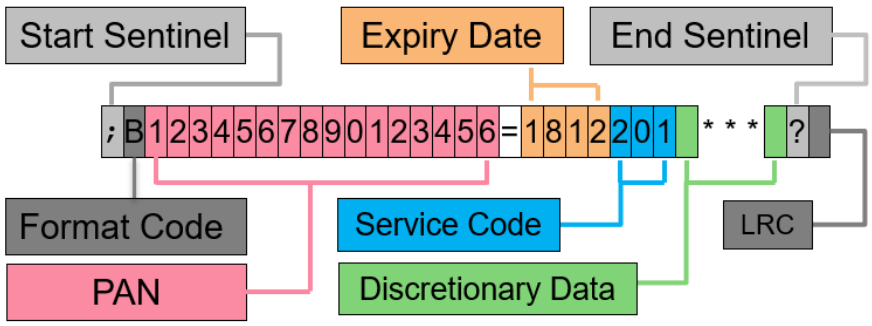
The main identifier for such a card is the Primary Account Number (PAN) which is usually printed and embossed on the front of the card. The first six digits of this PAN are the Issuer Identification Number (IIN), or Bank Identification Number (BIN), which identifies the specific financial institution which issued that card to the customer. When you use your card to make a payment with a merchant that is not directly connected to your bank, the transaction is routed through to your bank based on the knowledge of this IIN/BIN number. All but one of the remaining digits of the PAN are used to uniquely identify your actual bank account with that Issuing institution (although some digits may also identify certain sub-products within that issuer, such as frequent flyer cards, etc). The very last digit of the PAN is a check digit, called the Luhn digit, which is used to detect single digit errors during entry of the number.

Most of the data from the front and back of the card is also recorded on the magnetic stripe, in up to two of the three tracks that may be present on this strip. The first of these tracks (track 1, surprisingly) contains the most data, replicating everything but the CVC2/CVV2 value – which we'll get into later.



Track 1 data from a magnetic stripe payment card

The track 2 data is a shorter version of the data contained in track 1, dropping the customer name, but retaining the all-important PAN value.



Track 2 data from a magnetic stripe payment card

Data on the magnetic stripe is read using a 'readhead' on the payment terminal, which picks up on changes in magnetic field as the card is moved across small gaps in the head. This data is able to be read by anyone who has a readhead positioned within the card path, and this is how most skimmers work – positioning a secondary readhead in the path of the card as it is entered or swiped into the payment terminal or ATM.

Beyond the PAN, there are a few other important values contained on the magnetic stripe, such as the service code which is used to identify features of the card – for example if it has a chip or not. If you've ever swiped a card through an Magnetic Stripe Reader (MSR) only to have it ask you to insert the chip, this is because the terminal has figured out you have a chip on your card due to the service code on the magnetic stripe.

Also on the card stripe is a section of 'discretionary data', which are values set by the card Issuer based on their own needs. This will probably include values such as a Card Verification Code (CVC), or Card Verification Value (CVV) that is a cryptographically generated value that ties the card PAN to the expiry date and service code. It may also include a PIN Verification Value (PVV) as well, which does not store the PIN and cannot be used to calculate or derive the PIN, but can be used to validate the PIN with access to the right cryptographic keys (more on this later).

This CVC/CVV value is similar to the CVC2/CVV2 that is printed on the rear of the cards, the two are generated through a slightly different methods to ensure that the value on the magnetic stripe is different from the value printed on the card. This provides 'channel separation' for the card usage – compromise of card data from an online store cannot be used to create a magnetic stripe card as it does not have the correct CVC/CVV value. Similarly, compromise of a magnetic stripe value would not allow for use in an online shop where the CVV2/CVC2 value is required.

This speaks to the two main methods of using a payment card – 'card present' (CP) and 'card not present' (CNP). These concepts will pop up throughout this document.

CVM - I am not Just a Number

Most people will be used to using a four digit Personal Identification Number (PIN) to authorize some forms of card present transactions – usually when money is to be withdrawn from an Automatic Teller

Machine (ATM), or when there is a larger transaction amount involved. The PIN is a type of Cardholder Verification Method (CVM), of which there are four main types:

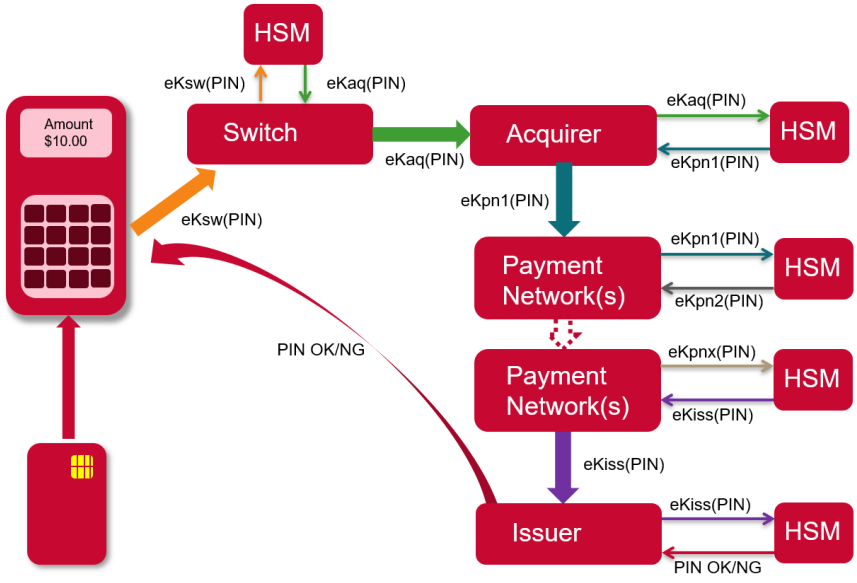
- None
- Signature
- PIN
- Biometric / On Device CVM / Consumer Device CVM

The 'none' and 'signature' CVMs are relatively clear – either no additional authentication is required beyond the presentment of the card itself, or a signature is required on the receipt or screen of the payment terminal – but the PIN and other CVM methods are worth expanding on a bit more.

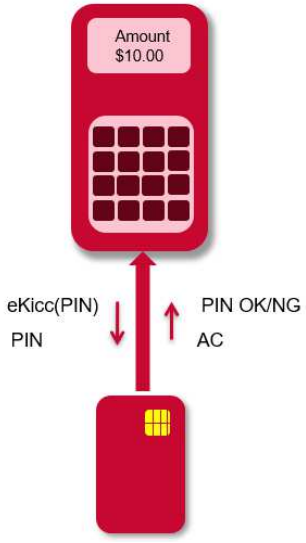
Have you ever wondered how your PIN, entered on a payment device in an arbitrary corner of the world, can be validated as correct when there is no clear relationship between that payment terminal and your Issuing bank? Part of this is based on the methods by which the transaction can be accepted at all – the global payment networks implemented by the larger payment brands which connect the various countries and banks throughout the world – and part is dependent on the way in which the PIN is handled.

Going off the Grid - Offline PINs and Offline Transactions

There are two primary types of PIN verification – online PIN and offline PIN. An online PIN is encrypted by the PINPad into which it is entered by the customer, into a format called a 'PIN block' (more on this later). The encrypted PIN block is then transmitted encrypted through the various networks and interfaces that connect that terminal to the Issuing bank, or an agency to which it has delegated PIN verification functions (it does this by providing that agent with the PVV keys discussed earlier). To ensure the PIN is properly protected using cryptography on each 'hop' through the networks, at each point it is passed into a Hardware Security Module (HSM) to be 'translated' from encryption under one key to encryption under the next key.



Online PIN processing



Offline PIN processing

This is illustrated visually in the diagram on the left above, with the PIN being translated under different (PIN encryption) keys until it is finally encrypted under the key of the Issuing bank (Kiss in this diagram), and verified as correct or incorrect in the Issuer HSM (for more details on HSMs, refer to the section on PCI PIN and PCI PTS later in this book).

It is possible in an online PIN scenario for one of the intermediate connections to verify the PIN using the PIN Verification Value (PVV) we discussed earlier, if the Issuer has supplied their PVV keys to that entity. The method for doing this is out of scope of this book, but put simply using cryptographic keys the Issuer provided to another entity that entity can validate any PIN and card data combination as correct. The PVV does not contain the PIN, and it is not possible to take a PVV and using just that determine the PIN value. PVV use varies across the world depending on a multiple of factors, but mostly online PINs are validated by the institution that Issued the customer card as illustrated above.

Offline PIN is quite different. In an offline PIN scenario, the customer PIN is transmitted directly from the terminal to the customer card for validation – it is never sent to the Acquiring or Issuing backends, and effectively the PIN never 'leaves' the tamper responsive boundary of the terminal (as the customer card is contained within the card slot). A PIN block is still used, but the format is different for offline PIN as opposed to online PIN. Offline PIN is often used when there is a concern around the connectivity for transactions, so that the PIN can be used and verified even when there is a loss of communications or only the ability to communicate in periodic 'batches' (such as if you were on a plane, ship, or using a ticket machine that did not have real-time connection to a broader network).

The transmission of the customer PIN from the terminal to the card may occur either in plaintext or encrypted under a PIN encryption public key (RSA based at this time) that is supplied by the card. The overall process for offline PIN transmission is illustrated in the picture above on the right, and we deliberately provided these two images side by side so that it was easier to compare and contrast the processes involved in both types of PIN verification.

There are currently five standard PIN block formats ratified in ISO9564 – formats 0, 1, 2, 3, and 4. Format 2 is only used for offline PIN use, and may be sent in plaintext or encrypted into the EMV encrypted PIN block prior to transmission to the card.
The other formats are used for online PIN, and historically format 0 is by far the most common. Most online PIN block formats contain both the PIN and part of the PAN of the card, binding these together. ISO format 1 does not contain the PAN, and because of this it is deprecated for new use.
ISO format 4 is a new online PIN block format that uses AES for encryption (the others use DES or Triple DES). Because of this, ISO format 4 is 128 bits long, as opposed to the 64 bits of the other formats.

Obviously, online PIN is more involved, requiring more parties overall to agree and handle the PIN, and generally is more complex to implement. Why have online PIN at all then? Historically PINs could only be implemented online with magnetic stripe cards (as they have no processing elements themselves). With the initial deployment of EMV cards part of the issue was the increased cost of cards that were able to process the RSA decryption of the customer PIN, which is required if you don't want to send the PIN in plaintext to the card (which ***is*** a security risk – more on this later). When EMV was first introduced, cards that could perform RSA decryption at an acceptable speed were a considerable cost uplift from non-RSA capable cards, and this was considered an impediment to the deployment of EMV. This is not so much an issue anymore, however.

There are other reasons as well, including some methods of attack against EMV which only apply to offline PIN use. Ultimately, it's often a decision made by the specific country or Issuer who's deploying their cards, based on their existing infrastructure and risk appetite. There are benefits to offline PIN, in that it reduces the vulnerability surface of the PIN use to just the terminal and the card, whereas online PIN may potentially be attacked at any of the hops from the terminal to the Issuer. However, it's often hard to avoid online PIN completely as ATM transactions never use offline PIN.

Generally offline PINs are more common in areas and countries where EMV was deployed early, or where communications infrastructure is not totally reliable. It is important to understand though that offline PIN and offline transactions are two separate things. An offline transaction, where the transaction is approved by the terminal itself and later batched for upload to the payment network is a separate thing to offline PIN. You can in fact have online PINs used in a batched offline transaction, where the terminal stores the encrypted PIN for later transmission (but this is not common), and of course you can have online PINs used in both offline and online transactions.

In Annex A of this book we provide specific formats of both online and offline PIN, and later on we will discuss the threat models in more detail, but for now it's enough to know that they both exist, are in common use, and do not in any way reflect if the transaction is processed online or not.

Terminal Velocity

Given the above information about payment cards and how they work, it's perhaps not surprising that the main purpose of a payment terminal – its raison d'être – is to accept payment cards and customer PINs. If these details are to be provided on a magnetic stripe, or on a physical chip, then of course the merchant must have a device that is able to receive information in these formats.

However, the ways in which we pay are changing and with these changes we're also seeing changes in how we're providing the payment information to the merchant. Contactless and QR based payments allow for the use of 'off the shelf' hardware to accept payments, and increasing options for customer authentication is challenging the use of PINs. With these changes, the role of the payment terminal is changing as are many other aspects of payment.

Section Take-Aways:

1) The Primary Account Number is used to route transactions to your Issuing bank.
2) Magnetic stripes can be easily copied.
3) The CVV/CVC printed on your card is different from the CVV2/CVC2 value on the magnetic stripe of your card.
4) There are multiple types of cardholder verification methods (CVMs).
5) Offline PIN may be used with online transactions.
6) ATMs only ever use online PINs.
7) Online PINs are 'translated' with different encryption keys at each hop of their journey from the payment terminal to your Issuing bank.
8) Customer PINs must only ever appear in ISO format encrypted PIN blocks outside of a Secure Cryptographic Device.

Much of the card based payment landscape around the world are based on the standards published by EMVCo, and this is increasing as the areas which have lagged behind on EMV deployment are brought up to speed. EMV is often referred to as 'chip and PIN', which was actually a marketing term used by the UK banks during their roll out of this technology and does not cover the breadth of what EMV can look like around the globe. More and more often, many EMV transactions do not actually involve a PIN, and the delivery of the EMV data can be over many different channels – some of which use a card with a chip on them, and some of which do not.

Touch Me, Feel Me, See Me, Need Me

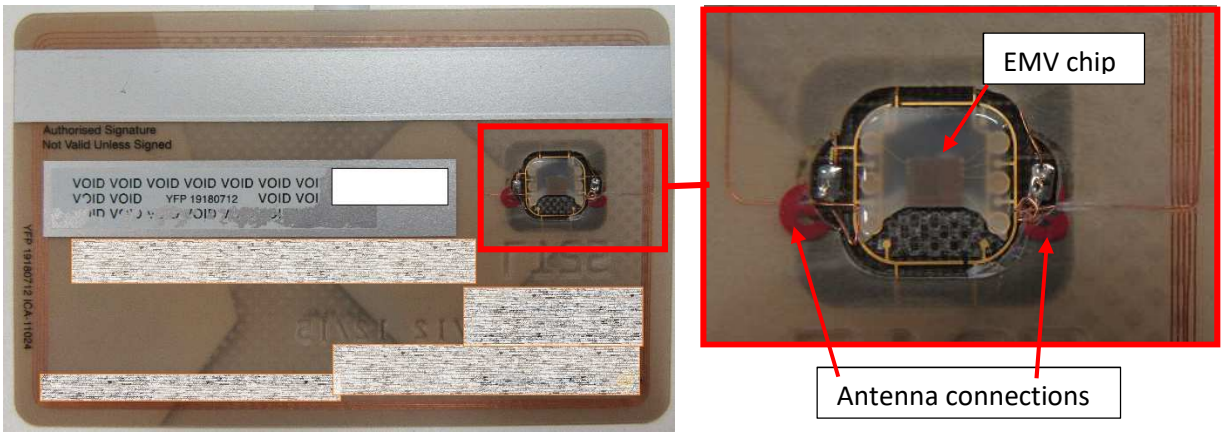
The most commonly understood formfactor for an EMV transaction is the 'contact' card – the traditional ISO7816 compliant payment card format with an internal 'chip' that has contacts in the upper left quadrant. In this format, communications from the card to the payment terminal occur over the physical electrical interface of those contacts. The terminal has a 'card acceptor' in the slot into which you insert your EMV card which provides physical connections to these contacts and allows for the data to be communicated – this is essentially like the SIM connector on your phone, but with a slot to accept the ISO7816 physical format of your payment card.

These cards may also, as an option or as a feature that replaces the contact interface, provide a 'contactless' communications path. Here, the card has an internal antenna which is also connected to the chip and this antenna both allows for the card to be powered by, and communicate through, a radio frequency field that is generated by the payment terminal. When you bring the contactless card into the 'contactless volume' of the terminal – which extends approximately 5cm above the EMV contactless symbol on the terminal – the card antenna picks up the radio frequency signals and the chip is able to convert this into sufficient power to then perform the EMV operations required.

It's a common error to conflate contactless cards with NFC and RFID, where really they're all different things. Most contactless payment cards comply with ISO14443 specifications for communications and interface (as either type A or type B cards), whereas RFID is defined in ISO15693. RFID cards can generally be thought of as 'contactless barcodes' – they have no real processing capabilities and can only convey a short numeric identifier (usually 40 bits). Whereas ISO14443 cards are actually able to perform calculations and processing when powered by the reader field.

NFC is defined in ISO18092 and is largely similar to ISO14443, but allows for operation as either a passive or active participant in the communications.

One aspect of the way that contactless cards work is that the 'tap' moniker that is used is a bit of misnomer – it is often understood that the movement of the card towards and away from the terminal is required to drive the power to the card, but this is not the case. However, the terminal drives the RF field itself, so you only need to bring the card into the contactless volume to have it power the card and for the transaction to be processed. In fact, 'tapping' the card quickly can be a reason that a contactless transaction fails, as the card may be taken out of the contactless volume provided by the terminal too quickly for the payment to be fully processed.



A transparent EMV contact / contactless card, showing internal chip and antenna

The physical and electrical interfaces described above are primarily tested during EMV Level 1 testing, but it's important to note that these 'traditional' formfactors for EMV payments are not the only ones that exist. QR code based methods are also specified for transaction process with barcode scanners or cameras, and of course digitization of payment card data onto mobile phones has become common place over the last few years. We will expand on how some of these new methods of processing EMV transactions occur later in this book.

EMV - General Transaction Flow

Although the electrical interface for contact and contactless cards differs, the actual data packet formats are largely the same, utilizing the ISO7816 Application Protocol Data Unit (APDU) format. If we take an analogy from network protocols such as TCP/IP, the APDU is the 'IP' packet of the EMV world, carrying information for larger data structures on the cards referred to as 'tags' which are encoded in Tag Length Value (TLV) format. It's these 'tags' which are of most interest to those interested in the EMV process, carrying the actual data from the card to the terminal.

It's important to note that this data transfer from the payment card is not encrypted in any way (at least in 'current generation' EMV – we'll discuss the potential for transport layer encryption within EMV 2nd Gen later in this document). For now, with currently deployed EMV systems, APDU data is transferred between the card and terminal in plaintext, regardless of if the transaction is based on contact or contactless EMV. There are many security features that EMV provides, mostly in the form of authentication of both the cardholder and the card itself, but datagram level encryption is not one of these.

Individual components of EMV transaction processing is validated during EMV level 2 testing, resulting in a compliant 'EMV kernel' that is listed on the EMVCo website. The correct flow of a transaction and the use of the data from a card may also be tested brand specific testing of the application interface, or with the newly discussed EMV Level 3 testing scheme.

A generic flow of the standard EMV transaction process is provided on the next page.

Initial card engagement	Card is powered on, and provides an 'Answer to Reset' (ATR)
Application selection	The terminal determines if there is a preferred application on the card that it supports, selecting the Application Identifier (AID) for that application.
Read Application Data	The terminal reads through the data on the card, fetching the tags that are stored as required by the application type and the directory structure on the card.
Card Authentication	The terminal and card agree on a card authentication method, which is one of SDA, DDA, CDA, (or no card data authentication). See the 'EMV security features' section for more details on what these are.
Processing Restrictions	The terminal determines if the card has any processing restrictions that prevent the transaction moving forward (such as geographic limits, incompatible version numbers, etc)
Cardholder Verification	The terminal compares the list of CVM methods it supports against its own list of supported CVMs, and the type of transaction being performed. Based on this, the terminal selects a suitable CVM for the transaction, or must reject the transaction if there is no match between the card and terminal (e.g. the card requires a PIN but the terminal does not support PIN).
Terminal Risk Management	The terminal uses values it has been provided to determine processing options for the transaction – for example, depending on the amount and type of card a terminal may process a transaction offline or it may go for online processing.
Terminal Action Analysis	Here the terminal selects a proposed action for the transaction as dictated by the risk management decision made in the above step.
Card Action Analysis **1st Generate AC**	The card also selects an action for the transaction, based on its own risk analysis and the input from the terminal. Some recommended actions by the terminal may be overruled by the card, and some cannot. As an output from this step, the card provides an Authorisation Request Cryptogram (ARQC) which is a form of Application Cryptogram (AC) generated by the card.
Perform Transaction (online/offline)	The transaction is performed, either online with a connection to the host, or offline, as determined by the result of the action and risk analysis steps above.
Online Secondary Card Action Analysis **2nd Generate AC**	For online transactions (only), the Issuer returns a response cryptogram that is validated by the card so that the card can authenticate the approval or denial message for the transaction.

If You Tap Me, Do I Not Pay?

So far, throughout this document, we've referred to the thing that the customer uses during payment as a 'card'. This is actually rapidly becoming an outdated term, as increasingly people are using their mobile phones to make payments, or are using 'wearables' such as rings, bracelets, watches, and even glasses to store their payment details so that they can be used to make payments. These alternative payment mechanisms often store card 'tokens' which are used to reference the primary account number for the customer (sometimes referred to as the 'funding' PAN in this case), and may use a

Secure Element (SE) in the same way that your ordinary EMV based payment card does, or may use alternative methods to store and secure payment data.

We'll discuss the specifics of how Secure Elements are tested later, as well as how mobile payments work and the types of security that are often used in these implementations. For now, it's important to understand that a 'payment card' may in fact not be a card at all, and that really does not affect the customer interaction with the payment terminal, or the payment process.

The ability to provide entirely new formfactors for payment mechanisms is really a function of contactless payments removing the need for a specific physical interface into which a 'card' must be inserted. Without the need for the card to be shaped and used in a specific way, payments have been opened up to new ideas and options.

EMV Security Features

EMV payments have a number of features that make them significantly more secure than magnetic stripe payments. These features include methods for the terminal to authenticate the customer payment mechanism, for the Issuer to confirm this payment mechanism was involved in the transaction, to protect the customer PIN during transmission and verification (in offline PIN transactions), as well as many others. New security features have been added to EMV over the years to address concerns and research over the years which have highlighted areas of improvement, and it is certainly correct to say that EMV transactions are not perfectly secure (no transaction can be). However, it is also equally true to say that EMV transactions are significantly more secure than magnetic stripe transactions, and this can be backed up by statistics and research into the levels of fraud before and after EMV deployment in any country that has gone through that process.

Offline Data Authentication (ODA)

Often the first security measure that is discussed as part of EMV security is the ability for the payment terminal itself to authenticate the customer card during the transaction, through a process known as 'Offline Data Authentication'. The term 'offline' here means the process is performed between the terminal and the card, without any interaction with the backend payment hosts or network, and does not have any interaction or implications regarding if the transaction will be an offline transaction, or if a PIN would be processed offline. The payment industry is great at overloading terms, and 'offline' is pretty much the worst case of this. Sorry.

The overall goal of the ODA process is for the terminal to use public keys it has been loaded with from each of the payment brands it can work with, to cryptographically authenticate data sent to it by the card. There are three main methods of how this is achieved; Simple Data Authentication (SDA), Dynamic Data Authentication (DDA), and Combined Data Authentication (CDA). A brief summary of each of these is provided below.

SDA is a process where a set of predefined data on the card has been signed using a private key held by the card Issuer before the card was created. This ensures that the terminal can verify that the data coming from the card has not been modified since the card was made. Because SDA does not require the card to do anything except pass this data to the terminal, it allows for use on cards which are not capable of performing public key cryptographic operations – but it also does not really validate the card itself, only the data from the card. It is still possible to copy this data from an SDA card to another card

and have the SDA process pass OK (although other EMV security measures will allow for detection of such a clone card later in the transaction flow).

DDA is a step up from this, implementing a challenge/response interaction between the card and terminal such that the terminal supplies some data for the card to include in the signature operation (along with the actual data on the card that is required to be authenticated). This ensures that a 'cloned' card can be detected by the terminal during the transaction, but it does require that the card is able to perform public key cryptographic operations. As noted, this is less of an issue now, and DDA is considered the minimum level of security to be implemented for ODA.

CDA was designed to prevent certain attacks against the EMV transaction process. A problem with both SDA and DDA is that the authentication is performed on the card data, separate from the actual transaction process. CDA alters this by including data from the transaction performed to prevent attacks where the data exchange between the card and terminal is altered (this can allow for altering of the CVM used in a transaction, e.g. so that the card thinks the transaction was authorized with a signature, but the terminal thinks it was authorized with a PIN – more on attacks on EMV later).

It's important to understand that Offline Data Authentication is not a mandatory step in the EMV process, and the application or use of ODA will depend on the card and terminal used, and how they are configured.

Application Cryptogram

The Application Cryptogram is essentially a Triple DES Message Authentication Code (MAC) that is calculated by the card across a selection of data from the transaction. As part of an EMV transaction the card generates an Authorisation Request Cryptogram (ARQC) which allows the Issuer to cryptographically authenticate the transaction process and validate that the actual customer card was involved in the transaction. Even if you can copy all of the data on the card, EMV cards are configured and used to prevent the extraction of the AC key, as well as other sensitive assets. This is the primary way in which an Issuer can determine if a card used in the wild is real or cloned, and so it's vital that the AC is correctly validated during online transactions, and checked when offline transaction data is provided in a batch. Unfortunately this does not always happen, and when you hear of EMV cards being cloned, it is often because the Issuer is not correctly validating their cryptograms (or the transaction is being performed entirely offline without ODA).

Similarly, during a (standard EMV) online transaction, the Issuer returns an Authorisation Response Cryptogram (ARPC) which the card can validate to ensure that the Issuer acceptance or decline of the transaction has not been tampered with. This step is actually removed in some forms of EMV processing, to speed up the transaction, as we discuss next.

Card Based CVM Verification

It has been noted that the terminal is able to send customer PINs directly to the card for authentication, either as plaintext or encrypted with a key from the card. For encrypted offline PINs, the terminal is able to authenticate the public key sent to it from the card through signatures that chain up to the payment brand root certificate which is loaded into the terminal. Details on the exact format of the EMV encrypted offline PIN format is provided in Annex XX at the end of this book.

It's also possible for the card, or another payment mechanism used by the customer, to provide the authentication through biometrics, a PIN, or other features which can be used by that payment mechanism itself to validate the customer is present and authorizes the transaction.

For 'normal' payment cards this is done through the integration of a biometric sensor on the card itself, which is held by the customer (usually with a thumb) as they present the card to the payment terminal. Such cards may be either contact or contactless, and the biometric data is not sent to the terminal for validation – the validation occurs on the card itself, and then the card returns a "no CVM" requirement for the transaction if the biometric matches (as the card has performed the CVM matching itself).

Similarly, you can have what is referred to as 'On Device CVM' (ODCVM) or 'Consumer Device CVM' (CDCVM), where a consumer device that is not a card (a phone, a wearable, etc) provides the authentication. This can be done through biometrics, such as through the fingerprint or facial recognition sensor on a phone, or can be

The Tortoise and the Hare

In some regions of the world, there has been pushback against the EMV rollout due primarily to the increased time taken for an EMV transaction. Anyone who has been in the US during the first 18 months of EMV migration can attest to the "Please Swipe" stickers that are sometimes found adorning the EMV card slots on terminals. Part of this is a reasonable objection to the difference between EMV and magnetic stripe transactions – a magstripe card can be read in less than a second at any time prior to, during, or after the POS transaction; but a contact EMV payment card requires data from the terminal about the transaction, and can't be removed before the transaction is complete to ensure any Issuer cryptograms are authenticated.

However, other aspects depend on how EMV has been implemented. During the initial stages of EMV roll out in the US, UL performed a study of various merchants and found widely varying results for transaction timing, from 4.4 seconds through to 22.6 seconds. This variance had two major components; the actual terminal / card interaction varied between 3.7 seconds to 17.3 seconds, and the online authorization process varied from 1.2 seconds to 18.9 seconds.

In an attempt to reduce this timing, and improve the acceptance of contact EMV in timing sensitive markets, a new form of 'quick' EMV (under the moniker of Quick Chip, M/Chip fast, amongst others) was introduced. This basically optimized some of the terminal and card interaction, and importantly removed the need for the card to authenticate the Issuer cryptogram. This means the card can be inserted and removed prior to the finalization of the transaction, speeding up processing and the apparent 'wait time' for the customer.

It should be noted that contactless cards and terminals have a maximum processing time requirement during their testing (<500ms), and therefore are generally quite comparable to the speed of magnetic stripe transactions.

Magstripe in Chips Clothing

Despite the clear value that EMV brings, not all chip cards are EMV cards. In areas where EMV deployment is just starting, or where acceptance on terminals is low, payment cards which provide magnetic stripe based data across a contactless (or contact, but mainly contactless) interface are possible. These types of cards allow for the chip based transaction to be processed in exactly the same

way as a magnetic stripe transaction, as the card simply delivers track equivalent data across the connection.

Of course, removing the various layers of security that EMV provides has the effect of … removing various layers of security. There's certainly a place for magstripe profile based transactions, but if you're looking to head down this route you do need to consider that it changes your threat landscape significantly. So what security do these types of systems have? Generally they will implement what is referred to as 'dynamic CVV/CVC', which is yet another form of Card Verification Code that further dislocates a contactless payment, from a magnetic stripe payment, from a card not present payment. Dynamic CVV/CVC values are generated on the card uniquely for each transaction, generally using some data provided during the transaction from the terminal, which means they are harder to copy to make cloned cards (although brute forcing of the values is sometimes possible).

A Payment by Any Other Name

Because payment cards are so universal, they have utility for operations that stray from the normal payment process. One such example of this is the use of contactless payments in transit, where the payment card becomes both the payment mechanism, as well as the identifier for the transit customer as they enter and leave the transport system. This is often referred to as 'pay-at-gate', which is can be a bit of a misnomer as the payment is often not actually performed at the gate itself.

Common pay at gate systems usually read the contactless card to obtain the customer PAN, and then use that (or a hash of this) as the identifier as the customer travels through the transit network. This data is collected and transmitted to a backend system known as a 'fare calculation engine' that collates the travel data and then bills the customer as required. This allows for accommodation of billing for different zones of travel, travel cost caps per day/week/etc, and calculation of fares based on distance or time of travel.

One issue with this, of course, is that if you are only collecting the PAN to perform the payment later, you may not have enough details to meet the authentication requirements for a card not present transaction. You also leave an opening for people to clone cards if the EMV authentication process is not used. For this reason, it is also common for a small or zero sum payment to be processed when a new card is used in the transit network, and it remains best practice to ensure that ODA authentication of the card is performed where the timing requirements of the network allow for this.

Mind the (App) Gap

In the diagram of the four corner model presented at the start of this section, you may have noted that we called out the POS App as a separate thing. This is because the applications that reside on a payment terminal are often out of scope of the existing payment security standards. These apps will be validated in terms of 'end to end' functionality, and to ensure that they provide the user experience that the POS vendor, merchant, or institution shipping the terminals wants; but they will generally not be tested for security.

This is an important gap that exists in the payment eco system, and one that is already closed in some of the over-arching solution standards that exist, such as PCI Point to Point Encryption and PCI Software PIN on COTS. Any institution or vendor shipping 3rd party apps with their terminals should consider the risk posed by this, if that fits in their risk appetite, and how any residual risk may be mitigated.

Risks posed by applications that run on terminals can take various forms. As we'll discuss later in this book, when a payment terminal is assessed under the PCI PTS requirements, many aspects are examined including the key management and protocols that are used for securing payment transactions. However, as many banks implement their own types of key management and even have different 'flavours' of standard protocols such as ISO8583, there can be a need to make changes to the way in which the terminal handles keys and PINs for each bank.

It's tempting to make these changes in the application, with an assumption that the terminal firmware does not change and therefore the PCI PTS approval is not affected. However, this is a false assumption as implementation of key management or changing of security protocols assessed under the PCI PTS standard actually violate the existing approval.

In this way, implementing security functions in a terminal application can be a compliance problem.

There is also the security aspect of the applications that execute on the terminal – if they have poorly written code, or are incorrectly using the security functions provided by the terminal firmware and hardware, vulnerabilities may be exposed that could lead to compromise of the terminal and payment process.

Because of this, deployment of applications onto terminals should be carefully considered and managed.

Section Take-Aways:

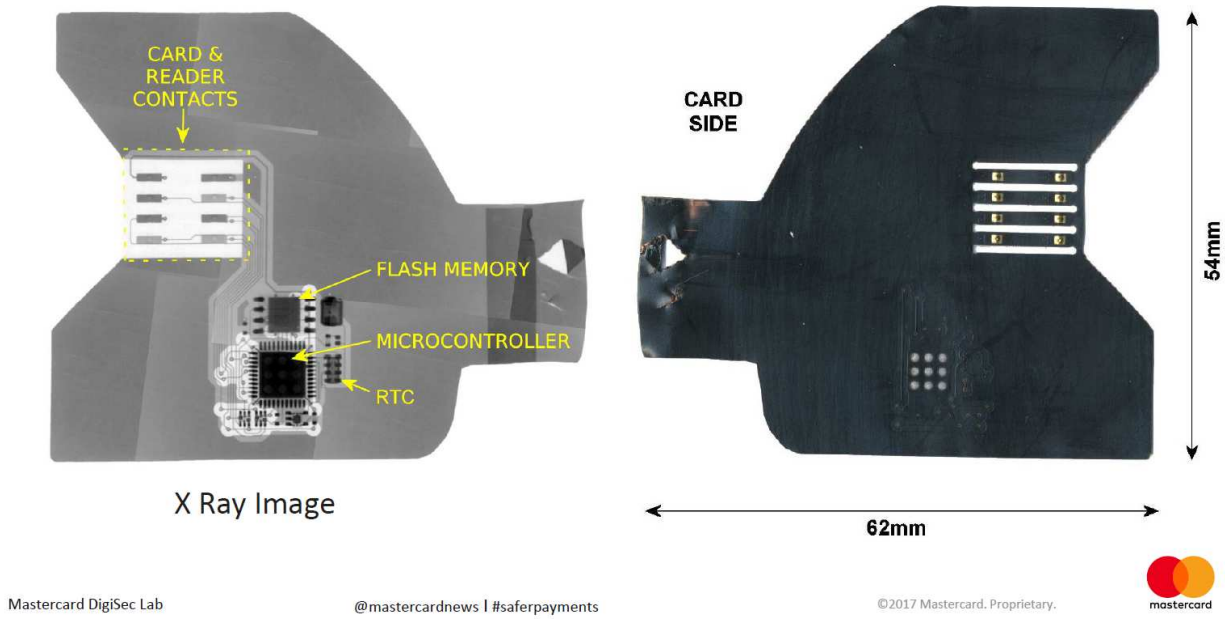
1) EMV based chip cards significantly more secure than magstripe for card present transactions.
2) Not all chip cards are EMV cards.
3) Validation of implementation and use of EMV security measures is important.
4) Contact and contactless EMV both transmit data between the card and terminal in the clear.
5) Payment terminals can authenticate the EMV card, but the EMV card cannot authenticate the payment terminal.
6) Offline PIN may be used with online transactions.
7) The speed of an EMV transaction has many different factors, but contactless transactions are generally quicker than contact transactions.
8) The boundaries between card present and card not present transactions are blurring, with systems like pay at gate using a card present at the time of travel to authenticate a later card not present transaction.
9) Terminal applications often fall between the gaps in compliance programs, and should be carefully managed to ensure that new vulnerabilities are not exposed.

Understanding and assessing your threat landscape is a very important part of deploying or interacting with any complex system, and payments are no different.

Skimming and Shimming

One of the most widely known attacks on payment systems is the 'skimmer' – an additional magnetic stripe card reader that is placed in the path of the card as it is normally used, such that this additional bug collects the cards track data for later use. These devices may appear connected to the slots of ATMs, as replacement or overlay components for payment terminals or ATM facias, or as a separate devices that are used through sleight of hand by an attendant. Skimming is hard to prevent as it is focused on collection and storage of data that is designed to be easily readable from the card, but where the (magnetic stripe) card has no active elements to uniquely link that data to one particular customer card.

Skimmers are often coupled with cameras or keypad overlays to capture the customer PIN along with the cards track data, so that the collected data can more easily be monetized at an ATM or for higher value purchases.



EMV Card Shimmer
(Image curtesy of Alan Mushing, Head of the Mastercard DigiSec Lab)

A 'shimmer' is essentially a skimmer, but for contact chip cards rather than magnetic stripe cards (see above image curtesy of Alan Mushing, Head of the Mastercard DigiSec Lab (where Mastercard undertakes forensic analysis of devices in support of the Industry – they've got all sorts of very cool tech there for forensic analysis, including X-Ray and Tomographic imaging machines!)). Shimmers are often made from flexible printed circuit onto which controlling circuitry is soldered, and the entire thing is then placed *inside* the EMV card slot of the payment device.

Having read the previous sections of this book, you're probably wondering how a Shimmer is useful given the security mechanisms implemented in EMV – well, they're broadly used in two cases.

1) For a PIN capture attack, where the magnetic data from the card is captured as well or the card is stolen afterwards from the customer. This only works if the CVM used is plaintext PIN.
2) For a card data capture attack, to produce cloned EMV cards.

Wait! How is (2) above possible, you ask, given all the EMV security measures we've discussed previously? Unfortunately, there are some instances where ODA is not used (remember this is a decision made between the terminal and the card based on their combined risk profiles), or SDA is still used, or where the transaction is processed offline, or where the Issuer is not correctly checking the Application Cryptogram values sent from the cards. In these instances, it can be possible to 'clone' enough elements of an EMV card so that the clone can be used for payments.

In this case the card is not a *perfect* clone, it does not have the secret or private keys used for DDA/CDA authentication or to produce the Application Cryptograms correctly, but in instances where these are not used or not checked, this does not matter. For this reason alone, if you're deploying EMV systems, it is ***strongly*** recommend that you have someone check that you are correctly implementing the security features that EMV provides.

Blackbox Attacks

Blackbox attacks are mostly associated with ATMs, but can be performed on any system really as they are best summarized as an attack on the internal communications of a complex system, using a 'blackbox' of electronic components. Of course, there may not be an actual box, or it may not be actually black, but there must be some form of electronics inserted into the target complex system, designed to capture or manipulate the communications between these components.

In an ATM, this often means 'sniffing' for the card data across the internal USB bus, or directly interfacing to the cash draw to have it output cash. Other attacks are possible as well, including placing the ATM Encrypting PIN Pad (EPP) that is used for customer data entry into plaintext mode when it should instead be outputting an encrypted PIN.

Physical Attacks on Card Acceptance and PIN Entry Devices

As well as ATMs, attacks on the 'brick and mortar store' based payment terminals which are used to accept your card and PIN are not only possible, but known to happen. Here, the devices are usually taken from the store, attacked so that the sensors that detect removal of the casing or access to the keypad are disabled, and then a bug is placed into the device to capture the card data and customer PIN.

These attacks can be quite complicated due to the security that is built into devices that are compliant to the PCI PTS requirements (more on this later), and so often this type of attack is a 'last resort' when other attacks are not possible. Where card data is exposed in the clear outside of the payment terminal itself, attacks on the Point of Sale system are more common.

Memory Scraping and Keyboard Logging

Point of Sale (POS) systems often connect to an external card reader or terminal for payments, and if that device does not automatically encrypt the card data as it is entered, then the data is exposed in the POS system in plaintext. Sometimes the POS may then encrypt the data itself, but the exposure of the

data in the POS still means that it may be captured at that point by criminals. This can be done through logging or 'hooking' of the interface to the terminal/card reader, through 'memory scraping' where a program looks through the memory of the POS to find the familiar patterns of card data, or through subversion of the encryption process itself.

Logical Attacks on PIN Entry Devices

Although payment terminals are assessed for logical security as well as physical security, this does not 100% guarantee that they are free from software vulnerabilities. New vulnerabilities are discovered all the time, and PCI PTS approvals can (currently) be valid for up to 9 years – so it is likely that logical flaws will be found in any given terminal during that time. Ensuring that your terminal provider has a history of vulnerability management is important to mitigate against these threats.

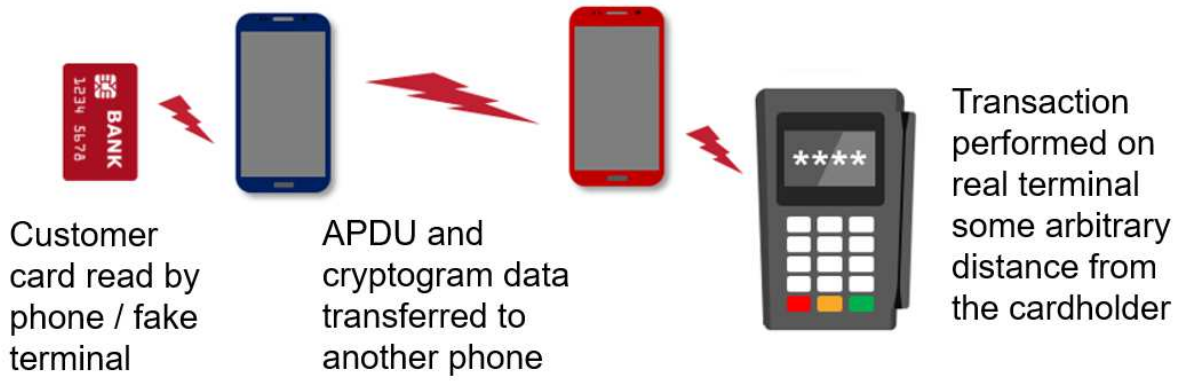
There is also the 'app gap' which was discussed previously; most new payment terminals allow for 3rd party applications to be executed on them, and these can introduce their own vulnerabilities to the system.

Recent versions of PCI PTS have helped to mitigate the deployment of logical attacks against terminals through two specific measures – the requirement that terminals must cryptographically authenticate all code that is executed on the terminal, and the requirement that the terminal performs a memory reset (essentially a reboot) every 24 to 48 hours. The authentication requirement prevents the deployment of persistent malware to the terminal, and the reboot mitigates any memory resident malware.

Relay, Replay, Preplay

EMV provides multiple layers of security to the historical magnetic stripe based payments, but one thing it does not currently provide is 'distance bounding'. That is, there is no function that ensures that the customer card (that is sending the data objects and performing the cryptogram calculations) is actually the same card that is physically placed into the terminal card slot or placed inside the contactless volume. This opens up the possibility for an attack called a 'relay' where a criminal may present a fake payment mechanism to the payment terminal, and an accomplice may fraudulently connect to another customers card to send the data from that customer card to the merchant terminal.

An illustration for this type of attack is provided below, where the attackers accomplice uses the blue phone to interface to the customers card, and the attacker themselves use the red phone to interact with the terminal. The actual card data sent is from the customer card on the far left.



Components of a contactless relay attack

This type of attack depends on attacker having clear access to the customer card – and as we'll detail later normal phones are not great at picking up contactless card details at a distance.

Replay and preplay attacks are different, as they don't need access to the card during the transaction itself. Instead, the data is collected prior and then used with a real terminal at some later time. Capture and use of magnetic card track data is essentially a form of replay attack; the data on the tracks are captured, and then 'cloned' to another card for replay as required. EMV is generally immune to such attacks through use of the cryptograms, but preplay attacks have been found to be possible if the terminal does not provide sufficient entropy in random numbers (Unpredictable Numbers, or UNs) which are required as part of the protocol. The attack referenced above for brute forcing the dynamic CVC/CVV values on magstripe-mode chip transactions is also an example of a preplay attack.

Your Number is Up

Although it's a requirement that PINs are only ever exposed outside of an SCD in encrypted format (well, except for with a SPoC solution … more later) this does not mean PINs are totally secure. Attacks may be leveled at PINs even when they are encrypted through a number of mechanisms, usually involving creating a 'PIN dictionary'. This can be done with a device that has a fixed (or slowly changing) cryptographic key to encrypt PINs, where a criminal can capture the encrypted PIN block and card data, and then re-do transactions with every possible PIN until the two encrypted PIN blocks match – indicating that the PIN the criminal has entered is the same as the PIN that the customer entered.

This is only possible with certain PIN block formats that do not contain random data for each transaction (such as format 3 and format 4), and is made more complex by the fact that the most common PIN block formats (such as format 0) contain the customer PAN, and so two customers with the same PIN but different PANs will have different encrypted PIN blocks, even if the same key is used.

However, format 1 PIN blocks do not contain the PAN, and with this format there is another attack where criminals can translate the PIN from encryption under a PIN block which contains the customer PAN, to format 1 which does not contain the customer PAN. This allows for the criminal to construct a PIN dictionary which is disconnected from customer PANs, thereby reducing effort to brute force PINs. This type of attack does require access to the HSM to perform translation functions, of course, and if you have criminals with that level of access you probably have other issues to deal with. But it is the reason that format 1 PIN blocks are deprecated, and have specific translation rules that must be tested in PCI HSM evaluations.

Data at Rest, and Objects in Motion

In many of the above scenarios we've discussed the bypass of inherent security controls implemented in the payment process. Sometimes these controls are not even present, making attacks easier. Card data should be encrypted when stored or transmitted, but this may not be implemented correctly or may not be implemented at all. There has been a general trend towards encryption of payment transactions as they are sent to the payment host, and much of this started in certain geographies specifically to counter extant attacks on that communication. Old-fashioned terminals that used modems over dial-up connections were sometimes the subject of attack, having the phone line literally tapped with an audio

recorder (MP3 recorders were used as technology advanced!), and this type of attack was only thwarted through use of encryption of the transmissions.

It may seem common now to find reports about databases with large amounts of card data being stolen, but the use of EMV, encryption, tokenization, etc are helping to reduce this data theft, or reduce its ability to be monetized even if it is stolen.

Reduce, Secure, Devalue

Given all of the above – the details on how payments work, and the types of attacks that may be applied – we can finally restate that payment standards exist to do two things:

1) Improve the chance that any payment made by a genuine card holder with a genuine merchant is processed correctly.
2) Reduce the chance that an attacker may subvert or hinder that payment process, steal or falsely use customer payment details, or extract value from the merchant or customer through that process in a way they have not authorized.

The many different stakeholders in the payment industry all play a part in ensuring those two requirements above are maintained, and each standard has a different role or aspect in achieving that goal.

Everything that is Old Shall be New Again

As the security of card present transactions has increased over the years, forms of fraud that were previously 'going out of fashion' have been making a resurgence. This includes things like enrollment fraud and lost/stolen fraud. Neither of these types of fraud are particularly new, but if you can't clone EMV cards, then either obtaining them fraudulently or stealing them becomes a viable option. There are new twists to the fraud types in the current payment environment, however.

Enrolment fraud may be committed in the same way it has always been, through exploiting stolen customer details and issues in the Issuers Know Your Customer (KYC) checks. However, the increased use of mobile payments and digitization of payments has opened new avenues for this fraud. As an Issuing institution it is important you have the ability to validate your customer digitzation process for mobile payments – do you know their mobile phone number? Do you require them to perform the digitization through their banking app, and validate the card to their banking details? If not, what else do you do to ensure that the process is indeed being performed by the real cardholder?

Digitization also opens up new avenues for account take-over fraud, where an existing account is ported to a different phone or app. This also relates to issues in KYC checks, and speaks to the need for something more than SMS or email messages to be used as the second factor in such authorization processes. The majority of mobile payment fraud is based on vulnerabilities in the digitization process, so securing this process (whilst still providing a positive user experience!) is both difficult but vital.

Lost and stolen fraud is generally performed the same way, but the avenues for use have opened up more – with the opportunity for no-CVM transactions, a stolen card becomes potentially more attractive as a crime of opportunity. In the past it was common for EMV cards to have internal counters that required the card to be inserted or otherwise authorized after a number of contactless or no-CVM transactions. This became less common with the wider proliferation of contactless transactions and

contactless only terminals, but can be expected to have a resurgence with the European PSD2 regulation requiring strong authentication after a specific number (5) or cumulative amount (€150) of no-CVM transactions.

Watch your Backend

A final example of fraud that also appears to be increasing is backend fraud. This is where the criminals gain access to the Issuer or Acquirer systems, not the merchant systems, to defraud the actual payment process itself. This is another area where the security of EMV can be seen to be driving fraud into different areas, and if you can't clone or compromise the cards themselves, maybe you can attack the systems and processes that validate the cards or transactions.

Examples of this type of fraud include global attacks on ATM networks, to allow for extraction of many millions of dollars from ATMs without proper validation on the backend. As more and more security is baked into the global payment system, and as open banking becomes more of an accepted thing, expect backend attacks to increase more.

The Future of Fraud

We've taken great pains throughout this document to highlight how payments are changing, and it's reasonable to expect that this will lead to changes in fraud as well. We're already seeing that in the case of the up-tick in lost and stolen fraud, but what brand new types of fraud will we encounter into the future? Will biometric fraud become a thing, with people lifting fingerprints from online photos, or recently used ATM machines? Will lost and stolen fraud extend to mobile devices, or will fraud against QR payments become the next big thing?

Of course, we just don't know right now. What we can be sure of is that criminals are not going to wake up tomorrow and decide that they've had a good run, but now it's time to get a real job. Fraud will always be a thing, and so our industry will always need to think one step ahead and maintain the fight against criminals around the world.

Section Take-Aways:

1) With all of the security placed in payment systems, attacks can still happen. The role of many of the standards discussed in this book is to limit the scope and impact of these attacks.
2) An EMV deployment that does not use the security features of EMV is no more secure than a magnetic stripe system.
3) Encryption of card data at the earliest possible point can help to greatly reduce threats to the payment system.
4) Different threats and attacker profiles exist in different geographies and verticals. Understanding your risk profile is vital to successful mitigation of attacks.
5) New methods of payment have opened new types of attacks, or changed the way existing attacks can be performed.
6) With all of the above, it is important to remember that EMV deployment objectively reduces card present fraud.

We've already provided some details on EMV in this document, and this section will act as an overview of the testing involved in validating EMV cards (and other customer payment mechanisms), payment terminals, and payment processes.

EMV Card Security Testing

EMV actually has a number of security testing programs, covering both the physical and functional security aspects of the chips that are used in payment cards (often referred to as Secure Elements), as well as the security of mobile payment implementations (from the Issuing side – PCI cover security of mobile payments acceptance).

EMV IC testing validates the security of the physical properties of the chip, ensuring that it is not vulnerable to physical probing or reverse engineering attacks. This requires detailed testing with physical equipment such as focused ion beams, and micro-probing stations, as well as more standard glitch and side channel equipment to validate any hardware based cryptographic accelerators.

EMV ICC and platform security evaluations confirm that the software – the application and operating system – that runs on the validated IC is secure. This involves ensuring that the code is correctly using the hardware security features of the chip, such as cryptographic accelerators, as well as confirming attacks to bypass things like incorrect PIN counters cannot be performed. For example, if the code checks an offline PIN and only sets an invalid PIN counter after finding it does not match the customer PIN for the card, the card could be attacked by resetting or glitching the card after the check, but before the incorrect PIN counter is incremented.

These tests also include ensuring that the cryptograms are calculated correctly, and the cryptographic keys cannot be extracted through glitching or side channel of the algorithm process.

EMV Levels 1, 2, and 3

EMV level 1, 2, and 3 testing – also known as EMV functional testing – is a process to validate that the various components in an EMV transaction will correctly interact and work with each other.

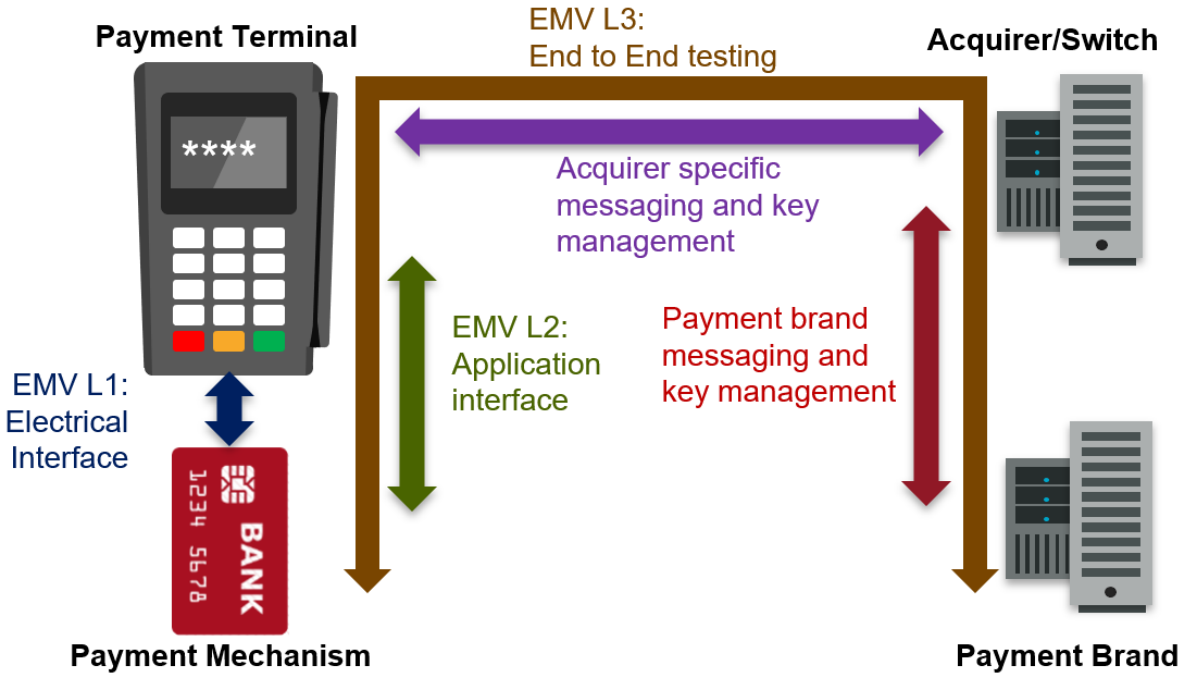
EMV Level 1 involves confirming that the physical aspects of the card, payment mechanism, or terminal are correct, as well as the electrical interface is performing correctly. This involves validation of electrical and timing properties, physical force for contact EMV and field strength for contactless. In many ways, EMV Level 1 testing can be thought of as testing of the physical and electrical layers in a network stack. The target of EMV Level 1 testing is known as an Interface Module (IFM) for contact terminals, and a Proximity Coupling Device (PCD) for contactless terminals.

EMV Level 2 concerns itself with the data that is sent across the interface validated under EMV Level 1 testing. Under this testing the individual commands, or data tags, which may be communicated between the terminal and the payment mechanism are tested. EMV Level 2 testing is often updated to include new tests to remediate issues in previous implementations – such as poor random number generation, or insufficient handling of invalid data structures.

Finally, EMV Level 3 is the testing of the end to end transaction, confirming that the actual payment process will work as expected between the card, terminal, and Issuer as expected by the EMV process. Historically this level may differ based on the geography and payment brand, but EMVCo has recently developed their own EMV Level 3 testing process. However, EMV L3 should not be confused with the

specific messaging formats and cryptographic relationships between the various parties involved in the transaction (terminal to switch/acquirer, switch/acquirer to payment brand, etc). This message format often follows some type of implementation of ISO8583, with key management that is often specific to the acquirer/switch in question (although ANSI x9.24 DUKPT is the most common 'standard' implementation globally). In the next revision of this book we'll dive into further detail on the specifics of the point to point messaging used.

A summary of the EMV functional testing levels is provided below.



Areas of focus for the various levels of EMV and messaging standards

Section Take-Aways:

1) EMV testing focuses on the security of the card or payment mechanism, as well as the correct interaction of the card and terminal, and end-to-end operation of the payment process.
2) EMV card security is separated into IC, platform, and ICC testing.
3) Payment cards/mechanisms require both security and functional testing under EMV.
4) Payment terminals require only EMV functional testing, with security testing managed under PCI SSC.
5) EMV testing is regularly updated to address new risks and functional issues.

EMV has been around for quite some time now, since the mid-1990's in a recognizable form, and along with the security issues that were noted in previous sections, there are also a plethora of updates and adjustments that have been made along the way. However, not all issues are able to be easily 'patched' out of the standard, and therefore since 2011 there has been a desire to design and specify the next generation of EMV functional and security requirements.

This is specification is called EMV 2nd Gen.

Set Specifications to Protect

EMV 2nd Gen aims to add in additional layers of security that were either not possible (or too expensive) with the technology of the 1990's when the initial EMV specifications were written, or are designed specifically to mitigate or prevent threats that were just not known at the time. From a security point of view, this new EMV specification has the following high level goals:

- Protect against eavesdropping on the communications between the EMV 2nd Gen payment mechanism and the payment terminal
- Protect against modification of the communications between the EMV 2nd Gen payment mechanism and the payment terminal
- Protect against replay or pre-play of the communications between the EMV 2nd Gen payment mechanism and the payment terminal
- Allow for detection of communications relay, through distance bounding of the payment mechanism to terminal communications
- Provide robust authentication of the payment mechanism(s) used in a transaction
- Provide security for the transport of CVM data from the terminal to the payment mechanism

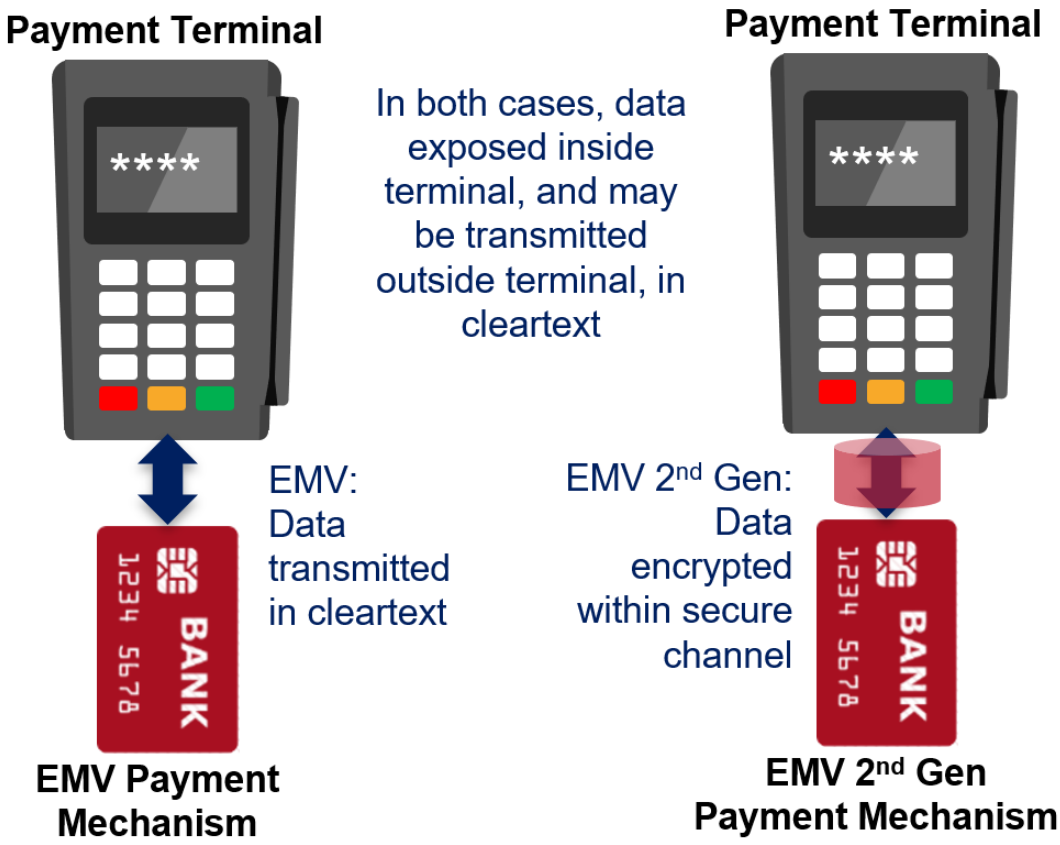
None of the above should really come as a surprise, as it covers the threat vectors we discussed earlier in this book. EMV has been very successful at reducing card present fraud, but it's folly to expect that any specification, at the time it is written, can cover all possible attacks into an indefinite future. Therefore EMV 2nd Gen is a new take on the old EMV process, in the same way we have updated TLS standards, or Wi-Fi security implementations.

Your Key to My Key, My Data to Your Data

One of the foundational security aspects of EMV 2nd Gen is the requirement to establish a secure channel between the terminal and the payment mechanism during the transaction. The term 'secure channel' is used in many different standards with many different contexts, and in EMV 2nd Gen the 'secure channel' is an encrypted transport layer between the payment mechanism and the terminal.

This encrypted link is established using the Diffie Hellman key agreement protocol, with the terminal and payment mechanism exchanging cryptographic values based on their own key pairs to agree upon a secret key that is then used to in an authenticated encryption mode to provide both confidentiality and authenticity to all further data communicated across the link.

The key pair on the customer payment mechanism is chained up to a root certificate owned by the relevant payment brand, allowing the terminal to authenticate the card prior to the establishment of the secure channel. This also prevents a wedge, shimmer, or other 'in-the-middle' system from being able to interpose in the key establishment process (as it would not have a valid key pair to provide to the terminal). The terminal keys are generated within the terminal, and as such are not authenticated.



EMV 2nd Gen creates a secure channel to protect data sent between the payment mechanism and terminal

The use of new keys during each communications session also prevents replay or pre-play attacks, as any data captured during one transaction cannot be wrongly interpreted as 'fresh' data during another session due to the change in cryptographic key providing the secure channel.

Lightspeed Communications

Protections against relay attacks is provided with the ability to provide 'distance bounding' between the payment mechanism and terminal. The term 'distance bounding' simply means that a method is applied to mathematically prove that the two devices in communications must be within a certain physical distance of each other. This sounds a bit like magic, but the fundamental process is quite simple.

We all learned in high school that light has a fixed speed, which is roughly $3x10^8$ m/s. This speed limit can therefore be used to determine how far away a thing is – if you can send a signal and expect an immediate response, the time between the transmission and receipt of the response bounds the minimum distance between the two objects. For example, if you send a signal and receive a response within approx. 33ns, you can be sure that the object you are talking to is no more than 10cm away, or 333ns would give you a distance bound of 1 meter.

In EMV 2nd Gen, the distance bounding is performed prior to the establishment of the secure channel (to remove the encryption/decryption of this channel as a potential for delay of the messages, and to allow for potential 'cloud' kernel implementations, which would introduce further delays in the

communications). Both the payment mechanism and the payment terminal generate random values, with the payment mechanism required to respond with its value as soon as the terminal value is received.

The time between the transmission of the terminal random value, to the response received by the terminal provides the distance bounding. Because the timing of this is critical, the constraints on the electrical properties of an EMVCo 2nd Gen system that implements distance bounding are quite strict, but the measurements do not have to take into account the time of the message transmission itself, just the time from the final part of the sent value to the start of the first part of the receipt value.

As the values are sent in the clear, the distance bounding calculation does not necessarily care what the values are; the terminal and payment mechanism must also then re-confirm the actual random values transmitted to each other once the secure channel is established, to validate that these were not generated by an attackers shim or wedge.

The use of distance bounding is currently an optional feature in EMV 2nd Gen, and proper support of this feature will most probably require an implementation in the hardware of the payment card interface, to ensure that the process is performed as rapidly as possible.

All Good Things … Must Not Happen?

With all of the added security and features of EMV 2nd Gen, when do we get it? Well, as the 2nd Gen requirements have been developed and established, there have been on-going changes to the current EMV standard (as noted). This, combined with the fact that writing any entirely new payment kernel is not a trivial task by any measure, has led to a recent press release from EMVCo that implies that a completely new EMV payment process may no longer be required. The relevant part of this is copied below:

> **Status**
>
> Recently, feedback from the EMVCo Board of Advisors (a geographically diverse group of different sector stakeholders in the card payments industry), has indicated that some features contained in EMV 2nd Generation are needed in the short term, and EMV 2nd Generation may not be the most appropriate approach to meeting these needs. To address this feedback, EMVCo is re-evaluating whether the original business drivers cited above are still relevant, and whether in the past seven years some infrastructure issues have already been addressed or new challenges have arisen.
>
> In the coming months EMVCo will be working with our Associates to consider whether the current draft of the EMV 2nd Generation Specification offers the most appropriate approach to support the future of consistent, interoperable and secure card payments, and what other approaches could be considered. Once the re-evaluation is complete, the approach and updated timeline will be communicated.

EMV 2nd Gen Status, as of November 2018

So, the current status of EMV 2nd Gen is … unknown. We'll update this section with any changes with the next revision of this book.

Section Take-Aways:

1) EMV 2nd Gen is a redesigned EMV system, with the goal of providing additional security and functional controls.
2) An important aspect of the increased security is the establishment of a secure channel between the payment mechanism and the payment terminal, prior to the transmission of any confidential data.
3) The terminal remains unauthenticated by the payment card, so any system or device can read and interact with an EMV 2nd Gen card.
4) Distance bounding is natively supported, but optional.
5) Although many years in development, recent press releases indicate there may be some rethink around the deployment of EMV 2nd Gen.

As noted previously, digitization of card data onto mobile phones is now a common occurrence, and there are a couple of ways in which this digitization process, and the transaction process itself (post digitization), can happen. These can best be outlined as using one of the following three processes on the phone to allow that phone to appear as a 'normal' contactless payment card:

1) Embedded Secure Element
2) Host Card Emulation (HCE)
3) Third party component

The first option is the most similar to a 'standard' EMV process, with embedded secure elements being how Apple Pay works, as well as some Android devices for specific bank or phone branded payment implementations. In this system, the phone contains an internal component which is essentially the same as the chip that is contained in your normal payment card, just with a different formfactor so that it can be installed into the phone during manufacturing. This Secure Element is connected to the phone NFC antenna, and is able to perform payment transactions in the same way once it is provisioned (more on this later).

Because these internal components are essentially identical in functionality to the chip in your card, it is possible for a phone with a physical secure element to be used for payments even when the phone is off, or has a flat battery. This functionality may not always be enabled, however, due to requirements for the customer to identify themselves on the device before the secure element is enabled.

Devices that use Host Card Emulation (HCE) – which is the majority of Android devices that use Google pay – do not contain a dedicated 'payment' chip, and instead 'emulate' the functionality and storage of that chip through software. This allows for HCE payments to be compatible with a larger range of devices, as the hardware features required are not so specific, and it also makes the provisioning process a bit more simple (as we detail later). However, it does remove the ability for the payment to be independent of the power of the phone, and it sacrifices some of the security features of the Secure Element to enable that broader compatibility.

The final option is for third party components, such as a microSD card, watch, bracelet, etc to provide the payment functionality. This is generally implemented through use of a secure element in that third party component, but can also be provided through communications with the mobile device in a form of HCE. The component usually also integrates the contactless antenna, removing the need for the phone to have one.

'Wearable' third party components sometimes take on the additional responsibility of continued authentication of the user; for example a watch with embedded payments functions may require the user to provide authentication on the phone for the first payment process, but then assume 'continued authentication' while the watch remains on the users wrist (as determined by the heart rate sensor).

The use of third party components was often considered the way to implement mobile payments as that vertical was first established, using either microSD cards or a SIM with contactless capabilities. This was seen as advantageous due to the lack of phones with embedded contactless antennas (at the time), allowing for payment functionality to be simply added to any phone. However the complexity in provisioning these systems, as well as perhaps many of them being too early to market, meant that most of these solutions did not succeed.

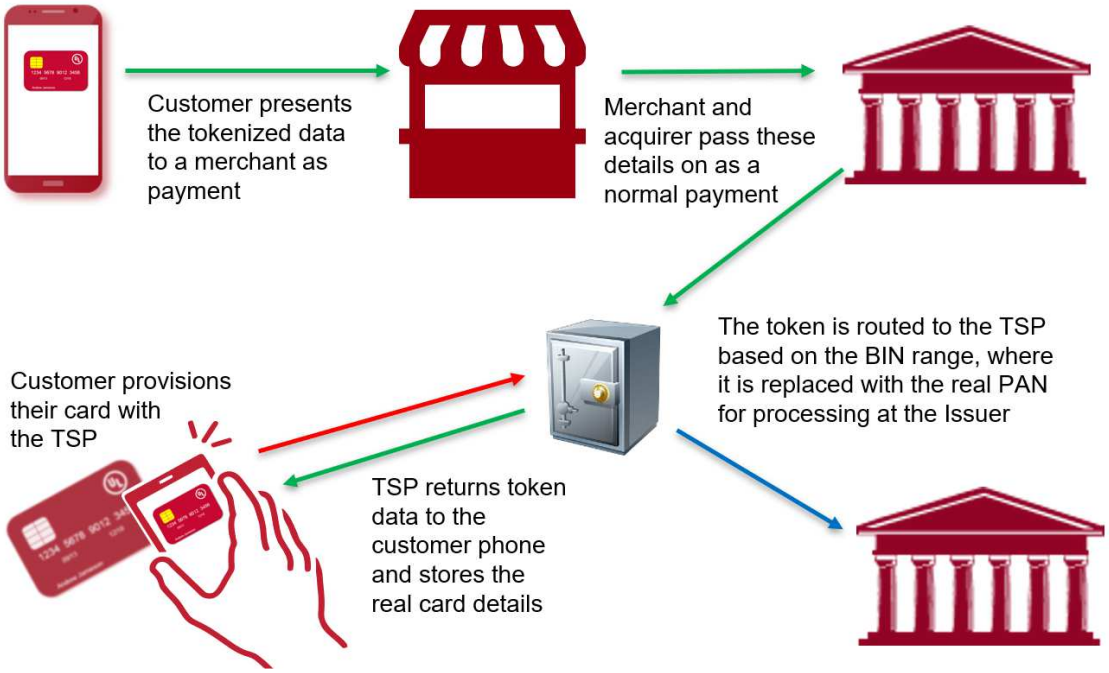
I've got Enough Provisions to Last A Week

The process for providing the payment details to the mobile payment implementation, sometimes referred to as 'provisioning', is of course essential not only to the functionality of the implementation but also to its security. This provisioning process must deliver both the 'visible' card details, such as the PAN and expiry date, as well as 'hidden' or more security sensitive details such as the cryptographic keys used for Offline Data Authentication and Application Cryptogram generation.

One of the complexities with use of embedded secure elements, SD cards, contactless SIMs, etc is that in order for these details to be securely provisioned to the device, the backend system must have the ability to use the provisioning keys stored therein. This means that these systems are either locked into use only by the manufacturer of that phone/device, or the manufacturer must provide the provisioning keys to another party. It is this complexity that caused must of the problems with deployment and uptake of the initial mobile payment offerings.

Of course, even when you own the complete hardware and software stack of the phone or tablet, you still need to provision card data and keys to that system. This is where a Token Service Provider (TSP) comes in, acting as an intermediary between the card Issuer and the mobile device. Many of the most commonly used TSPs are actually implemented and deployed by the payment brands, as a method to facilitate the wider use and acceptance of their brands. In EMV parlance, a TSP may also be known as a 'BIN Controller'.

The tokenization process can vary depending on the implementation, but generally follows the process illustrated below – the customer enrolls their card with the TSP using their phone, and here the TSP returns tokenized values to the customer for storage. These are essentially 'replacement' card values, which can be used instead of the real card values during purchases.



Mobile payment tokenization and payment flow

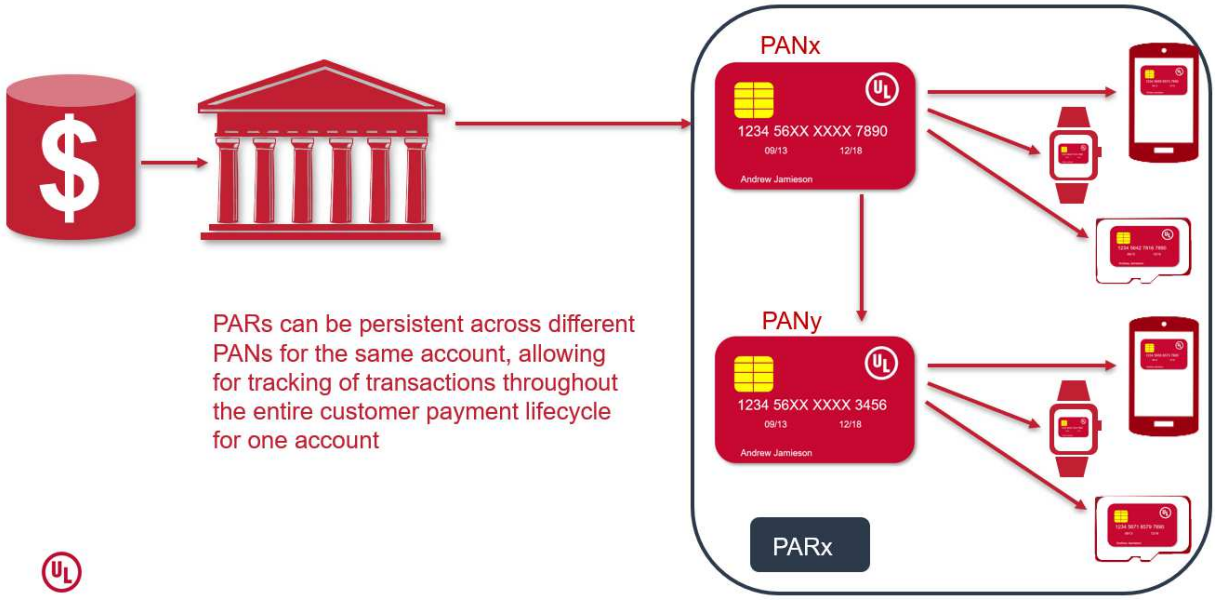
When the customer then uses their mobile payment implementation at a store, they provide these tokenized values to the merchant, who passes them along to the acquiring institution in the same way as any normal payment. The acquirer then forwards these to the TSP, to whom the transaction is routed as they are the registrant of the BIN range used on the token, and the TSP replaces the tokens with the real values for processing by the Issuer. An alternative here is that the Issuer BIN range is maintained and the Issuer contact the TSP to get the real values.

In the image on the previous page, the red line represents the only instance when 'real' card data is sent from the customer phone. At all other times the green lines represent the token values supplied by that customer instead, and the final blue line shows the 'detokenised' values sent from the TSP to the Issuer.

There are more complexities to this process. Usually it is not just the PAN that is tokenized; the cryptographic keys used in the EMV process are generated and supplied by the TSP as well. This means that the Issuer is not always able to directly validate the cryptograms generated by the card, and the TSP can act as an agent for this process as well.

PAR for the Course

If the card data is tokenized, how do you track individuals across payments? This is often desired for reasons of loyalty programs, anti-money laundering, or to help with charge-backs and other post-payment financial mechanisms. To solve this problem, EMVCo introduced the concept of a 'PAR' – a Payment Account Reference – as a separate data element to the PAN to allow for these sorts of uses. The PAR is a 29 alphanumeric value that is assigned by the BIN controller (that provides the tokens for the EMV card in question) which provides linkage to a specific payment account or PAN.



The Payment Account Reference (PAR) tracks different PANs or tokens for the same account

The BIN controller can assign the PAR in any way they want, although it is a requirement that it cannot be reversed or otherwise used to determine a PAN or token value by any other party. Only the first four

characters of the PAR must be intelligible, providing the routing information for the BIN Controller used to generate the PAR. Once assigned, the PAR remains constant between tokens applied to the same PAN, and may even remain constant between PAN values that are assigned to the same account (for example when a customer is issued with a new card).

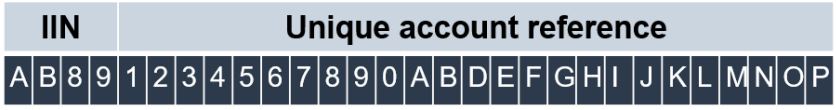
The PAR cannot, however, be assigned to an individual – so if you have two cardholders on one account, using the same PAN, they must have the same PAR.

Payment Account Reference

25 Alphanumeric identifier
First 4 digits for IIN/BIN of BIN controller
Remaining 25 assigned by BIN controller

Unique to an individual & account
Persistent across token/PAN issuance
Not routable for transactions/payments

Obtained from *either* card (tag 9F24) or through acquirer response



Details of a Payment Account Reference (PAR)

Staying on Brand

The security of mobile issuance and digitization of cards is mainly assessed through brand specific standards – so if you want a solution that works with two or more card brands, this can increase the cost and complexity of your evaluation process. However, EMVCo recently introduced the Software Based Mobile Payment (SBMP) evaluation process, which aims to consolidate all of the brand evaluation requirements into a single standard that can be assessed by a single lab for a single result that can be reused across different brands.

Section Take-Aways:

1) EMV Software Based Mobile Payments (SBMP) is about the Issuing side of mobile payments, not the acquiring side.
2) Mobile payments involves the use of replacement values, or tokens, for the PAN and cryptographic keys of the enrolled card.
3) Mobile payments usually requires the presence of a trusted party to supply these tokens, acting as a Token Service Provider (TSP).
4) Routing of mobile transactions is based on the BIN of the tokenized PAN.

Throughout this document we've tried to use the term 'payment mechanism' in place of the more traditional 'payment card' because the concept of the 'card' is just a little old fashioned these days. In the previous section we already discussed the ways in which traditional payment card data can be stored on, and used by, mobile devices such as cellular phones. In that context, we were still talking about one of the communications layers used by 'normal' contactless cards – but what if that was no longer a constraint, or even a desire? What other methods of transport can be used to communicate payment data, are easy and low cost to implement, and widely supported?

Welcome to the rapidly expanding world of QR payments.

A New Vision for Payments

At its most basic, QR payments are simply payments but using an optical physical layer rather than electrical or EM (for contact and contactless EMV). The QR code itself is used as a way to communicate data essential to the payment process, with one party presenting the code and the other using a (2D) barcode scanner or camera – such as may be found on a mobile phone – to read that code and then process the payment.

Example of a QR Code

What is a QR code? The term stands for 'Quick Response' codes, which is a reference to how they are often used to provide a relatively large volume of data in a quickly consumed format.

The most used format of QR codes is specified in ISO18004, which outlines how data can be formatted into a 'square' 2D barcode that can be read by a scanner or camera. This standard makes use of fiducial marks – the three little boxes you see amongst the other dots of a QR code – to allow for the correct orientation of the code when it is parsed by the reading system.

However, if we dig a bit deeper, QR codes both require and enable some fundamental shifts in the payment process. In some ways, QR payments is more similar to magnetic stripe payments than EMV; unlike EMV, in a QR payment there is not a two-way dialogue between the two parties involved in the payment. However, because QR codes are fundamentally based on the customer having a complex processing device with them during the payment (i.e. a mobile phone), these payments also allow for us to rethink both the implementation and need for the merchant Point of Sale systems.

As tangible examples of this QR based payments allow for the customer phone to become 'the POS', and the merchant to present the payment mechanism. This both instantly enables micro-merchants to easily take digital payments, as well as solving at least some of the 'chicken and egg' problems that face new payment systems.

Of course, we have certainly learnt somethings from the security issues that plagued magnetic stripe payments, and so QR payments also have security embedded into the code and the payment process to help prevent cloning, skimming, replay, etc.

I See You

QR based payments have three high level modes of operation:

1) Customer presented
2) Merchant presented
3) Interactive

The 'interactive' mode is where both the merchant and customer present QR codes during the same payment process, so that there can be some form of two way communications between the two parties at the point of sale. However, this is generally not implemented and will not be discussed further other than to point out that nothing prevents this from a technical point of view (although it does present some user acceptance issues).

In a Customer Presented transaction, the customer system (usually a phone app, but it does not have to be limited to this) generates the QR code and this is presented to the merchant. This code usually includes the details of the transaction, the amount for the payment, where the funds are to be transferred from, and any authentication and freshness data that is required. The merchant scans the customer QR code, this data is validated and authenticated, and the payment is processed by the merchant.

In the Merchant Presented modes, the relationship between the customer and the merchant is flipped – the merchant now has a QR code that they display, and the customer scans this with their (usually mobile) device and *the payment is processed by the customer*. This last part is important, as it allows for digital payments with merchants who have no connectivity or even electronic systems at all. The merchant QR code can be as simple as a sticker, which is unchanging between transactions and simply provides a merchant identifier to which the payments is to be directed.

However, Merchant Presented systems are not only limited to static codes and these may also have dynamic QR codes generated by the merchant, to be displayed on the merchants POS screen, mobile phone, etc.

Both Merchant and Customer presentment methods have been standardized in EMVCo's own QR based payment specification.

Securing QR Payments

Many of the same attack vectors that affect traditional payments also affect QR code based payments (which definitely includes relay and replay). However, by introducing a new way to perform payments, QR codes also introduce new attack methods that may be applied.

QR Replacement

Perhaps the most obvious is the replacement or replay of static codes – in a system where a merchant has a simple sticker showing their QR code, it's perfectly possible that someone could replace or overlay this sticker with their own and thereby 'steal' all of the payments made to that merchant.

This can be mitigated somewhat by having the QR code include (or reference for online retrieval) the merchant name, location, or other details which can be verified in a Merchant Presented transaction by one or both parties. Similar methods could be applied for a Customer Presented static code, but as this

would probably require personal details or biometric data, it is greatly more complicated than with a Merchant Presented system. Generally speaking, static Customer Presented codes are to be avoided.

Data Manipulation

Not all QR code systems provide for authentication of the data that is transmitted in the code, in fact this can be quite difficult. The common authentication methods implemented in systems is either a Message Authentication Code (MAC), which is calculated with symmetric cryptography, or a digital signature which is calculate with asymmetric cryptography. A MAC is a smaller value, sometimes as small as 32 bits (but better to be a full block size of the symmetric algorithm used) – but the use of a symmetric key requires that the authenticating entity also has that key.

With QR systems not (generally) allowing for two way communications, use of a shared symmetric key is … difficult (note I often use the term 'difficult' where others use the term 'impossible'). So we're left with a digital signature, using asymmetric cryptography, but these systems often require quite a large data payload of many hundreds to thousands of bits. So, use of cryptographic authentication of messages in QR payments is often left to some out of band method (such as online verification with the Issuer).

However, if communications is patchy – which is often a foundational reason to use QR payments – or if the authentication is not implemented at all, then any data in the code can be changed as required by a malicious entity. This could change the amount of the transaction, the payment account reference, etc.

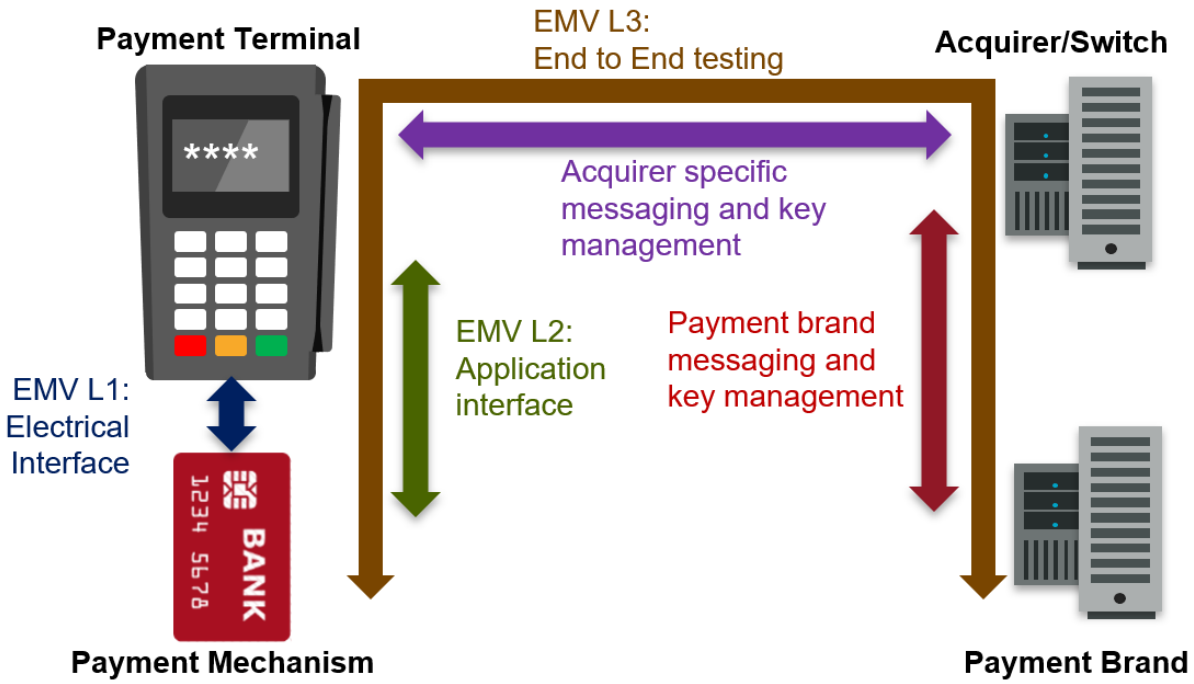
Mitigations to the lack of authentication tend to fall into three categories; don't worry about it, use online communications, or something is better than nothing. The first two are obvious, but the last one perhaps needs some explanation. It was noted that digital signatures demand quite a large dataset to be implemented, but this can have less impact of you encrypt the entire payload – of course, without a certificate (for which you don't have the payload size) you need a common private key, and so you may as well have a shared symmetric key. With a high density QR code you may just be able to pack in enough information, but it's a big ask. Generally, online verification is the best way to go if you want to authenticate a QR.

Section Take-Aways:

1) QR codes allow for a change in the existing merchant / customer dynamic, where the customer can essentially bring the POS system along with them.
2) Payments based on QR codes are easily deployed to a wide range of technologies (different phones), customer segments, and merchant types.
3) QR payments can expose an increased attack surface, as they lack two-way communications between the customer and merchant devices, and the implementation of data encryption, or authentication, is often complex.
4) EMVCo have their own QR code specification.

At this point it would be reasonable to expect that you know the high level details of what's required to perform a payment – follow the EMV payment process, then you're done. However, unfortunately, it's not that simple. Generally each acquirer and switch implements their own interface specification which you have to follow to make sure that your messages can be correctly decrypted and interpreted by them. Key management is often different between each bank as well, further complicating the issue.

This is the purple 'Acquirer specific', and red 'payment brand' messaging shown in the diagram below (repeated from our EMV section for this new context).



Areas of focus for the various levels of EMV and messaging standards

Fortunately, there are only two main protocols that are used for terminal to bank communications, so the scope of how the interface(s) work is somewhat reduced. These specifications are ISO8583 and ISO20022. *Unfortunately*, each institution does implement their own 'flavour' of these specifications, from different encoding schemes all the way through to different use of what should be formalized standard fields, but understanding the high level of how these protocols work is often important for gaining a full understanding of payment security.

One Message or Two?

Before diving into the specifics of the main protocols, it's useful to take a higher level view of transaction messaging specifically to understand the difference between a single message system and a dual message system. In a dual message system, financial requests are performed in two parts – an authorization request and then a financial transfer message that completes the transaction. It's like asking "Do you have enough money", and if the answer is affirmative then asking "Please give me the money that I asked about".

In a single message system, this is changed so that there is only one request that combines the two parts above – now the request is "Please give me this money or tell me why you can't", and the response is either affirmative or an error code that explains why the transfer did not work (which could be incorrect cardholder authentication, insufficient funds, etc).

Systems that implement dual messages can help, or at least notionally make more sense, when there is a reason for a delay between the initial request for authorization and finalizing the transaction. A good example of this would be when a signature is required by the cardholder (and needs to be verified by the merchant), or when there is a tipping amount to be applied. However, single message systems can (and do) also accommodate these types of transactions, so the decision to use a single or dual message often comes down to a question of choice and technology when the payment system was first implemented.

Of course, as we've previously discussed PCI DSS it's interesting to tease out what the two types of messaging means for the rule under PCI DSS about storage of sensitive account data 'after authorization'. With a two message system, authorization is a clearly and separately defined process, but with many single message systems it is not quite so easy – the response *is* the authorization. This can cause complexities when the response is missed for some reason (communications loss is a typical example), and the reversal message that must be sent requires the inclusion of data from the original message – potentially including sensitive data.

ISO8583 – The Standard Deviation

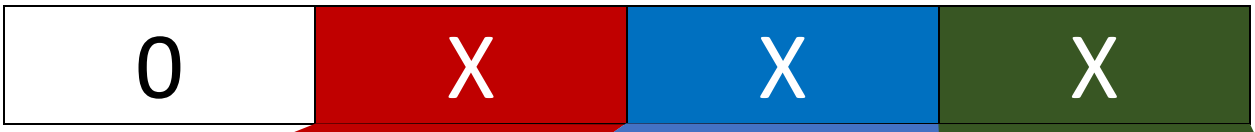
Today, the most widely used payment message protocol is ISO8583 or one of its many variants. This standard was originally released in 1987, and has been updated a few times before reaching the current version as of 2003. ISO8583 defines messages as a single data blob, with the presence of any element in the message defined by a set bit in a 'bitmap' that is transmitted at the start of the message.

To parse an ISO8583 message, you have to first decode the bitmap to determine which message elements are present, as some of these have a fixed length and some have a variable length (yes, really). Once the message parser knows what elements are present it can work through the message blog (after the bitmap) selecting and identifying each element based on if that element is noted as being present in the bitmap, and its specified length.

For example, the customer PIN is identified in bit 52 and has a fixed length of 8 bytes. Track 1,2, and 3 data are identified as present with bits 45, 35, and 36 respectively, and each has variable length specified at the start of that element. EMV data is sent in the location identified by bit 55, and also has variable length.

Message types are defined by (up to) four digits in ISO8583, with the first digit defining the ISO8583 version used (this is often set to 0 for the 1987 version), the second digit sets the high level class for the message being sent, the third digit sets the function of that message within the class, and the final digit indicates if that message is an original or a repeat transmission.

The specific details of these values are outlined in the table below (with the first value set to the common '0' value for simplicity).



Main Class	Description	Message function	Description	Message origin	Description
x0xx	RFU	xx0x	Acquirer to Issuer request	xxx0	Acquirer
x1xx	Authorisation message	xx1x	Issuer response	xxx1	Acquirer repeat
x2xx	Financial message	xx2x	Action performed advice	xxx2	Issuer
x3xx	File action message	xx3x	Advice response	xxx3	Issuer repeat
x4xx	Reversal and chargeback	xx4x	Event occurrence advice	xxx4	Other
x5xx	Reconciliation message	xx5x	Event occurrence response	xxx5	Other repeat
x6xx	Administrative message	xx6x		xxx6	RFU
x7xx	Fee collection message	xx7x		xxx7	RFU
x8xx	Network management	xx8x	RFU/proprietary	xxx8	RFU
x9xx	RFU	xx9x	RFU/proprietary	xxx9	RFU

As an example from the above, if you have an authorization request that is being set from a terminal to the acquirer for the first time that's a x100 message. A financial presentment message would be a x2xx message. Ever wondered how a restaurant can add a tip to your card payment after your card has been used in the machine? Tipping transactions are often an x200 followed up by an x220 after the tipping amount has been added (usually after you leave!).

When you check into a hotel and they 'reserve' an amount up-front during your stay? That's a x1xx authorization, which should then be finalized at the end of your stay with a financial advice message (but some hotels will send a separate financial advice, which then leaves you with less money, as both the amount charged is paid, and the authorized funds remain on hold for some period of time afterwards).

An example of an ISO8583 based 0200 transaction is included in Annex D of this book.

ISO8583 requires that any financial message must be 'reversed' if a response message is not received within a given time-out period. Reversals are x4xx messages, and are generally used very much as you would suspect – to 'reverse' the previous financial message. The idea being that when you do not receive a response, the reason for this is almost always unclear: Was the message not received by the host to which it was sent? Was the message received, but the response not sent? Was the response sent, but the transaction originating system did not receive it for some reason?

Depending on which of the above is the reason for not getting a response, the customer may or may not have been charged for the financial message. So, the only way to be sure that everything is OK after a time-out is to reverse the last *transmitted* financial message, to 'clear the slate' as it were, so that all sides are back to the same understanding about financial state.

Of the other message types, x3xx messages are sometimes used for terminal management operations – changing configurable values in the terminal, and x5xx messages are used for settlement (more on this later). The x8xx messages are 'network management' messages which are used for 'logon' of the terminal, as well as distribution of cryptographic keys to the terminal as required.

ISO20022 – The New Variance

As commonly used as it is, ISO8583 is getting quite old, and the way it is implemented is not easily adapted to new systems and technologies. For example, the standard specifies that the PIN block is only 8 bytes long – which is fine for most PIN block formats encrypted with TDES, but does not work for AES encrypted PIN blocks. Simply adding in a longer PIN block doesn't work either, as the bitmap decoding requires that the PIN block is a fixed length so that it can work through the rest of the transaction blob.

To fix this, we really need an extensible transaction format that can be more easily adapted to various changes that can be expected throughout its useful lifetime. This is the goal of ISO20022.

This standard formats messages using XML, rather than just data blobs, so that they are essentially human-readable from the outset (assuming you know what all the data values actually indicate!). A number of different message types are also defined, based on the industry, vertical, or purpose of the transaction; ISO20022 is the founding basis for many open banking implementations, and as such exists as a superset to 'just' transactions originating from Points of Interaction with a cardholder.

Each high level message type has its own data dictionary, and is defined by a four letter description – for example in terminal payments we have "caaa" (card acceptor to acquirer transactions), "cain" (card acquirer to issuer transactions), and "caam" (card ATM management transactions). Individual elements are then provided with their definition within the XML schema, such as the example for online PIN below (from the 'caaa' dictionary).

```
<xs:complexType name="OnLinePIN6">
    <xs:sequence>
      <xs:element name="NcrptdPINBlck" type="ContentInformationType17"/>
      <xs:element name="PINFrmt" type="PINFormat3Code"/>
      <xs:element maxOccurs="1" minOccurs="0" name="AddtlInpt" type="Max35Text"/>
    </xs:sequence>
  </xs:complexType>
```

The field used for the customer card data is quite a useful exemplar of ISO20022, of which we provide the schema example below.

```
<xs:complexType name="PaymentCard28">
    <xs:sequence>
      <xs:element maxOccurs="1" minOccurs="0" name="PrtctdCardData" type="ContentInformationType17"/>
      <xs:element maxOccurs="1" minOccurs="0" name="PrvtCardData" type="Max100KBinary"/>
      <xs:element maxOccurs="1" minOccurs="0" name="PlainCardData" type="PlainCardData15"/>
      <xs:element maxOccurs="1" minOccurs="0" name="PmtAcctRef" type="Max70Text"/>
      <xs:element maxOccurs="1" minOccurs="0" name="MskdPAN" type="Max30Text"/>
      <xs:element maxOccurs="1" minOccurs="0" name="IssrBIN" type="Max15NumericText"/>
      <xs:element maxOccurs="1" minOccurs="0" name="CardCtryCd" type="Max3Text"/>
      <xs:element maxOccurs="1" minOccurs="0" name="CardCcyCd" type="Exact3AlphaNumericText"/>
      <xs:element maxOccurs="1" minOccurs="0" name="CardPdctPrfl" type="Max35Text"/>
      <xs:element maxOccurs="1" minOccurs="0" name="CardBrnd" type="Max35Text"/>
```

```
        <xs:element maxOccurs="1" minOccurs="0" name="CardPdctTp" type="CardProductType1Code"/>
        <xs:element maxOccurs="1" minOccurs="0" name="CardPdctSubTp" type="Max35Text"/>
        <xs:element maxOccurs="1" minOccurs="0" name="IntrnlCard" type="TrueFalseIndicator"/>
        <xs:element maxOccurs="unbounded" minOccurs="0" name="AllwdPdct" type="Max70Text"/>
        <xs:element maxOccurs="1" minOccurs="0" name="SvcOptn" type="Max35Text"/>
        <xs:element maxOccurs="1" minOccurs="0" name="AddtlCardData" type="Max70Text"/>
      </xs:sequence>
    </xs:complexType>
```

This shows that data elements can have various values, from Boolean (TrueFalseIndicator) through text, alphanumeric, or binary values with a maximum (or exact) size. There can be one or numerous occurrences of each value. The card value itself is provided with various ways for it to be provided, from 'protected' (PrtctdCardData), through 'private' (PrvtCardData), plaintext (PlainCardData), and finally as individual data objects such as the masked PAN (MskdPAN), Issuer BIN (IssrBIN), or card brand (CardBrnd).

Obviously this is quite different from the ISO8583 format, and bears more resemblance to the formatting used in online messages than those historically used in terminal payments. This may be why the common uses of ISO20022 are open banking implementations, or bank to bank interfaces, rather than terminal to acquirer messages. However, it is quite possible that this will change over time with the increased Internet based footprint of POI systems.

Settling Down in Real Time

Most often, the payment processes we're used to on a day by day basis are not 'instant' – there is some delay (often measurable in days) between the performance of the payment and the actual transfer of funds into all associated accounts. Usually, this process is not visible to the payer as the amount is often rendered inaccessible as soon as the transaction is processed. In this way, we often do consider payments 'instant' as we 'instantly' lose access to our moneys when we pay.

However, the payee – the merchant in most cases relevant to this book – does not necessarily get the funds at the same time you lose access to them. In order for this to happen there needs to be a 'settlement' process which batches a number of transactions and transfers the monies as required in that batch. In ISO8583, settlement transactions are performed as part of a x5xx message.

There are some good reasons for the delay in payments; it allows for an increased period of oversight to transactions, so that payments can be halted or reversed if required, and because payments are batched it can allow for reduced 'chatter' across the banking network. If two banks need have 1000 $100 transactions between them, there may only be a balance of a few hundred dollars at the end of the day given the to-and-fro of the payments between them.

However, in this Internet age, the volume of traffic between institutions is much less of a concern, and there is a growing desire for 'real-time' payments. This can be implemented in many different ways, and the term 'real time payments' is used in many different contexts. However, often the backbones of a real-time payment system will use ISO20022 messaging, and people expect real-time payments to be possible with open banking APIs (which also often use ISO20022).

So Many Flavours of Vanilla

Unfortunately, with ISO20022 there are many ways to interpret the standard and format the data, regardless of the well-defined datasets that are provided. With the extensibility of ISO20022, it's also common to find implementations where there are different messages or XML tags implemented, or different ways of communicating the same data payload (for example do you put an encrypted card track in the 'PrtctdCardData' field, or the 'PrvtCadData' field, and do you put Track 1, Track 2, or both?).

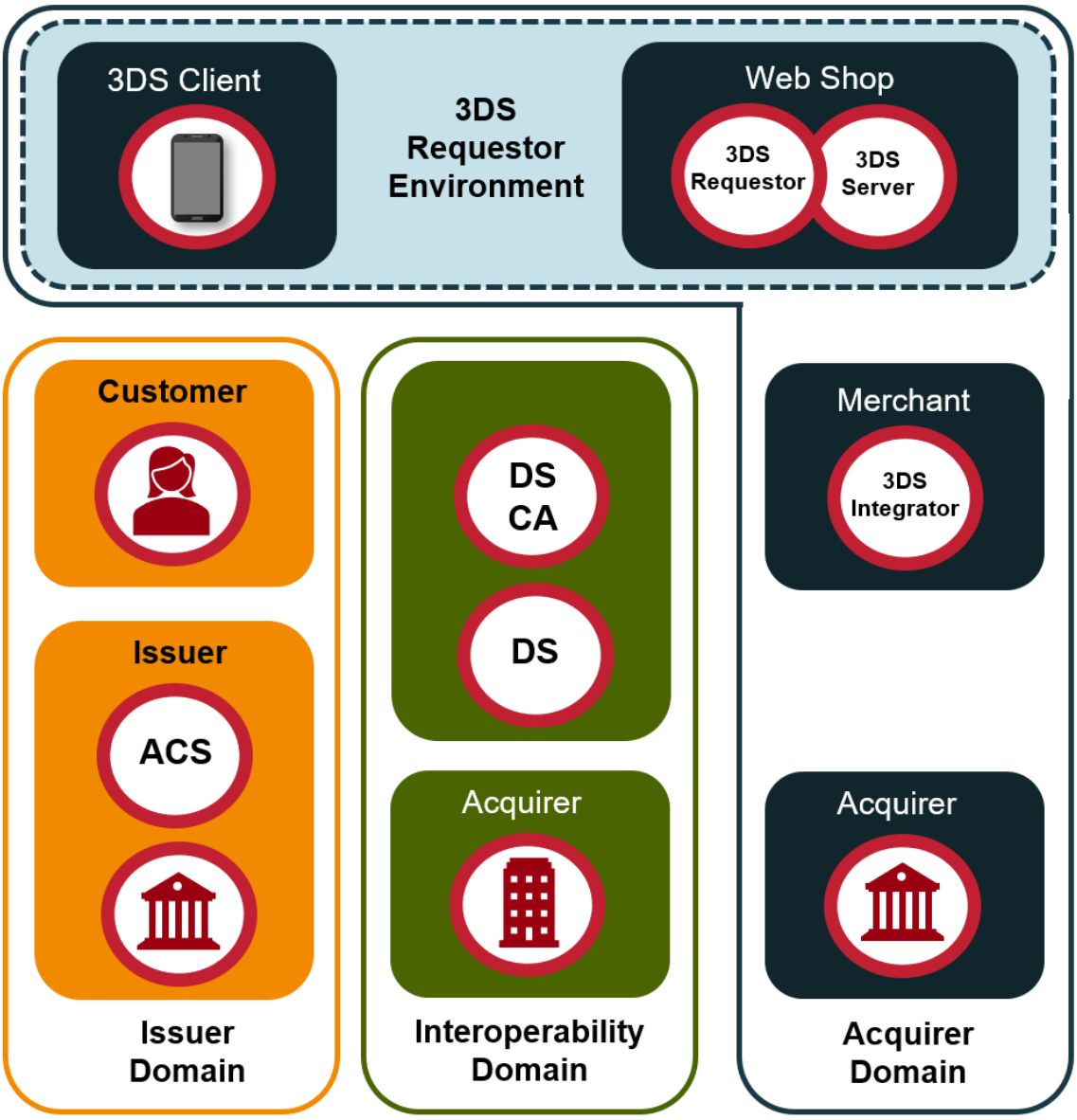
Similarly, although ISO8583 is commonly used around the world, there are almost as many interpretations and individual differences in implementations as there are organizations that have these implementations. Many different key management systems are used, which means the cryptographic key distribution changes often between banks. The exact use of each field can change with different encoding or padding types, or even uses of the field which are somewhat beyond the normal.

Discussion of the different implementations and uses of ISO8583 and ISO20022 could take up a book itself, and to be honest probably would not be that interesting. For now just be aware that if someone tells you they implement one of these protocols, ask for some more specifics on how that has been done.

Section Take-Aways:

1) ISO8583 is commonly used for payments messaging around the world.
2) ISO20022 is used for open banking, instant payments, and increasingly taking over from ISO8583 implementations.
3) ISO8583 messages are data blobs defined by an initial bitmap and subsequent payload.
4) ISO20022 messages are formatted in extensible markup language, with each field having its own designator and data definitions.
5) There are many different interpretations and implementations of both of these 'standards' in use around the world.

So far in this book we've discussed payment methods that are primarily designed for card present transactions. EMVCo 3D Secure is a specific payment process designed to secure transactions performed in a card not present (CNP) mode, primarily through Internet channels. Before we look into this in depth, we need to make a bit of a revision to our aging 'four corner' model.



The various stakeholders of a 3DS transaction

The '3D' in 3DS stands for the three domains of payment processing; the Issuer domain, the Acquirer domain, and the Interoperability domain that connects these. In the traditional four corner model, we tend to ignore the interoperability domain for simplicity, but here that domain is vitally important to the processing of the transaction as it provides the ability to directly connect the Issuer to the customer during the transaction so that the authentication process can occur.

Previously, we outlined how your PIN can be entered into a payment terminal in one country and then make it all the way back to your Issuer for authentication during the transaction, leaning on the security of PIN entry devices and HSMs to encrypt and translate the PIN through different cryptographic keys. In an online environment, such infrastructure does not exist – and more importantly it does not need to exist. When you are already using a globally connected network for your shopping, why not leverage that network to allow for the customer to directly connect to the Issuer during the transaction?

That, fundamentally, is the premise and goal of EMVCo 3DS which is enabled by the Interoperability domain.

Although the underlying technology and process for this is known as '3DS' in the industry, it's common for the payment brands to add their own marketing names to this when it is deplpoyed. Therefore, when you hear of 'Mastercard SecureCode', 'Verified by Visa', 'American Express SafeKey', or 'JCB J/Secure' – they're all talking about 3DS.

This section discusses the functional aspects of a 3DS system. Details on the security requirements are covered later in this book. The various components used in an EMVCo 3DS system are:

3DS Client – This is the end-user technology that is used to capture details from the customer during the transaction. This may be a function embedded in a web-page, a dedicated payment application, or a pay-in-app function for some other app (using a 3DS Software Development Kit, or SDK). Usually, the 3DS Client is provided by the merchant or payment facilitator.

3DS Requestor – This is the system that interfaces to the 3DS Client, accepts the data from that and passes it onto the other systems as required. Think of this as an API interface to which the client system connects.

3DS Server – The connection between the merchant environment (in the acquirer domain) and the Directory Server in Interoperability domain is provided by the 3DS Server. This obviously must be aware of the various Directory Servers that exist, to allow for further processing of 3DS transactions.

Director Server (DS) – The directory server hosts and is responsible for a number of services, which can be summarized into processing and routing of 3DS transactions according to the specific rules of that payment brand. This includes merchant on-boarding, routing of data to the Access Control Server (ACS), and management of transaction specific parameters such as timeouts, etc.

A sub-component to the DS is the DS CA, which is a Certificate Authority that is responsible for management of the TLS certificates and handling of some of the other cryptographic controls for the Interoperability domain.

Access Control Server (ACS) – The access control server is responsible for storing and maintaining the authentication information used to validate the customer during the transaction, and communicating the outcome of this validation back through to the system so that the merchant can correctly finalise the transaction.

EMVCo 3DS leans on the previous work and implementations of 3DS v1, which was first introduced in the early 2000's as a way to help improve the security of ecommerce. This security problem is based on the (relatively) anonymous nature of the Internet – the old "on the Internet no-one knows you're a dog" problem. How do you authenticate the customer when you don't know who the customer is beyond

their ability to produce a set of values which are simply printed on all payment cards? EMVCo 3DS solves this problem through two different processes, referred to as the 'Challenge flow' and the 'Frictionless flow'.

Although building on 3DS v1, EMVCo 3DS is a new system that is not backwards compatible with that technology.

I Challenge Thee to a Secret!

The Challenge flow is similar in concept to the operation of 3DS v1 – the customer is referred to an Issuer controlled (or endorsed) system to provide an authentication credential that they have previously enrolled in the system. Commonly this is a password, but other authentication mechanisms are supported by EMVCo 3DS as well.

This provides a good way to authenticate the customer, but also introduces 'friction' into the buying process; which was one of the main complaints with 3DS v1. Because of this, EMVCo 3DS also introduces a new payment flow – the 'frictionless' flow.

On The Internet, Some People Do Know You're a Dog

The frictionless flow allows for 'risk based' authentication, where data other than the authentication data you have previously enrolled with the ACS is used to validate that you are permitted to perform the transaction. This data could be simply value based – are we really concerned if you are making a $2 transaction, or if you are repeating a transaction you have made every month for several years? – or the data could be more details about you and the platform you use, based on information from the SDK, your IP and geolocation, etc.

The idea of the frictionless flow is to ensure that customers who are simply going about their business on the Internet are given the best user-experience possible, but that when there are issues or concerns, it's possible to escalate the authentication process to the Challenge flow to ensure that potentially fraudulent activity is prevented.

Section Take-Aways:

1) EMVCo 3DS is a new system that is not backwards compatible with 3DS v1.
2) EMVCo 3DS still requires technology at the merchant payment system, in the form of 3DS Requestor and 3DS Server.
3) A 3DS Client is also required, which may be integrated into the payment web-page, or the application (through an SDK).
4) Two main payment flows are supported; the Challenge flow (requiring customer input for authentication), and the Frictionless flow (using a risk-based authentication approach).
5) There are specific PCI security programs for the assessment of 3DS components.

For most people, when they think 'PCI' for payments, they think about the PCI Data Security Standard. This is because the scope of this program is so broad – it applies to all entities that store, process, and/or transmit credit card data. Under PCI DSS that data is separated into two types; cardholder data (CHD) that 'triggers' the PCI DSS requirements to be considered in scope, and Sensitive Authentication Data (SAD) that may be processed, but can never be stored (even if encrypted). The definition of these values is provided below.

Cardholder Data (CHD)	Sensitive Authentication Data
• PAN (Primary Account Number) • Cardholder name • Expiration date • Service code	• Full magnetic stripe data or equivalent data on a chip • CAV2/CVC2/CVV2/CID • PINs/PIN blocks

Definition of cardholder and sensitive authentication data types

As with the other PCI requirements, the PCI DSS is formed from a few high level security objectives, which are then broken down into more granular requirements, and finally individual security controls. At the high level PCI DSS expresses six security objectives, with twelve requirements:

Build and Maintain a Secure Network and Systems

Requirement 1: Install and maintain a firewall configuration to protect cardholder data
Requirement 2: Do not use vendor-supplied defaults for system passwords and other security parameters

Protect Cardholder Data

Requirement 3: Protect stored cardholder data
Requirement 4: Encrypt transmission of cardholder data across open, public networks

Maintain a Vulnerability Management Program

Requirement 5: Protect all systems against malware and regularly update anti-virus software or programs
Requirement 6: Develop and maintain secure systems and applications

Implement Strong Access Control Measures

Requirement 7: Restrict access to cardholder data by business need to know
Requirement 8: Identify and authenticate access to system components
Requirement 9: Restrict physical access to cardholder data

Regularly Monitor and Test Networks

Requirement 10: Track and monitor all access to network resources and cardholder data
Requirement 11: Regularly test security systems and processes

Maintain an Information Security Policy

Requirement 12: Maintain a policy that addresses information security for all personnel

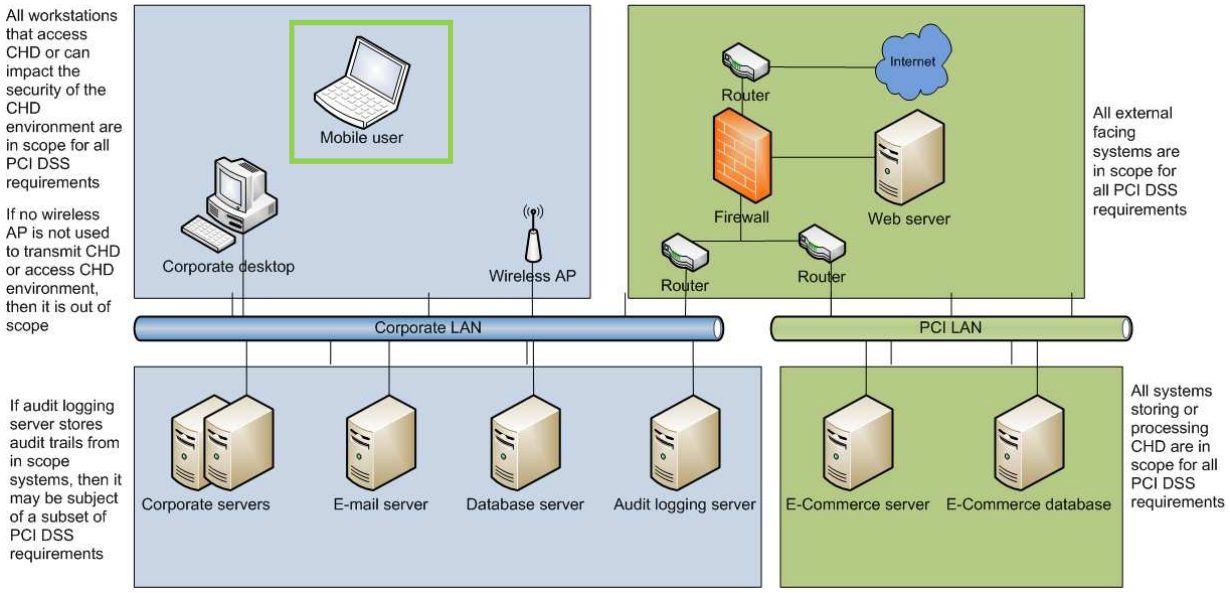
At this level it is hard to argue with PCI DSS – what reasonable organization would not agree that they need to build and maintain secure networks and systems, or that they need to protect the cardholder data they store, process, or transmit? When issues or concerns are raised regarding PCI DSS it is usually with the specifics of the more granular security controls and audit checks, where the breadth of applicability that the standard has can cause problems. A control that may be simple for one organization or merchant may not be so simple for another, depending on their business and the way that they have set up and use their systems.

So what are the specific needs of each of the twelve requirements? If you need exacting details, it's always the best idea to get those from the source standard itself, but for this book we've summarized the main points below. Given the broad applicability of PCI DSS, and the weight it has in the market, we've taken a bit more time detailing this standard than the others.

Requirement 1: Install and maintain a firewall configuration to protect cardholder data

PCI DSS does not mandate or require a segmented network, but it's certainly true that you'll want to have one if you have a network of any significant size or complexity. Network segmentation for security and PCI DSS compliance is almost a topic for its own book, but the essentials are to isolate the cardholder data you have, as well as the systems that provide security to that data. Areas in which cardholder data is exposed or processed are generally referred to as the 'Cardholder Data Environment' (CDE).

As noted above, the definition of what is in scope for PCI DSS starts with PAN data and how that is secured. In the picture below we provide a simple diagram of a segregated network that separates out 'PCI' network segments from the rest of the corporate network, based on what systems have access to cardholder data. The laptop in the diagram is also given a green box as it is assumed that this is used to manage the PCI network, or to access data on that network (through a secure connection, using two factor authentication – as stipulated by other requirements later).



Network segregation example to create a separate CDE

Requirement 1 outlines the need for a 'DMZ' to isolate systems that expose services to the Internet, and other systems, the need for secure firewall and router configurations, and having proper network diagrams. A key part of requirement 1, which will be used throughout any PCI DSS assessment, is the need for a flow diagram showing how card data enters, is stored, and leaves the various systems and network areas of the organization.

Requirement 2: Do not use vendor supplied defaults for system passwords and other security parameters

Where requirement 1 outlines the need for a secure network and configuration, requirement 2 states the need to ensure that default settings, passwords, cryptographic keys, certificates, and other configuration items are changed before components are deployed. This needs to be done as part of a specific system hardening requirement for each of the components used (example of standards such as those published by CIS and NIST are referenced), so that the process is clear, auditable, and repeatable.

In this part of PCI DSS it is also stated that there must be only one primary function per server. The intent here is to isolate any potential risks posed by each service. Having multiple services as different VMs on a single physical server is generally OK, but risk analysis is required – for example, having DMZ services on the same VM host as you have internal services should be avoided.

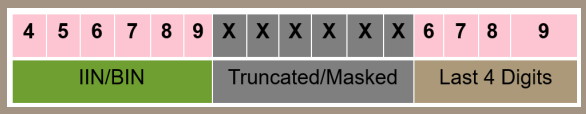
Requirement 3: Protect stored cardholder data

PCI DSS does not mandate or require that cardholder data is stored – in fact, it is generally best practice to avoid storage where possible. When you do have to store it, PCI DSS requires that you must store data in such a way that it is protected from easy oversight or compromise. This can be achieved through encryption or tokenization, for example. Where direct storage is not required, hashing or truncation may be employed.

Requirement 3 outlines the minimum requirements for cryptography and key management to ensure that the cardholder data is stored securely. It also requires that there is a process for validating all data stores at least quarterly – to confirm that they are still needed, and that there a no new ones that have been added over the course of the last twelve months.

Use of HSMs or other Secure Cryptographic Devices to manage cryptographic keys and encryption processes is not mandated by PCI DSS – but it is certainly best practice to use these where possible.

PCI DSS defines a difference between 'truncation' and 'masking' when it comes to securing the customer PAN. Both of these methods must prevent oversight of (at least) the middle six digits of the PAN, but truncation removes this data so that it cannot be retrieved or restored, whereas masking merely hides these middle values when the PAN is displayed.



Requirement 4: Encrypt transmission of cardholder data across open, public networks

Where requirement 3 discussed the needs for encryption (or other security measures) for stored cardholder data, requirement 4 outlines the needs for encryption during *transmission* of that data. The key management requirements are not as strict for this requirement as for storage, and generally most organizations will meet this requirement through use of VPNs, TLS connections, or direct encryption of the cardholder data on systems such as payment terminals.

In regards to what constitutes a 'public network' – the Internet and wireless networks / transmissions are called out specifically, but as a rule of thumb it's good practice to treat everything outside of your CDE as a 'public' network for the purposes of PCI DSS compliance.

Requirement 5: Protect all systems against malware and regularly update anti-virus software or programs

This requirement can be essentially summarized to; "use anti-virus". It's not quite that simple, the higher level intent is to have technologies and systems that monitor for malicious software and you may need other mechanisms when assessing this requirement against systems where anti-virus just does not exist, or where it does not make sense to deploy such mechanisms. For example in a BYOD scenario you may be using a Mobile Device Management implementation that provides protection in other ways, or for bespoke or uncommon computing systems (such as mainframes) you may be validating the software executing on that system in other ways.

Requirement 6: Develop and maintain secure systems and applications

We'll talk about PCI DSS and the new PCI Software Security Standards later in this book, but you can think of section 6 of PCI DSS as the software security requirements that are to be applied to in-house developed software. This requirement talks both about the need for a secure Software Development Lifecycle (SDLC), as well as the need for training of staff, testing of software, and maintenance of software throughout its deployment.

A common criticism that is leveled at this section of PCI DSS – and to PA DSS – is that it can be difficult to adapt and comply with the requirements when implementing an agile software development process. Such processes are of course common today, and so it is certainly not the intent to have these requirements prevent or stand in the way of developing software in this way. However, it is this sort of feedback that has led to PCI developing the Software Security Standards and this is also the reason why it's reasonable to expect that those requirements may eventually be folded back into PCI DSS in some way.

Requirement 7: Restrict access to cardholder data by business need to know

Requirement 7 and requirement 8 are relatively closely aligned, and maybe should be switched around in their order – this requirement outlined that you have policies and procedures in place to limit the access those people have to cardholder data based on their business need to know. It is highly likely that not everyone needs to see or have access to cardholder data, so you need to know how does and who doesn't – and prevent access by those who do not 'need to know'.

Requirement 8: Identify and authenticate access to system components

Where the previous requirement noted that access to cardholder data must be restricted, this requirement outlines the need to be able to uniquely identify individuals – so that you can then do things like restrict access (as noted, these two requirements maybe should be flipped around in their ordering).

Beyond unique identifications, user authentication is also discussed in this requirement; password length and complexity, the need for multi-factor authentication when connecting into the CDE, secure storage of user credentials, etc.

Requirement 9: Restrict physical access to cardholder data

For the last of the requirements in the 'access' section, we have this requirement which details the need for physical security. The previous two requirements really outlined the need for logical identification and authentication of users, but requirement 9 details how the computing systems and 'hardcopies' of data within the CDE must be physically secured. Use of video cameras, access control systems, isolation and protection of network ports, etc are outlined. There must be methods to clearly identify employees from visitors, to log visitor access, and to revoke temporary access from any visitor or contractor automatically when they should no longer have that access.

Security of backups and equipment that may be transported outside of the direct control of the company is covered, as well as how any data that is no longer required (see requirement 3 previously!) is to be securely destroyed.

Requirement 10: Track and monitor all access to network resources and cardholder data

It's a truism that regardless of how much security you put in place, you will never be able to keep out all attackers – if a malicious actor wants in badly enough, if they have enough resources and skills, they will always find a way in. Because of this, logging and monitoring is vitally important to the security of a system, and the requirements for this are covered in this section of PCI DSS.

However, despite how important it is, logging and monitoring can be very difficult to achieve effectively. Filtering the sheer volume of alerts, securely backing up logs to prevent tampering, ensuring proper management oversight of the data collected – all of this (and more) is not trivial. For an environment of any sort of complexity, you'll need automated log management systems to deal with the complexity of this, which then adds some complexity in how that is setup and managed. If you look at any breach report, you'll commonly find that exploits and compromises existed in the network and systems for a long time before discovery – this is a failure of the logging and monitoring systems, or how they are used.

Requirement 11: Regularly test security systems and processes

The famous cryptographer and security pundit Bruce Schneier has an adage that "Anyone can invent a security system that they themselves cannot break". This "Schneier's law" was created specifically for cryptographic systems, but applies equally to network and computing systems security as well – we are not necessarily a reliable narrator of our own security story. Section 11 of PCI DSS is designed to accommodate for this problem, by requiring that organizations have regular tests run on their systems to validate the security. This involves both scanning of externally facing IP addresses and websites by an Approved Scanning Vendor (ASV), conducted at least quarterly, as well as internal and external penetration testing conducted at least yearly.

This requirement also covers the need for systems to monitor changes in the approved configurations (through use of File Integrity Monitoring systems), as well as Intrusion Prevention / Intrusion Detection Systems (IPS/IDS). There's some argument to be had that these items may belong more aptly in a broadened requirement 5, but ultimately the need is there so let's not quibble too much.

Requirement 12: Maintain an Information Security Policy

Whilst on the topic of the correct place for PCI DSS requirements, section 12 is the requirement that I generally suggest people try to read first – despite the fact that it's at the end of the standard. This requirement details the need, and specific items to be included within, a company's information security policy and associated documentation. When many people first come to PCI DSS they quite reasonably consider it to be a technical standard, starting with networking and configuration, moving through cryptography and anti-virus, user control, monitoring, testing, etc. It's a perfectly understandable point of view, but it's incorrect.

PCI DSS is very much a business and risk standard, which has technical controls. There are a large number of documentation and scoping requirements to PCI DSS that must be first addressed before you start considering the technical controls themselves. Without the documentation, without buy-in from the C-level executives in your organization, without the correct understanding of what card data you process, where and how it is processed, and why you need that data – you will never be compliant.

Beyond the security policy, section 12 also considers the need for an incident response plan, validation and control over 3rd parties that either have access to your systems, or are providing services which affect the security of those systems.

Was it all worth it?

One way to look at PCI DSS is that by requiring a certain level of security for areas that handle cardholder data, the management of that data becomes a clear cost to the business. This allows – indeed forces – businesses to make a decision regarding if the value of that data is worth the cost of meeting the security required. Therefore PCI DSS ensures that businesses understand what cardholder data they manage, and ultimately requires that they reduce the handling of that data as much as possible, to reduce their on-going costs of maintaining compliance to the DSS standard.

Outsourcing the handling of cardholder data to other organizations has become much more common since PCI DSS has been in effect, allowing the merchants to focus on their business of selling, and the payment service providers to focus on their business of processing payments. Scope reduction can also be achieved through methods such as encryption or tokenization of cardholder data, through compliance with other PCI standards such as PCI P2PE.

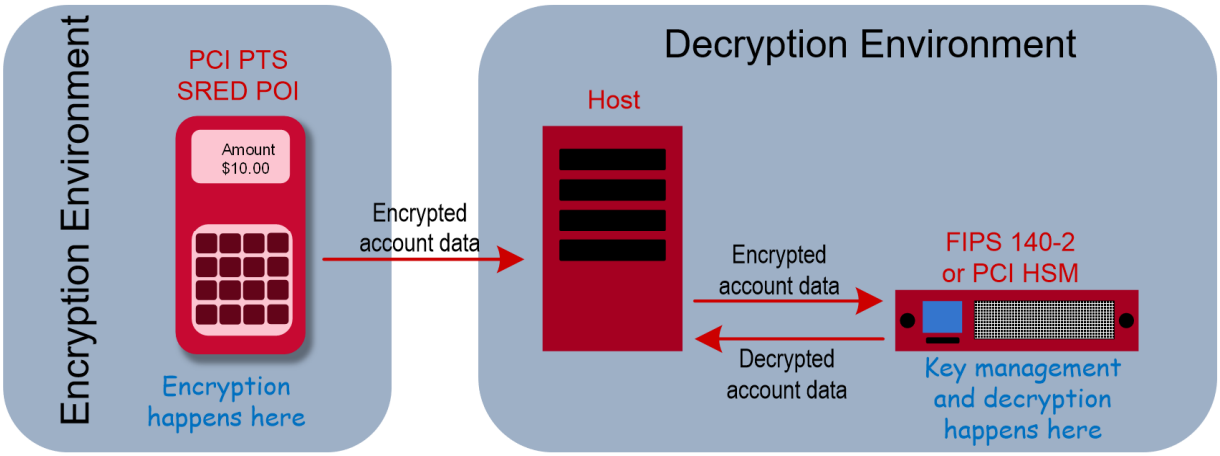
Section Take-Aways:

1) PCI DSS applies to all entities that store, process, and/or transmit credit card data.
2) A segmented network is not necessary, but is strongly recommended.
3) PCI DSS is both a technical and business risk standard.
4) PCI DSS requires a significant volume of documents in terms of policy, procedures, configuration settings, etc.
5) Consider outsourcing cardholder data handling where you can to reduce the scope of PCI DSS on your organization.
6) PANs used in mobile, QR, or EMV 2nd Gen payments remain in scope of PCI DSS.

With PCI DSS often considered a significant burden to comply with, maintain compliance, and provide validation to, it is common for organizations to seek ways in which they can reduce the scope for some or all of the PCI DSS requirements. One such way is through the use of data encryption at the point of card acceptance – from the ***point*** of acceptance to some other logically distant ***point*** where that data is decrypted. Between these two points, it is possible to descope many of the PCI DSS requirements.

Welcome to Point to Point Encryption (P2PE), where card data is accepted with a PCI PTS Point of Interaction (POI) device, encrypted using its functions for Secure Reading and Exchange of Data (SRED), and then transmitted to some other point for decryption.

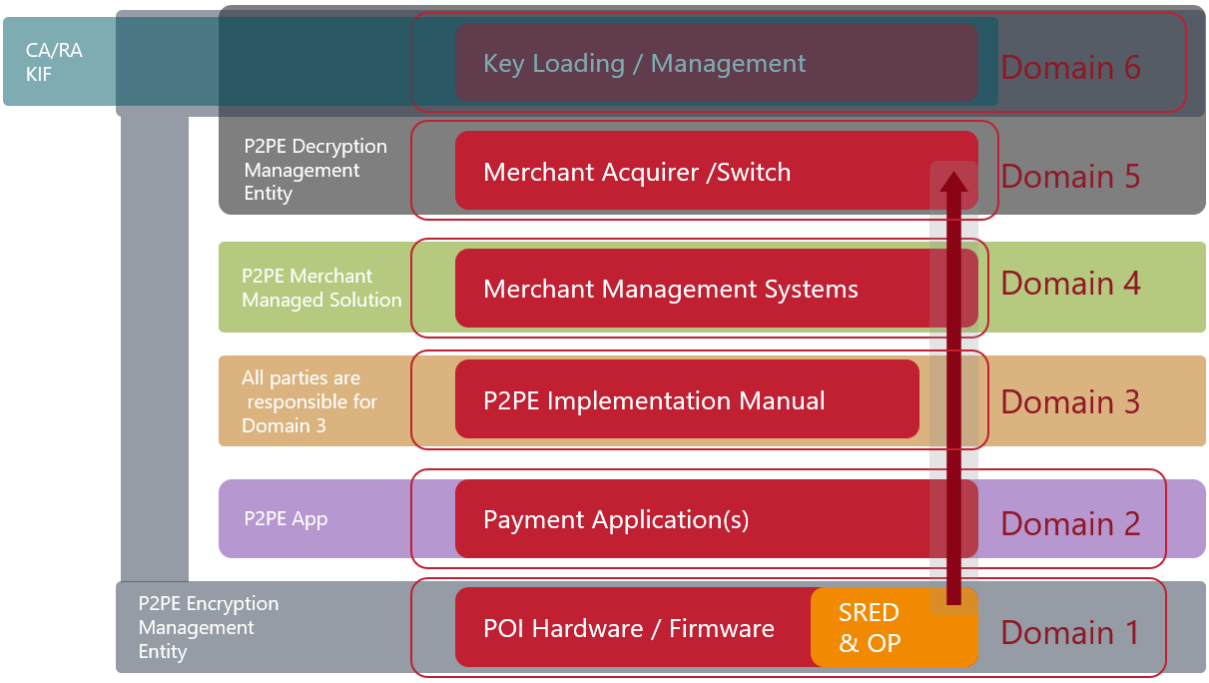
PCI P2PE defines two main scenarios for where the two 'points' of decryption may exist. The encryption must always happen at a PCI PTS compliant POI device (we'll cover more on PTS later), but the decryption point may be either an entity separate from the merchant, such as an acquirer or payment switch, or it may be the merchant themselves. Both of these scenarios can be expressed in the illustration below, where the decryption environment no the right is either some separate area of the merchant, or a different organization altogether. The space between the two blue boxes, where only cardholder data encrypted by the POI device passes, can be assumed to meet the requirements of PCI DSS for securing that data.



Example of a PCI P2PE system

PCI P2PE expresses its requirements in six domains, with each covering a different aspect of the implementation. These different domains may also apply to different entities – one of the main differences introduced in v2 of PCI P2PE was the ability to modularize a solution and have different aspects implemented and approved separately (previously, compliance required a single 'solution provider' to take ownership of the entire set of requirements).

The illustration below shows how the different modules of P2PE fit together to ensure that the cardholder data encrypted at the POI device (using the approved SRED functions) is secured through to the point of decryption.



PCI P2PE domains and their scope

Domain 1 essentially confirms that the solution is using validated PCI PTS devices, and that those devices are received, implemented and used properly. The payment applications which are being used to collect the payment data and encrypt these through the POI SRED functions is assessed in Domain 2; this is important to ensure that the application is in fact actually using the correct encryption functions provided by the PTS platform, and that it is not storing cardholder data when it should not be, or implementing security controls or functions itself which have not been validated through the PCI PTS process (this covers the 'app gap' we discussed earlier in this book).

Domain 3 outlines the requirements and content for a 'PIM', or P2PE Implementation Manual, which makes clear how the system should be deployed and used. For systems where the cardholder data is also being decrypted by the merchant themselves (say where they implement their own switch, or other back-office functions), Domain 4 lists the additional requirements they must meet to do this. Domain 4 would not be assessed against systems where the merchant is not doing their own decryption.

Domain 5 provides the details for the decryption environment, which would apply to either an acquirer/switch or a merchant who is doing their own decryption, and finally Domain 6 outlines the key management requirements for how cryptographic keys used for the encryption/decryption processes are generated, secured, deployed, etc. This domain 6 is quite similar to the requirements in PCI PIN, which we will discuss later.

Compliant? Not NESA-ssarily

Having a Point to Point solution assessed and found compliant to the PCI P2PE requirements is not a trivial process, and there are still many merchants who are using systems that provide encryption of cardholder data, but that are not compliant to PCI P2PE. To help bridge the gap, to allow such merchants to still enjoy some of the benefits of scope reduction for PCI DSS through use of these types of solutions, PCI developed the Non-listed Encryption Solution Assessment (NESA).

This is basically an assessment of a non-PCI P2PE encrypting solution, to the PCI P2PE requirements so that any gaps to that compliance can be identified and understood by the both the merchant and auditor that they use. This helps ensure that the gaps can be mitigated through other controls in deployment, so that the overall protections can provide scope reduction as per PCI P2PE approval.

Section Take-Aways:

1) PCI P2PE allows for scope reduction during a PCI DSS assessment for merchants that use these systems for the capture of cardholder data.
2) Compliance to P2PE requires the use of PCI PTS approved devices, which must be approved to provide SRED functions.
3) P2PE consists of six different domains, which may apply separately to different entities involved in the overall encryption/decryption process.
4) It is possible to use an unapproved encryption solution if it is assessed through the Non-listed Encryption Solution Assessment (NESA) process.

We've previously discussed the need for secure hardware, known as Secure Cryptographic Devices, for handling customer PINs and cryptographic keys. How do we validate that any specific device is 'secure' enough to qualify for this, and what are the requirements that apply? This is covered under the PCI PTS requirements and testing process, and in fact one of the defining factors to determine if PCI PTS applies to a device or not is to question if that device process or manages PINs in some way (there is a single exception to this rule in that Software PIN on COTS solutions can accept PINs on consumer grade hardware – we cover this standard later in the book).

Tamper protection in payments security is often quantified in three main aspects:

Tamper Resistance – this is where attacks are made difficult, but not necessarily prevented or detected. A safe is an example of a tamper resistant device.

Tamper Detection – where attacks are actively detected as they are being performed. An alarm system is an example of a tamper detective system.

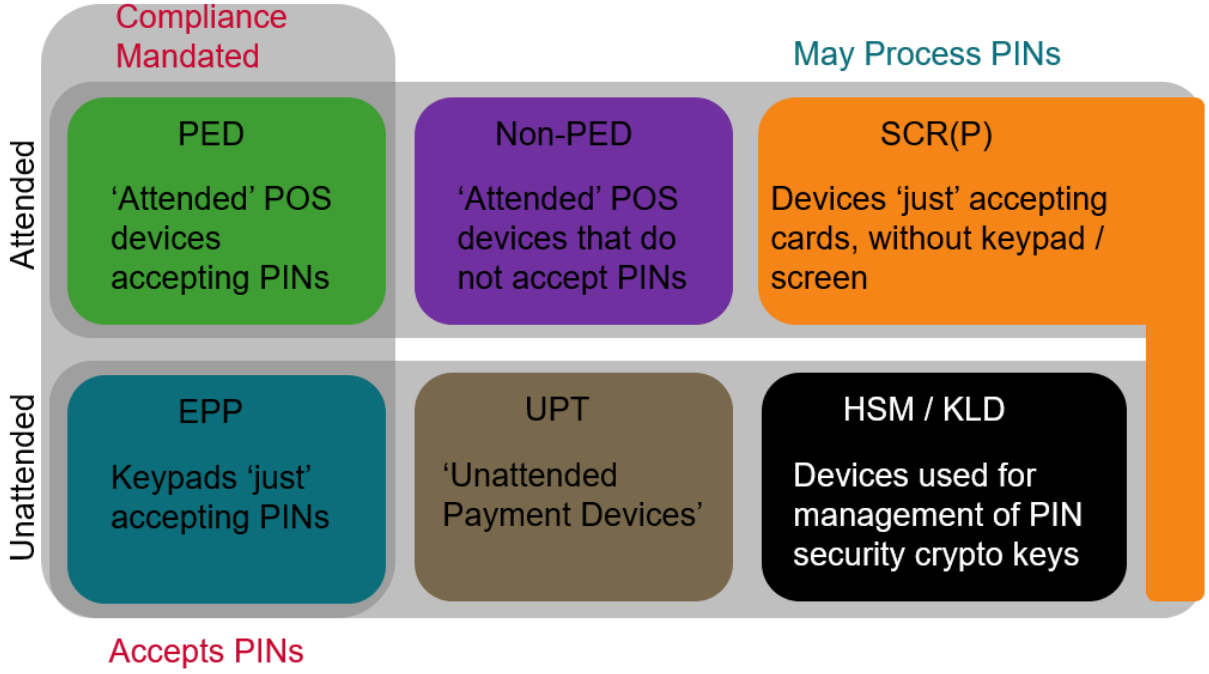
Tamper Responsive – is an extension of tamper response, where upon detection of an attack the system responds to that attack in some way. The police or fire departments attending an alarm is an example of tamper response.

PCI PTS devices generally must provide tamper responsiveness to secure the assets that they protection (initial devices approved to PCI HSM could be purely tamper resistive, but this is no longer acceptable). The tamper response implemented is most often the erasure of the cryptographic keys that are contained by the device, and that erasure results in the device entering an inoperative state.

Under PCI PTS there is actually two sets of requirements – those for 'POI' devices, literally Points of Interaction for a customer such as a payment terminal or ATM keypad, and a standard for Hardware Security Modules (HSMs) which are used at payment backends to store and manage cryptographic keys. Within these standards there are eight approval classes;

- PED – a PIN Entry Device; your 'standard' payment terminal you'd use at a merchant store
- non-PED – a payment device that has security for cardholder data but is not used for PIN entry
- EPP – an Encrypting PINPad for PIN entry only, such as an ATM keypad
- UPT – An Unattended Payment Terminal such as a kiosk, ticketing or vending machine, etc
- SCR – A Secure Card Reader for accepting cardholder data only
- SCRP – A Secure Card Reader (PIN) which is an SCR to be used with a SPoC solutions (more later)
- HSM – A Hardware Security Module, which stores and manages cryptographic keys
- KLD – A Key Loading Device, which is used for loading keys into one of the devices above

These approval classes are pictorially represented below, along with how they are used and the compliance mandates that apply to these systems.



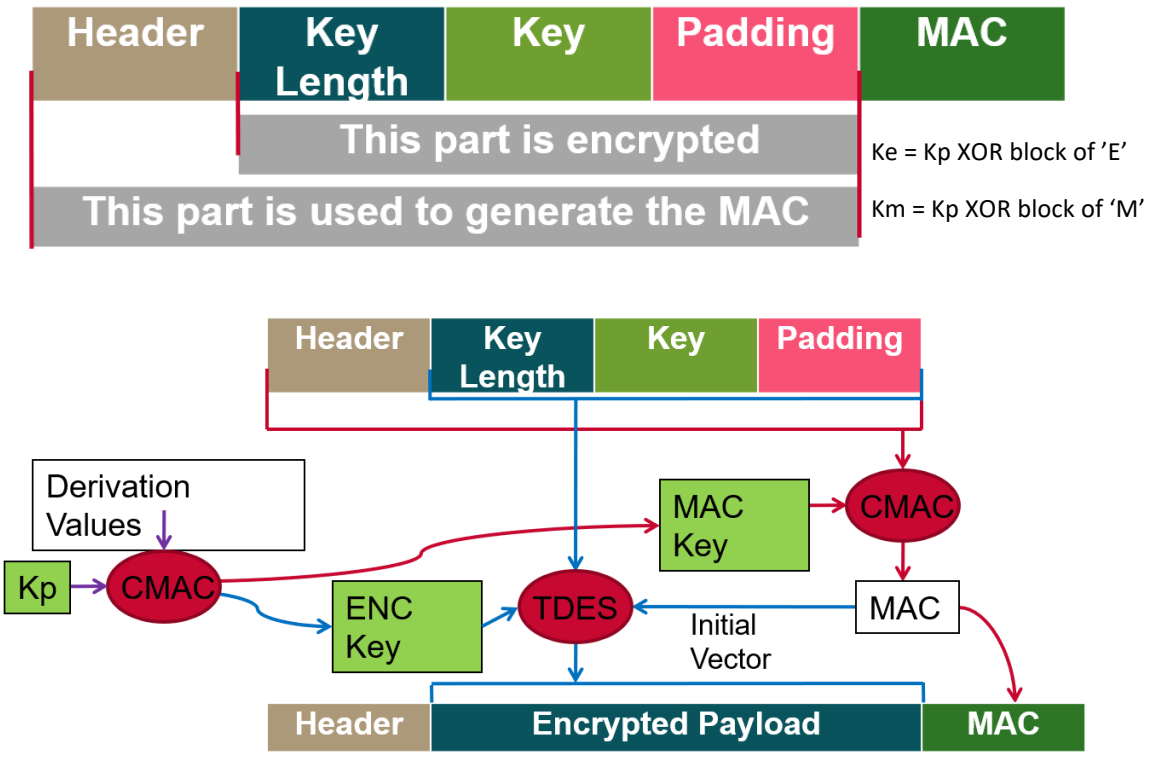
PCI PTS approval classes, their uses, and compliance mandates

HSMs - Protecting the Crown Jewels

A Hardware Security Module is a primary security foundation of any PIN based payment system, and are used is many other areas as well. At their most simple, a HSM is used to secure cryptographic keys during their use so that they are not exposed in the memory of general purpose computing systems, across networks as they are transmitted, or on media where they are stored. This protection is provided through tamper responsive features of the HSM, physically securing the processing electronics of the device, as well as logical hardening of the HSM.

Usually a HSM does not actually store the working keys that it uses for the various commands that are used for its operation. Instead the HSM stores only a few, or in some cases only one, high level master keys under which all other keys are stored. The various HSM commands then actually contain, or reference, the encrypted working keys for that command which are decrypted under the relevant master key as part of the operation of the command.

An important aspect of this operation is that it must be ensured that the right working key is used, and it is not possible for a malicious actor to replace (for example) a data key with a PIN key so that encrypted PIN blocks could be easily decrypted. This protection of key use is achieved through the application of 'key blocks' or 'key wrapping', which is a specific format for an encrypted key such that the encryption process ensures the key integrity, confidentiality, an some metadata on the key such as key use, algorithm, size, etc. A common standard used for key wrapping is ANSI TR-31 (although other standards and methods do exist). The two methods of applying TR-31 based key wrapping are illustrated below.



ANSI TR-31 Key Wrapping, 2005 version (Top), and 2010 version (bottom)

In the above illustrations, the 'header' is the metadata for the key block, which contains the important information about the key use and type. Key blocks are protected with a single 'Key Block Protection Key' (Kp) which is then diversified to two working keys – the Key Block Encryption Key, and the Key Block Authentication Key. The main reason for the change between the 2005 and 2010 versions of ANSI TR-31 is to remove the use of variants to diversify Kp into the encryption and authentication keys – as use of variants is generally deprecated for new use in cryptographic processes and key management. Use of CMAC is considered best practice for such key derivation moving forward.

Point of Interaction

PCI PTS POI is divided into several modules, which are then divided into specific Derived Test Requirements (DTRs). A list of these high level requirements (for PCI PTS v5.1) is provided in Annex C of this document, with a summary of each module below.

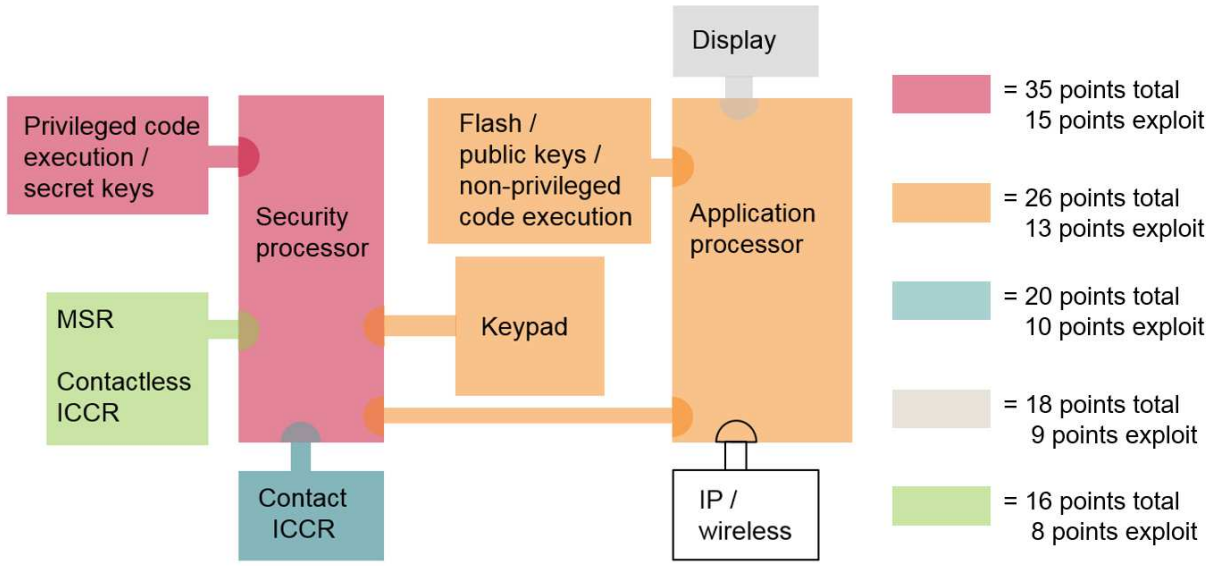
Module 1 is the core module, which must be assessed against in-scope components of any device being testing. This module is further divided into Core Physical (DTR A1 – A10), Core Logical (DTR B1 – B20), Online PIN (DTR C1), and Offline PIN (DTR D1 – D4) requirements. Each requirement outlines the security for a particular asset class (PINs, cryptographic keys, display prompts, etc), or security function provided by the device. For example, DTR A1 covers the security requirements for the customer PIN, how that is entered, the path from the keypad to the security processor, etc.

Physical security in PCI PTS is quantized in terms of an 'attack costing' which allows for a (hopefully) objective assessment of any arbitrary attack that the PTS laboratory has developed to extract or access the asset in question. Each asset type has a different level of security required, which is reflected by the

minimum attack cost that must be achieved for attacks on that device – the more points required, the higher the level of security that must be implemented to protect that asset type.

Attack costings are divided into 'identification' and 'exploitation' phases, where the identification phase quantizes the effort and equipment required to identify and develop an attack, and the exploitation phase is the effort and equipment required to perform that attack on a device in the field.

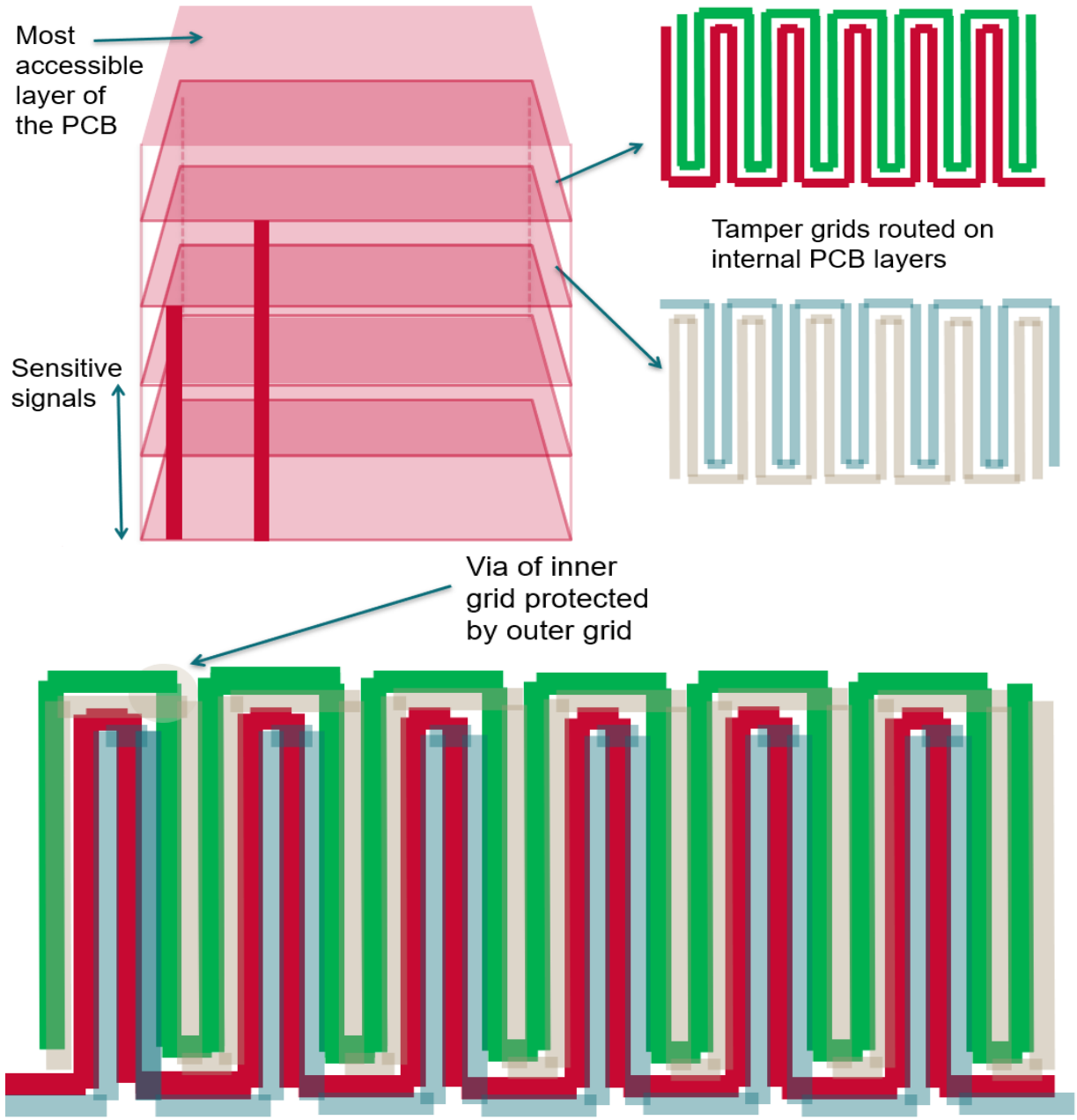
The picture below shows the overall costs associated with each asset type within a PCI PTS device.



PCI PTS minimum attack points per asset type for a two processor POI design

One important point to take away from this diagram is that there is a founding assumption that attacks will be possible on all asset classes. No device is 'tamper proof', only tamper resistant or tamper responsive to a certain level. That level in PCI PTS is assessed through the costing process.

So how do you protect the sensitive signals and information within a POI device? There are many ways, but most designs will implement at least some tamper switches and internal tamper grids. A tamper grid is a group of traces routed within the PCB which carry a signal that, if cut or shorted, will cause the security processor to erase sensitive keys and cease further operation. Tamper grids should be routed so that they protect the sensitive signals of the keypad or processor memory, etc, and PCI PTS requires that there are at least two layers of tamper grids used to protect such signals, and that they are routed so that the most accessible tamper grid layer protects the vias (inter-layer connections) of the inner most tamper grid. This is illustrated below.

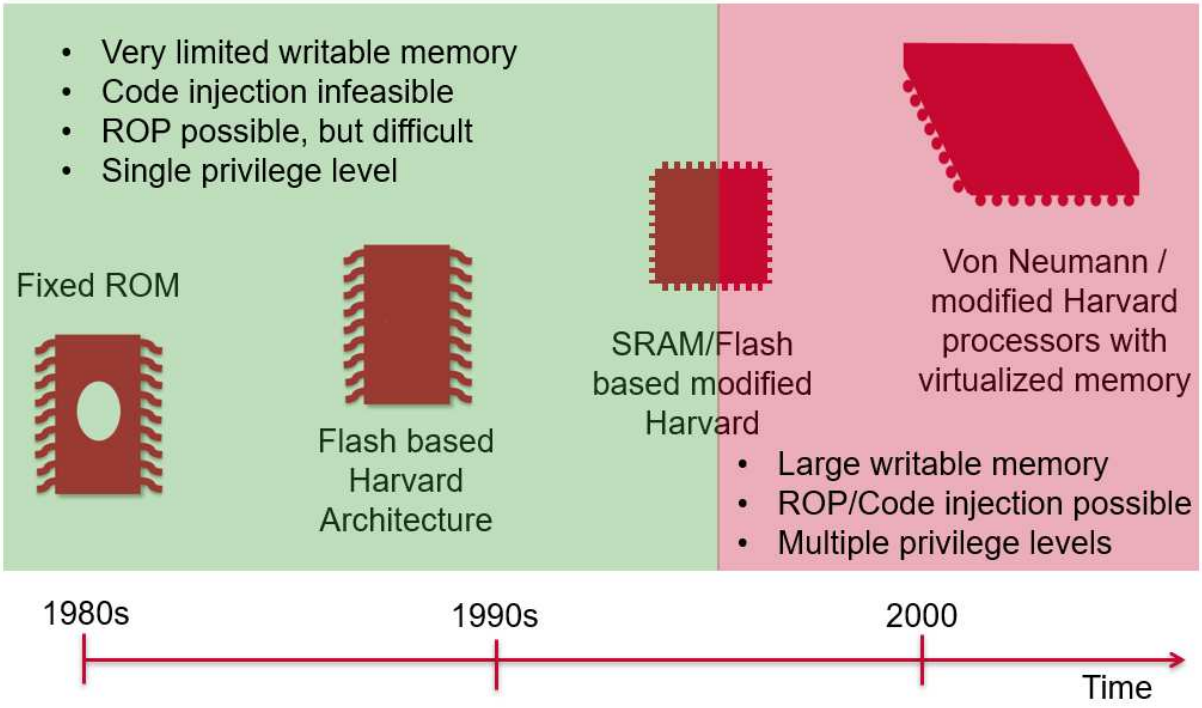


Examples of a tamper grid layout over two internal layers of a POI PCB

Tamper switches are literally switches that are held closed by the POI casing, or by the connection between two internal components (such as two PCBs). They are designed in such a way that opening of the casing or other attempts to access the internal areas of the POI will result in the disconnection of the switches, resulting once again in the erasure of keys and rendering the device inoperative.

The core logical requirements outline the security that must be applied to the firmware of the device. In PCI PTS context, 'firmware' means all software in the device that is required to protect the assets and meet the PCI PTS requirements. The logical requirements have been subject to the most change over the course of the life of PCI PTS, since its inception in 2004. This is due to the large change we've seen in

the way in which PIN entry devices are implemented, largely because of the increase in processor speed, capacity, and requirements from the market for new functionality. Early POI devices would use simple 8 bit processors which had segmented data and program memory, making many logical attacks much more difficult. Today, Linux and Android are very common platforms to be used in these devices, which greatly expands their security exposure.



As processors have become more advanced, logical complexity has increased

Additional logical requirements are found in the Open Protocols section (Module 3) of PCI PTS, which covers the security of switched and wireless connections, such as Ethernet, Bluetooth, WiFi, etc.

Module 4 of PCI PTS is for Secure Reading and Exchange of Data (SRED) which outlines the specific security requirements for devices which are to designed to protect cardholder data through encryption. This module is optional for most instances, but must be assessed for devices that are designed for use with mobile phones, or with P2PE solutions. It's important to note that SRED devices may still be able to output card data in the clear, based on their configuration, and they do not mandate the use of encryption at the magnetic readhead, or at the IC card reader module – only that the data is output from the POI encrypted (unless the POI is configured to not do this).

For HSMs the requirements are broadly similar in terms of having physical and logical aspects, but here there is more focus on how the HSM manages the keys and PINs that it processes. PCI HSM was developed with this specific focus – how are these devices securing payments and payment data. Prior to the release of PCI HSM, standards such as FIPS140-2 were often relied upon, which is a generic cryptographic module standard published by NIST. Although there is nothing necessarily wrong with this standard, its focus is more on the correct use of cryptography, rather than any payment specific functions; hence the need for PCI HSM.

One of the most important aspects of PCI PTS is assessment of the key management of a device, whether that device be a POI device or a HSM. These devices are designed to encrypt, translate, or verify PINs, and they do that with cryptography – any cryptographic system requires key management, and a cryptographic system that has more than one key implies the need for some form of cryptographic key hierarchy. Such a key hierarchy has one or more 'master' keys, under which other keys are loaded or stored, until we get to the point at the lowest levels of the hierarch where we have the 'working' keys that are actually used for the encryption of PINs or data.

The assessment of the actual implementation of key management once the PCI PTS devices are deployed is the scope of the PCI PIN audit, which we discuss in the next section.

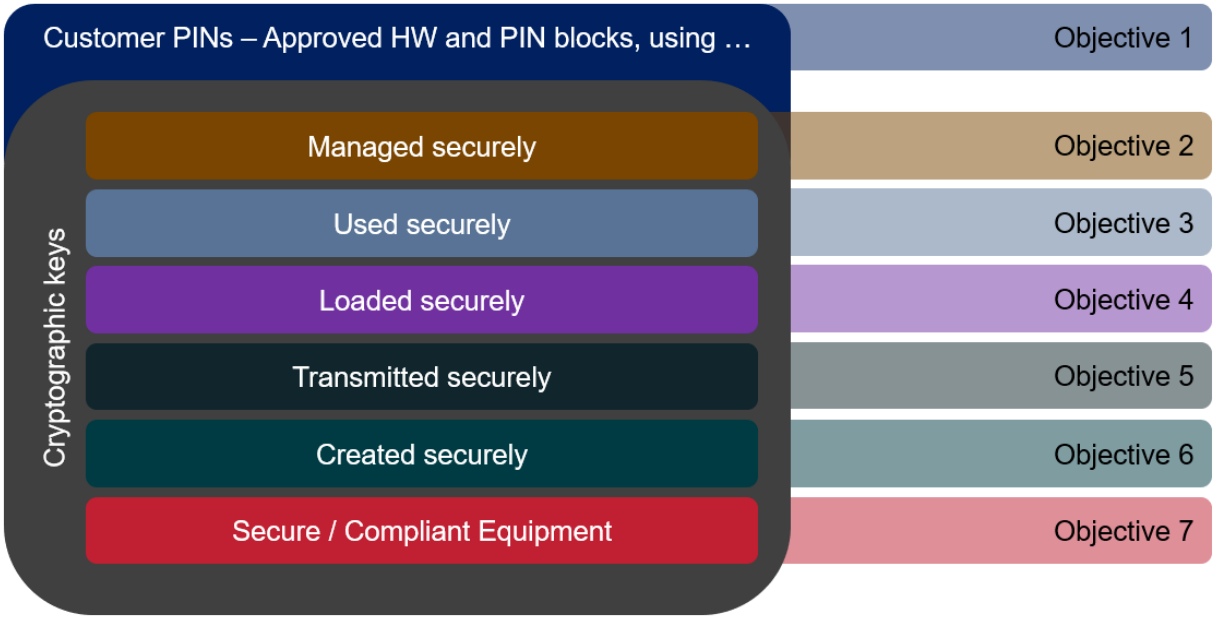
Section Take-Aways:

1) PCI PTS primarily assesses the security of devices designed to handle PINs.
2) A PTS device may provided different levels of security to different asset classes.
3) PCI PTS confirms the ability of a device to implement secure key management, but is not able to confirm that any implementation in the field does this – that is the role of the PCI PIN audit.
4) A device approved to PCI PTS is not tamper proof – it is tamper responsive. Attacks are still possible.
5) The tamper response of PCI PTS devices is generally to erase their cryptographic keys.
6) Not all terminals encrypt cardholder data for transmission, even if approved to SRED.
7) SRED encryption does not necessarily mandate encryption at the point of the card read, only encryption on data egress.

The PCI PIN standard is an audit standard for the implementation of key management in payments – specifically for keys that are used for PIN security, either directly or indirectly. PCI PTS also involves assessment of key management in devices, so what's the difference? During a PCI PTS evaluation the laboratory determines if the POI device or HSM is able to implement secure key management processes and practices. In PCI PIN, an audit is performed on the actual implementation of key management by organizations using these devices, to make sure that they are using them correctly, following the right processes and procedures, and generally 'walking the walk' for good key management.

The Seven Levels of Hell

The PCI PIN requirements are divided into 7 objectives which are then further segmented into 33 individual audit requirements. These requirements are designed to provide oversight on what should be good financial key management – the general rules for this are provided in a 'cheat sheet' in Annex A.



PCI PIN Objectives

PCI PIN defines certain 'approved forms' which are the only ways cryptographic keys may appear or be used. These forms are:

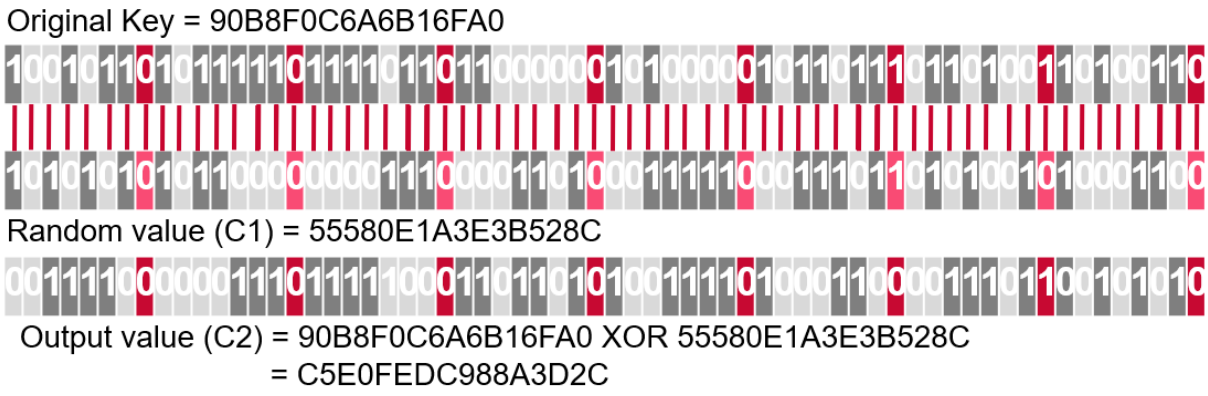
- Encrypted under another key of equal or greater strength
- Stored within an approved Secure Cryptographic Device
- Managed as two or more full length components, or as part of an M of N secret sharing scheme

Much of the audit process for PCI PIN is validating that the keys are used only in these approved forms, and that the Secure Cryptographic Devices are compliant to the relevant requirements (such as PCI PTS).

Start at the Beginning

With the PCI PIN audit focused on the 'lifecycle' of cryptographic keys, it's important to understand what security exists around the 'birth' of these keys – how they are generated and loaded into the HSMs and POI devices that are used for PIN processing. With HSMs this usually starts with the manual loading of keys through a physical interface of the device in a process that is sometimes called a 'key ceremony'. Here, two or more 'components' of the HSM master key(s) are manually entered into the HSM, through a physically keyboard, 'dumb' terminal (these days this is usually a laptop with a Linux 'live CD' and no non-volatile media), or using a dedicated secure entry device that is designed for use with that HSM.

A key component is a value that in and of itself offers no details about the resultant value of the cryptographic key that it is used to generate, not even a single bit. A common method for creating cryptographic key components is to take a key and then XOR that key (K) with a random value of equal length (C1) – thereby generating a third value through this process (C2 – see illustration below). C1 and C2 are then your key components, which when combined through another XOR process will regenerate the original cryptographic key.



Generation of key components (C1 and C2) from an originating cryptographic key value

Another process for regenerating key values is through an 'M of N secret sharing scheme'. Without going into the details of the mathematics involved, this process allows for the use of some subset of total shared values to regenerate the original key – so for example you may have five people who each have access to one of the values, but need only two of these people to come together at any time to regenerate the original key value. This sort of scheme allows for more redundancy and operational robustness to a standard XOR based component scheme, as you can still manage key recovery if there are some of the people away on holiday or unavailable for other reasons.

Staff who have access to key recovery values, such as components, are referred to as 'key custodians'. These people obviously have a position of high trust, and must go through a process of being correctly vetted and educated on their role and responsibilities. Validation of this process, of the way in which keys are loaded and managed, how key components are stored and secured, etc – this is a large part of the PCI PIN audit.

Got to Keep Them Separated

The need for use of key components and key custodians speaks to two of the foundational requirements for key management – dual control and split knowledge. These items are often misinterpreted or mixed up, and so it's worth taking some time to detail what they mean, why they exist, and to what they apply.

Dual control means that there must be at least two people involved in the authorization of an activity or process. Another way to look at dual control is that it must not be possible at any time in a dual controlled process to find a way to blame one single person for the failure, insecurity, or un-approved authentication of some process. Key loading often requires dual control, to ensure that no one person can subvert the process, expose or manipulate key material. This may involve having two passwords entered as part of the key loading process, or ensuring that there are always two people in any room observing a process underway.

Split knowledge is a requirement for securing data to ensure that no one person may ever know any information about the data they are protecting. In key management this is implemented through enforcement of the key forms as detailed above. Splitting a key 'in half' and giving a half to each custodian ***does not*** meet the dual control requirements because it allows for each custodian to know half of the key.

So, if we look at key loading into a HSM, dual control is usually enforced through use of two password – one entered by each of the two custodians (or two of the N custodians implementing an N of M scheme). Split knowledge is implemented through the use of the components or secret shares. Why does the use of components or secret shares not provide both split knowledge and dual control? After all, it ensures that there are two or more parties involved in the key entry!

This does not work for dual control as nothing is stopping the one custodian from entering both components or secret shares. That must be implemented through passwords (or some other control), ensuring that you cannot ever blame a single custodian for entering all data for a single key.

That's dual control.

Working Key; I am your Father

Common cryptographic hierarchies assessed under both PCI PTS and PCI PIN involve one of the following types of key management:

a) Fixed key
b) Master/Session
c) Unique Key Per Transaction

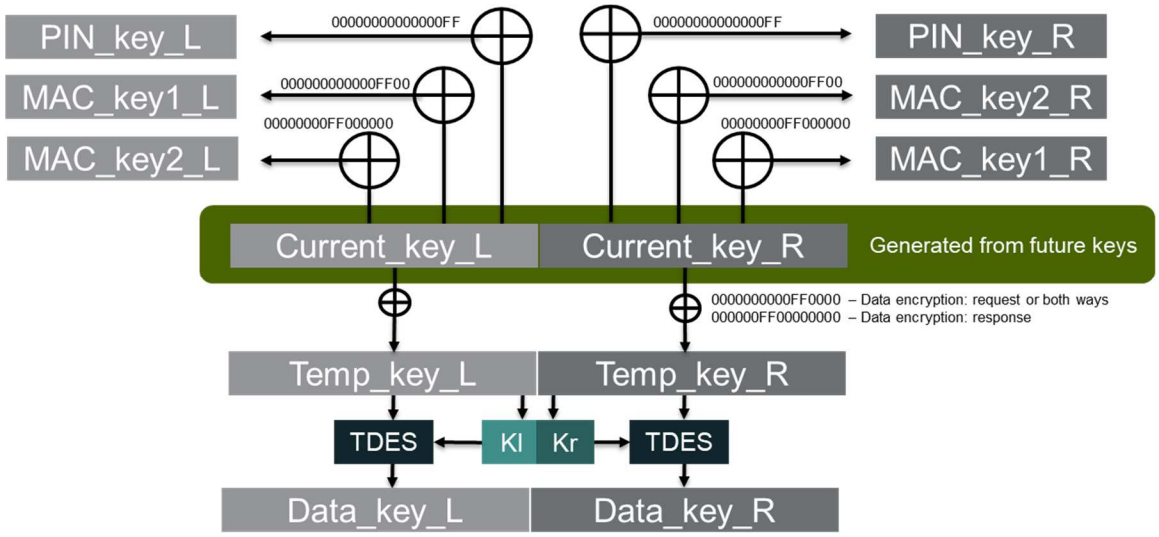
Fixed key management is where one or more keys are loaded into the POI device directly as working keys, and these keys never change over the deployment of the device. Because of the lack of key rotation, fixed key is not considered ideal from a security point of view, but is still in common use in some areas and verticals.

Master/Session key management is where there is some form of key hierarchy, such that a 'master' key is loaded into the POI prior to initial deployment, and then that 'master key' is used to deploy new working keys to the device as needed. Many master/session implementations can have multiple levels to their hierarchies, such that the master key loads another Key Encrypting Key (KEK) which is then used

to load a working key. Best practice would have a different KEK for each working key function, so you'd have a PIN KEK loading the PIN key, a data KEK loading data encryption keys, etc.

Master/Session key management can start simple, but can become very complex very quickly if there are multiple layers to the key hierarchy, and different ways of loading keys. The master key itself may in fact start as a public key, loaded into the device during manufacture, which is then used as part of a 'remote key injection' (RKI) process to load the first secret key into the device.

Unique key per transaction implementations ensure that there is a different key used for each communication session to the financial host. The most common implementation of this type of key management is defined in ANSI X9.24 and is referred to as DUKPT (usually pronounced as 'Duck Put', which I personally hate …). In this implementation, a new 'current key' is derived for each transaction, and from that current key multiple working keys for different purposes can be derived. An illustration of this process is provided below.



Deriving working keys from the DUKPT current key (for 2010 TDES DUKPT)

The illustration above shows how to derive keys for TDES based DUKPT. The most recent version of ANSI X9.24 defines an AES version of this, which uses different derivation methods (using CMAC, which is best practice for key derivation). Some devices take the existing TDES version of DUKPT and simply change the key to an AES key, and still refer to that as DUKPT (but with AES) – this is not correct. DUKPT is defined as a TDES method and an AES method, and they are quite different; taking the TDES method and altering this to use AES keys is ***no longer*** DUKPT.

The + symbol in the diagram above indicates the use of binary addition (or XOR) between the key and another value (shown on the arrows, or above the + in the diagram) to generate a new key. In key management this is referred to as 'varianting' the key, and generally varianting is something that should be deprecated for any new solution. Why? If you can compromise any particular key variant, you can directly calculate all other keys from that (through the application of the binary values that are added to each).

To further protect keys PCI PIN has recently mandated the use of key blocks or key wrapping techniques to protect the purpose and use of cryptographic keys – we discussed how this is applied with HSMs in the PCI PTS section of the book. Checking key values can be done using a Key Check Value (KCV, sometimes also called a KVC), which is a standard method of using the key to create what is effectively a checksum (encryption of an all zero block is used often, although the standard for AES KCVs is more complex). It's good practice to ensure your HSM is not using default keys by checking the KCVs before use!

A large part of the PCI PIN requirements covers documentation – each of the seven objectives ends with a requirement that you must have documentation for the other requirements in that section that exists and is 'demonstrably in use'. This documentation aspect is often something that catches people off guard – it is not enough to be doing the right thing, you must have documentation showing people how to do the right thing.

You must Choose Wisely

Another area of confusion in PCI PIN is the requirement to use PCI PTS approved devices for PIN entry. Compliant devices are listed on the PCI webpage as illustrated below (this is a fake example), and can be easily checked, but knowing exactly what to check is important. Here the listing shows us that this device model name is 'SuperPED' with a hardware version of 4.00.xx, a firmware version of 1.00.05, and an application version of 1.0. All of these items must be in place and correct for any deployed device of this type, in order for it to be compliant. Further the listing shows us that the device is approved for online and offline use, with specific key management types.

SuperPED
http://www.superped.com

SuperPED

Hardware #: 4.00.xx
Firmware #: 1.00.05
Applic #: 1.0
View Security Policy

4-49999 | 4.x | PED | 30 Apr 2023

Company	Approval Number	Product Type	Version	Expiry Date	PIN Support	Key Management	Prompt Control	PIN Entry Technology	Functions Provided	Additional Information
SuperPED										
ACR900										
Hardware #: 4.00.xx Firmware #: 1.00.05 Applic #: 1.0 Approved Components:	4-40198	PED	4.x	30 Apr 2023	Online & Offline	TDES: Fixed,MK/SK,DUKPT AES: N/A	Acquirer-controlled	Physical Keypad	Display,ICCR,MSR,CTLS,PIN Entry,OP	Bluetooth is disabled in firmware. Enabling Bluetooth will affect the PCI PTS approval.

Example of PCI PTS listing

Under PCI PIN, each of these items should be checked, and in fact the auditor should download the Security Policy of the device and check the specifics of the key management to ensure that this is being implemented correctly. Use of key management in the application, using an offline only device for online transactions, having communications functions enabled that should be disabled, etc. All of these things would render an otherwise 'approved' PCI PTS non-compliant during deployment.

When purchasing POI devices, it's strongly recommended that your tender language does not just ask for PCI PTS approval, but specifies what you need the devices for, and the defines that level of compliance you need in each scenario. This can be complex to understand and write, seek advice where you need to – it can save you in the long run, from non-compliance or even at worst the cost of a breach due to the deployment of hardware assumed to be secure, but deployed in a way that that they are not approved.

… And We're Back to Offline Again

As PCI PIN concerns itself with the way in which PINs, and the cryptographic keys used to manage PINs, are handled it can actually mean that devices that only support Offline PIN use do not fall into scope of this standard. Such devices would not be loaded with keys that are used to secure PINs (Offline PINs are either sent in the clear, or are encrypted under the public key of the card being used), and as the PINs are not sent anywhere but the card, there is no PIN translation happening.

However, there are edge-cases where the decisions are not quite so clear cut. For example, if the offline payment device implements a 'cloud kernel' – where the EMV kernel is implemented in some remote server or 'cloud' system – then it's reasonable to question how Offline PINs are handled. Sometimes the PIN may be cached in the terminal for 'reinsertion' into the Verify APU as it is returned to the terminal, and sometimes *the Offline PIN may be transmitted Online* to the cloud kernel for remote creation of the Verify APU.

In such cases, an Offline only terminal would still fall under the requirements of PCI PIN, as the PIN is being transmitted externally from the device – and is being translated by the cloud system.

Section Take-Aways:

1) PCI PIN audits validate that the actual implementation of the key management practices used by the financial institution are compliant and secure.
2) A large part of the PCI PIN requirements cover documentation of process and procedure.
3) Understanding dual control and split knowledge is essential. Refer to the key management cheat sheet at the in Annex B of this book.
4) It is not enough to have PCI PTS approved devices, you need to check the hardware, firmware, and application versions, and if they are not deployed and used correctly.
5) Key management can get complex quickly. Seek advice to avoid non-compliance or lack of secure practices.
6) Although terminals that are only used for Offline PIN may usually be out of scope of PCI PIN, the use of Cloud (or remote) kernels can complicate this simple rule.

Most of the other standards we've discussed so far require backend, infrastructure, or hardware based components to be part of the overall assessment of a software product. However, particularly in the payment industry, there are many products that are composed solely of software. The assessment of these falls under the requirements of PA-DSS, and the PCI Software Security Standards which were newly released in 2019.

The Payment Application Data Security Standard (PA-DSS)

For many years, the PA-DSS has been the only formal security standard for payment software that could be applied separately to any payment solution. Different components of various other standards, such as PCI DSS and PCI PTS did cover some aspects of software for their own targets, but if you wanted an evaluation on a software only payment product, PA-DSS was the only game in town for some time.

The PA-DSS requirements are designed to be assessed against commercial payment software, to validate that software will not prevent PCI DSS compliance. The intent is not that installing PCI PA-DSS approved software will bestow compliance to PCI DSS, just that it won't break it. Therefore, the requirements validate the way in which the cardholder data is managed by that software, and how that software is to be installed and used in the customers environment. It checks the development processes used, the training of the development staff, etc to make sure that there is a sufficient level of security baked into the software that in and of itself it will not become a non-compliance during a PCI DSS audit.

However, over the years, the scope of PCI PA-DSS, how the requirements are set out, and how it is to be assessed has come under fire somewhat. Applying this process to agile development has proven difficult, and as both the development and threat landscapes have changed, PCI have created a new software standard to work with payments into the future.

PCI Software Security Standards (PCI S3)

The Software Security Standards were initially mooted during 2017 prior to the Community Meetings, and so their arrival should not be a surprise to anyone in the payment industry. However, what may be of interest (to both the payment and software development communities at large) is that there are actually two documents that have been released – the Secure Software Lifecyle (Secure SLC) Requirements, and the Secure Software Requirements and Assessment Procedures – as part of an overall software assessment framework, and understanding how these documents interact and work together to drive secure software development into the future.

The Source Codes Apprentice

So how did one software standard become two? Both of these standards are written to create what is referred to as the PCI Software Security Framework (that is, individually they are standards, and collectively they are part of the framework), and they address two very distinct aspects of software security.

The Secure SLC Requirements (SSLC) are designed to provide an assessment of the vendors software development processes, to ensure that security is 'baked in' throughout the process from design to on-going maintenance. This is an over-arching standard, not designed to be assessed against individual software products, but instead against the policy, procedures, and processes that are used to develop all (in-scope) software within that company.

The Secure Software Requirements and Assessment Procedures (SSRAP) document, on the other hand, ***does*** put forward requirements for the security assessment of individual software products. This standard considers individual aspects of software security for such products, and outlines testing procedures to validate those aspects.

Both of these standards also address the specifics of risk; why it's important to embed risk assessment into software development, what to include when you do this, and how the methodology used can be assessed. Expect 'risk based approach' to become the new hotness for PCI during 2019.

First, Know Thy Self

Unfortunately, secure software is not produced by accident – it requires diligence and expertise to ensure that the risks are understood and accommodated for. This is the purpose of the Secure SLC Requirements, to provide clear definitions around what processes a company must have to ensure they are able to produce secure software. The standard considers four high level aspects, with additional subordinate control objectives:

Software Security Governance

1) Security Responsibility and Resources
2) Software Security Policy and Strategy

Secure Software Engineering

3) Threat Identification and Mitigation
4) Vulnerability Detection and Mitigation

Secure Software and Data Management

1) Change Management
2) Software Integrity Protection
3) Sensitive Data Protection

Security Communications

4) Vendor Security Guidance
5) Stakeholder Communications
6) Software Update Information

Another way to look at these items is to consider that to develop secure software you should:

Know Who	Is developing your software and systems Is responsible for the security of each aspect Is involved in design, implementation, review, and release Is accountable for issues Your customers, integrators, and end-users are
Know What	Your product should do Your product should not do Software and hardware sub-components are included in your product Your relationships and reliance on third party suppliers is
Know Why	Understand and justify the reason for added features and code What interfaces will be used for Security is important to your customers and end markets All design decisions were made, and all code was used and implemented Any updates in code or subsystems is necessary
Know How	To develop secure software To correctly use the subsystems and sub-components of your product To check for new vulnerabilities in your systems and sub-components To update, or inform of an update to, your systems and products in post-sale

Know When	The length of time you will maintain your product post sale The length of time your subcomponents will be maintained To patch a vulnerability or to mitigate through other means

By validating the items above, it can be confirmed that the software vendor has a sufficiently documented, thorough, and robust process for any software it develops. Although this does not speak to the specifics of any individual software product, it does allow for some leeway in the way in which software updates are managed and assessed (but more on that later).

That SSRAP!

When we start to look at individual software products, we need different requirements to outline the specific security controls and considerations that must be included; this is the intent of the Secure Software Requirements and Assessment Procedures. However, given the diverse nature of software that may be covered by this standard, how does one document provide for the full breadth and depth of security required? The simple answer is it does not! The current SSRAP document is considered the start of a broader set of requirements that may be created to cover new software under this standard – it outlines the 'core' requirements of payment software, but this may be added to over time as specific aspects or features of software need to be considered.

Currently the SSRAP document provides for four high level aspects, once again with subordinate control objectives (and this time also an Appendix!):

Minimizing the Attack Surface

1) Critical Asset Identification
2) Secure Defaults
3) Sensitive Data Retention

Software Protection Mechanisms

4) Critical Asset Protection
5) Authentication and Access Control
6) Sensitive Data Protection
7) Use of Cryptography

Secure Software Operations

8) Activity Tracking
9) Attack Detection

Secure Software Lifecycle Management

1) Threat and Vulnerability Management
2) Secure Software Updates
3) Vendor Security Guidance
4)

Module A – Account Data Protection

A.1) Sensitive Authentication Data
A.2) Cardholder Data Protection

Looking at the above you can start to envisage the methodology the standard is taking; identify the assets and potential security issues, confirm that each of these items is mitigated or secured in some way through the software design, and then ensure there is an on-going program for maintaining the software over time as well as informing the users of the software on how to use it securely.

Interestingly, looking at the brief description above and the fact that there is then a separate module defining the specifics for "Account Data Protection" which defines the base assets in payment software, you may note that this standard is actually quite generic – in the sense that the requirements and tests could be applied to any software, not 'just' payment software.

Pay it Forward

This unbundling of the actual 'payment' aspect from the assessment standard itself is quite deliberate, and ensures that this standard can be modularized and applied to different types of software moving forward. Does this mean PCI is elbowing its way into non-payment areas? No, the intent is that we don't really know what 'payments' will be like 5 or 10 years from now, and vulnerabilities that exist in payment software can often be exactly the same vulnerabilities that exist in non-payment software. So, it makes sense to create this generic, overarching assessment *framework* which can be bent to the will of specific areas as required.

This has two main implications:

- It can be expected that there will be new modules and contexts added to this standard as required over time, or that the standard itself will be leveraged by other PCI standards into the future.
- If you're in the business of developing (or assessing) software, this remains a great tool to use to validate that you're doing the right things from a security point of view – even if the software you are developing is not specifically (or at all) payment related.

If you look at the different PCI standards, you see that there has traditionally been a somewhat disharmonious approach to software security – there are software security requirements in PCI PTS, PCI DSS, and a whole software assessment module in PCI SPoC. Although it's still early days, it's reasonable to expect that this software security framework (and now I'm talking about one or both of the current standards under this framework) may be applied to these areas to provide a more harmonious and consistent approach to software assessment throughout the PCI landscape.

Get with the Program

So, what to expect into the future for the PCI Software Security Standards? Those of you familiar with PCI may notice that of the three documents currently released, there remains a lack of a 'Program Guide' which is generally used to outline the management and process aspects of any new PCI standard. This can be expected to appear in due course, but some indications of what this may have inside already exist. In the FAQ document that ***has*** been released, PCI state:

> ***Q6 Who will be qualified to perform assessments to the Secure Software Standard?***
>
> ***A PCI-qualified Secure Software Assessor Companies and their employees. PCI Secure Software Assessor (SSA) Companies are independent security organizations that have been qualified by PCI SSC to validate payment software adherence to the Secure Software Standard. Secure Software Assessors are employees of SSA Companies that have satisfied and continue to satisfy all requirements of the SSA program.***

So, another PCI program will be produced with its associated accreditations and requirements for assessors. It's also noted in the FAQ that assessment of an organization against the SSLC requirements may be used to reduce the requirements for on-going delta assessments when changes are inevitably made to the software product over time – this is a great move, IMHO, and something that has been a point of discussion and contention in many of the other security programs I've worked in over the years. Full kudos to PCI for moving in this direction (although the first time this happened was actually in the PCI HSM standard … but that's for another day).

What does this all mean if you have PA-DSS approved software right now? Well, right now it does not mean that much – at the time of writing this program is yet to be released as noted. However, PCI are talking about phasing out the PA DSS program sometime into the future, as also noted in the FAQs:

> **New PA-DSS validations will be accepted through mid-2020 and be valid through late 2022. Assessments against the PCI Software Security Framework are anticipated to begin in Q3 2019 and will have a three-year validity period, putting the expiry date of those validations at roughly the same expiry date as PA-DSS validations.**

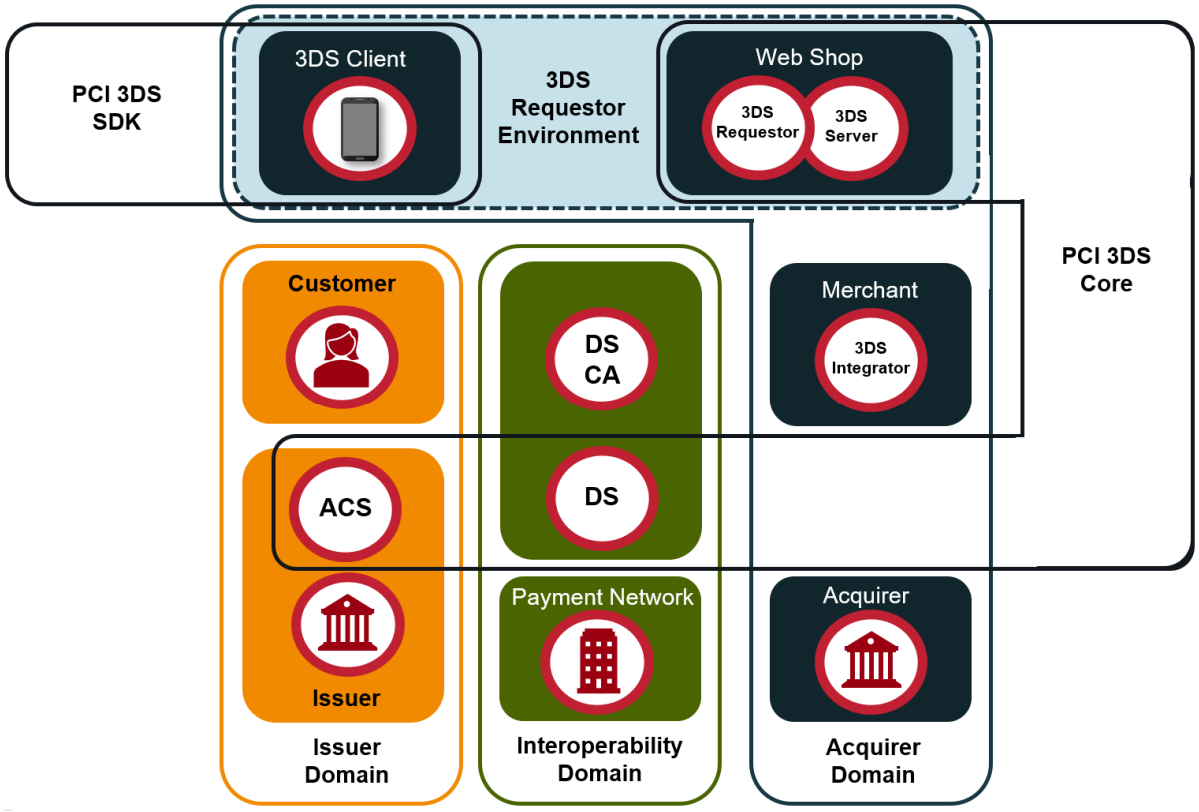
So, no reason to panic just yet. But given the general development times for software, the consideration of if you go with PCI PA-DSS or PCI Software Security is something that you should be undertaking right now if you have something you're working on but not yet released. Also, transition plans are worth considering as well at this point for existing software.

It's also worth taking note of the fact that PCI are increasingly talking about consolidation in their standards, and it can be expected that the Software Security Standard(s) will play a key role in this. So, even if you are not in the business of needing PA-DSS or other approvals, it may be worth taking note of what's included in these new software standards and considering how they may apply to your area of payments.

Section Take-Aways:

1) PCI PA-DSS is focused only on commercial payment software. It does not apply to bespoke software developed in-house, or to software that is not programmatically involved in the payment authorization process.
2) Applying PA-DSS to an agile development process can prove difficult.
3) The PCI Software Security Standards are designed to eventually replace PA-DSS, and consists of two separate standards; one for the Secure Software Development Lifecycle (SDLC), and one for the assessment of individual software products.
4) Compliance of your SDLC to the PCI Software Security Framework will mean reduced compliance burdens for changes to that software over its lifetime.
5) A transition from PA-DSS to the Software Security Framework will occur over the next three years.

As we learnt previously, a 3DS implementation has four main components, and each of these components are covered by one of two PCI testing requirements and regimens, as illustrated below.



The applicability of the PCI 3DS standards to the various domains of 3DS

The illustration above shows that the SDK component is in scope for the PCI 3DS SDK evaluation, and the other components are in scope for the 3DS Core requirements. These core requirements are actually split into two parts – Baseline Security Requirements, and 3DS Security Requirements. Below we'll go through aspects of each of these standards, and how they may apply to your environment or solution.

Pack Up Compliance in Your SDK

The 3DS SDK standard applies to the software components that are used to secure data at the customer end – usually this is through the utility of a 'Software Development Kit' (SDK) that is integrated into another software component (such as a phone app) that then allows for in-app purchases using 3DS protocols and security. The standard is divided into five high level objectives:

1) Protect the Integrity of the 3DS SDK
2) Protect Sensitive 3DS SDK Data Elements
3) Use Cryptography Appropriately and Correctly
4) Manage Risks and Vulnerabilities
5) Provide Guidance to Stakeholders

It can be seen that this takes a similar approach to other PCI standards – protect the system, protect the assets processed by that system, use appropriate cryptographic controls, have a vulnerability

management program. One item that is perhaps not as common across all PCI standards is the requirement for guidance – which is because as an SDK, the scope of assessment here is in fact not an end product, and so guidance on how to integrate and use the SDK is of paramount importance. This can be seen to be similar to the Implementation Guidance for a P2PE system.

Working on your Core

The 3DS Core – Baseline Security Requirements, provide – as the name implies – baseline technical and operational security requirements that are necessary to secure the 3DS operational environment. These requirements align with the control objectives of PCI DSS, and can be broadly seen as a translation of PCI DSS into the 3DS realm, where the customer PAN may not be present to otherwise create a need for PCI DSS assessment. There are seven high level objectives:

- P1-1: Maintain security policies for all personnel
- P1-2: Secure network connectivity
- P1-3: Develop and maintain secure systems
- P1-4: Vulnerability management
- P1-5: Manage access
- P1-6: Physical security
- P1-7: Incident response preparedness

The 3DS Core – Security Requirements then cover the actual 3DS data that may be processed within the 3DS infrastructure during a transaction. These requirements also have seven objectives.

- P2-1: Validate scope
- P2-2: Security governance
- P2-3: Protect 3DS systems and applications
- P2-4: Secure logical access to 3DS systems
- P2-5: Protect 3DS data
- P2-6: Cryptography and Key Management
- P2-7: Physically secure 3DS systems

PCI additionally provides a 'data matrix' that details each of the data elements and the levels of protection that it requires, which includes if that element is permitted to be stored, and for some elements if that storage is only permitted temporarily, or even if it must be managed within a HSM. It is this data matrix that forms an essential part of the validation of 3DS scope, as the PAN is no longer the primary value that must be protected.

One way to look at the 3DS Core standards is that the Baseline Security Requirements exist to cover the infrastructure on which the 3DS application is executed – the servers, networks, etc. – and the Security Requirements themselves are really designed to be applied to the actual application and processing performed as part of the 3DS transaction.

Relationship Between Core and PCI DSS

So if we say that the 3DS Core standard is similar to the PCI DSS standard in many ways, how do these two standards interact (if at all)? Ultimately, there's nothing in PCI 3DS Core that forgives or removes the necessity of compliance against PCI DSS – if you have PAN data, PCI DSS is in scope. However, where the Core standard comes in is when you don't have PAN data, but you do have sensitive 3DS assets.

Without PAN, there's no soup for you in regards to compliance against PCI DSS, so an additional standard is required to ensure that the security of these is considered and correctly assessed. That's where Core comes in.

What if all of your 3DS assets are within your PCI DSS Cardholder Data Environment, and covered under those controls already? Well, just as Core does not forgive PCI DSS compliance, PCI DSS does not obviate the need to confirm compliance to the Core standard. However, if all of the controls are correctly in place and protecting the 3DS assets as well as the PCI DSS assets … it should be a much easier process to validate against 3DS Core. The main hurdle will be confirming that scope and coverage are entirely correct.

Section Take-Aways:

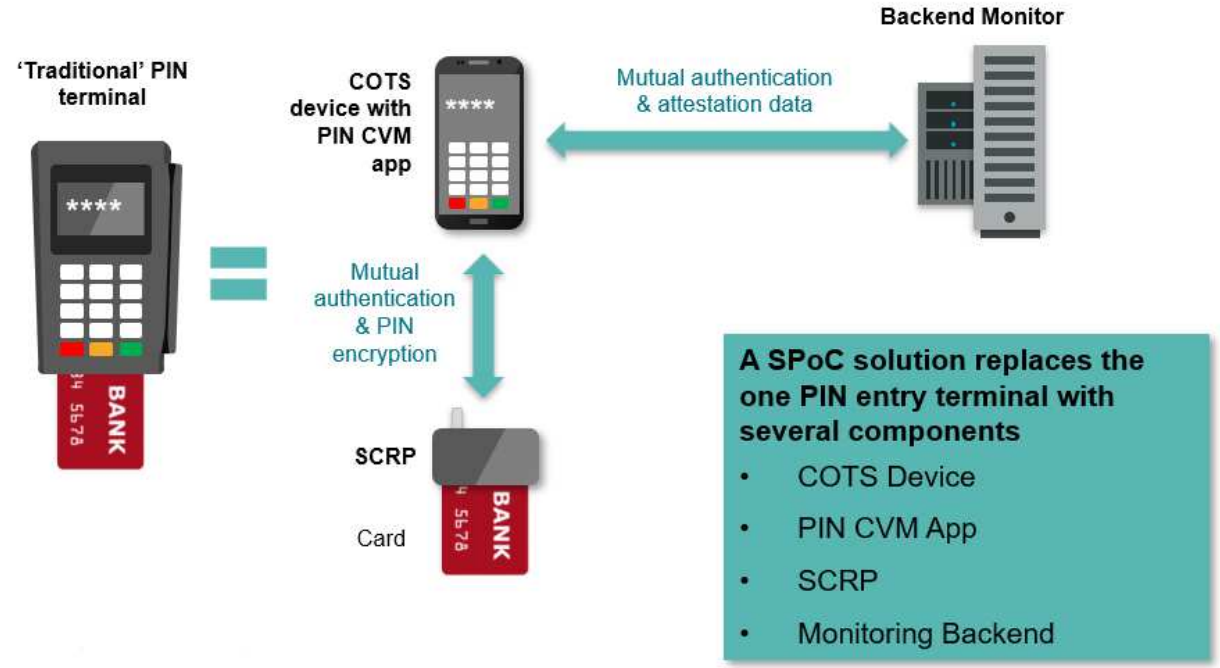
1) PCI 3DS scope is not defined solely by the presence of the PAN.
2) SDK evaluations are on a software object that is to be integrated into other end-use systems such as phone apps.
3) Implementation guidance is an important part of securing a 3DS SDK.
4) There are many similarities between 3DS Core and PCI DSS, excepting the scope.

Throughout this document we've often mentioned the need for secure hardware for entry of PINs and securing cryptographic keys. This is not always the case – use of an SCD for handling keys is not mandated for PCI DSS – and recently there has been a shift in the way in which PIN entry may be performed. This shift allows for entry of PINs on consumer grade systems, which provide no hardware security features, and is known as PCI Software PIN on COTS (SPoC).

COTS – What's in a Name?

The term 'COTS' stands for 'Commercial Off The Shelf', and is used to refer to devices and products which are designed for 'mass market' deployment, that have not been modified, designed specifically, or made bespoke. In the context of this new standard, it can be interpreted that 'COTS' means that the PIN entry process is not performed on a dedicated 'PINPad' that is specifically designed for that purpose. Therefore, PIN on COTS is literally a standard for 'PIN entry on devices which are not PINPads'; indeed, the examples given in the standard show merchants implementing this standard to allow for PIN entry on their own mobile phones.

This may seem a little strange, even fundamentally insecure, at first blush. However, it is of course not quite that simple and although a dedicated PINPad is not used, there are methods and systems required within the PIN on COTS standard to provide security to the PIN entry process. Indeed, as outlined later, a PIN on COTS system relies on the interaction of multiple components to secure the customer PIN, rather than the 'normal' PTS approved terminal relying on its own tamper responsive features alone.



SPoC transplants the traditional security of a POI with four different components

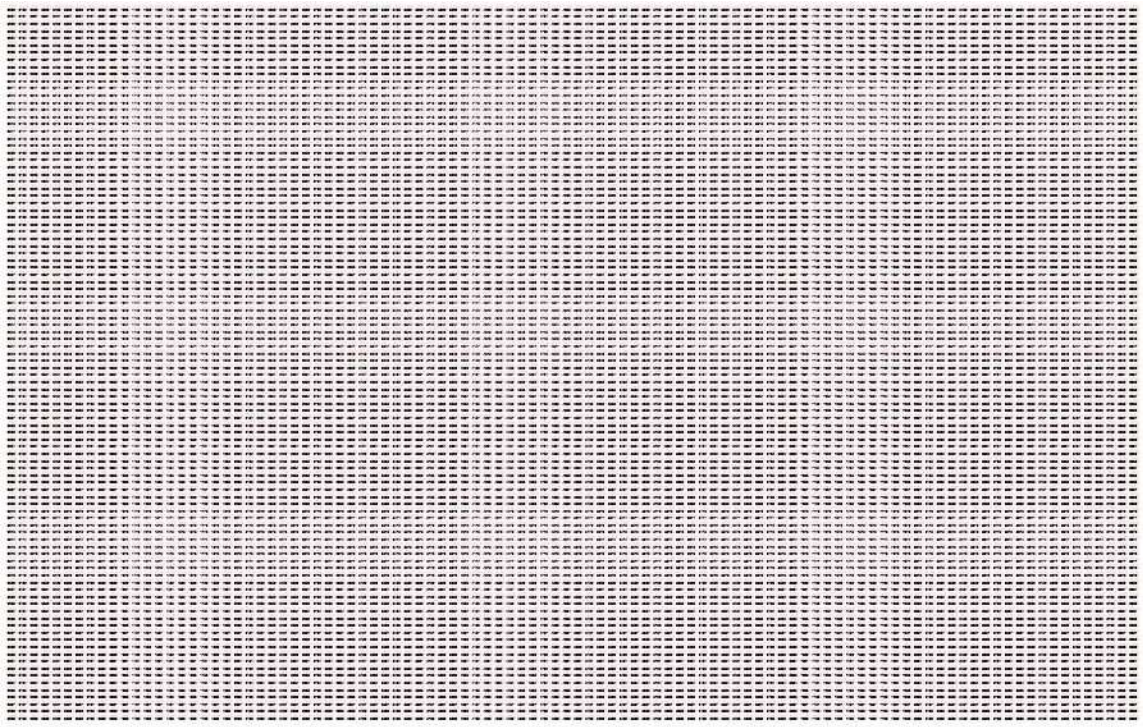
Before going into these, however, it's important to consider that this new standard is not called 'PIN on Mobile', even though that seems to be the use case implied by the standard itself. This is an important, if subtle, distinction. 'Mobile' is a use case, rather than a type of technology, and through the use of the term 'COTS' this standard does not appear to constrain itself to any specific use case: it is not a 'mobile

payment' standard, although it will most likely be seen that way (at least initially). It is a standard for replacing dedicated PINPads with non-dedicated devices, regardless of exactly how those non-dedicated devices are implemented in the field.

I Know Your PIN …

One of the paradoxes of PIN security has always been the reliance on what is essentially just a four digit decimal number as a security and authentication feature. Of course, technically, a PIN can be anywhere from four to twelve digits in length (as defined in ISO9564), but most commonly it is implemented and used as a four digit number. You can list out all of these numbers, from 0000 to 9999 … in fact, in the table below we have done just that. If your PIN is four digits long, and it likely is, then ***your PIN is listed below*** (although to be fair it may be a bit small to see).

I know your PIN.



List of all possible four digit PIN values

Of course, this information provides me with no benefit unless we *also* know your card details. Otherwise, your PIN remains 'just' a four digit number. This is the fundamental idea that underlies the new PCI PIN on COTS standard – separating the customer card data from the customer PIN, so as to 'devalue' the PIN. This is done through requiring the card data to be entered into a 'SCRP' – or Secure Card Reader: PIN – which is a new approval class under the existing PCI PTS program that outlines the incumbent requirements for PINPad devices.

Chipping Away at Fraud

Additionally, the requirements call for the PIN entry only to be accepted when the transaction is EMV (or 'chip') based. This is because, as noted earlier, the 'chip' in an EMV card contains secret

cryptographic keys that are used to uniquely authenticate and identify EMV cards and transactions performed with that card. Unlike magnetic stripe cards, EMV cards cannot be easily cloned (I use the weasel word 'easily' here, as breaking things is kind of my job; very little is impossible, but it's certainly hard enough that you'll spend more than you make out of breaking any specific card). This essentially prevents card-present fraud with EMV. This reduction in fraud is backed up by statistics in countries which have gone through EMV deployment; it's a fact.

This fact is the pivotal piece of information necessary to understand the concept of the new PCI PIN on COTS standard. Traditional PIN security programs have been about protecting the customer PIN, because the customer PIN is the only secret that secures the transaction; if a criminal knows your PIN and card data, they can create a cloned magnetic stripe card and use that to perform fraudulent transactions. In this way, PINs can be seen as a solution to a magnetic stripe problem – and we are no longer in a magnetic stripe world. We live in an EMV world, or at least a world rapidly reaching saturation of EMV deployment. In this EMV world, the PIN is not the only secret anymore because we have the secret cryptographic keys on the customer card; even if you have a customer's card data (as output from the EMV card during a transaction) and the associated customer PIN, you can't perform an EMV transaction.

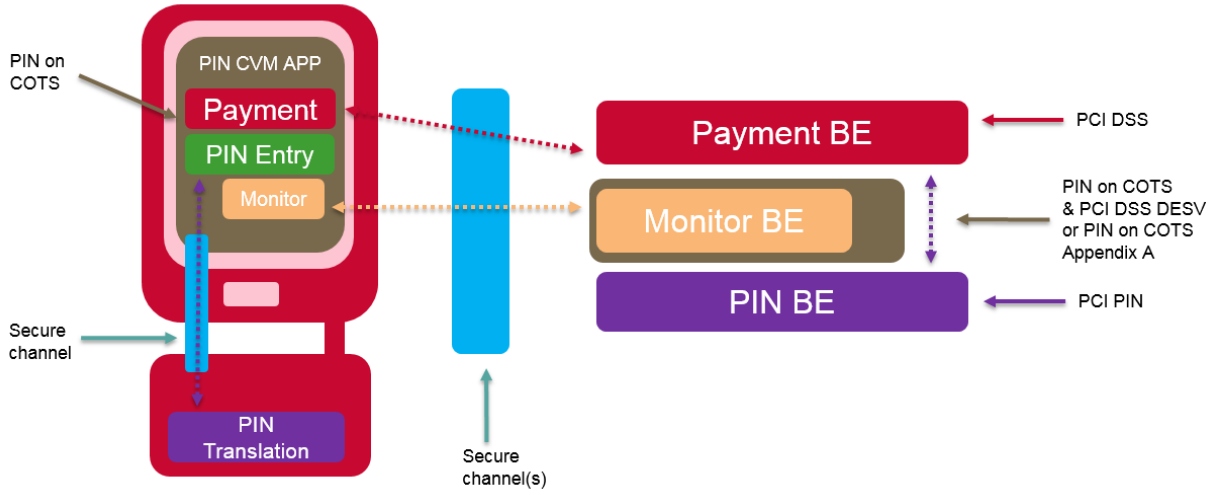
At UL, we've talked about similar issues before; about how the changing technology of EMV, phone-based Issuance (Apple Pay, Google Pay, etc), and advanced authentication methods are changing the way we pay, changing the way fraud is performed, and ultimately changing the need for PINs. This new standard produced by PCI is essentially a stepping stone from a world dominated by PINs and magnetic stripe cards, to a world where different payment form factors and different authentication methods are used. (Note as of the time of writing PCI has an RFC out for an addendum to SPoC to allow for magnetic cards to be used with these solutions – just not with PIN, so this maintains the core idea of separation between PIN and magnetic stripe card data that is the foundation of SPoC).

PIN on COTS Components

So how does this all work? The PIN on COTS standard calls for four main components for any solution, and those components must all work together in tandem to secure the overall transaction and PIN entry:

1) The PIN on COTS application. This is the component that resides on the merchant COTS device (phone/tablet) that is used to accept the customer PIN.
2) The 'monitor' system. This component is actually split between the merchants COTS device and a back-end system. The monitor is used as a mechanism to constantly check the 'health' and security of the COTS device that is being used. You can think of this as the software equivalent of the tamper detection mechanisms that normally reside in a PINPad, but instead of checking for physical tampering, the monitor constantly checks for 'software tampering' of the application or COTS device used.
3) The Secure Card Reader: PIN (SCRP). As noted, this is a device that must be tested under the existing PCI PTS program, and it is used to accept and encrypt the customer card data before it is sent to the merchant COTS device. This is how the separation between the card data and the customer PIN is achieved.
4) Payment and PIN processing backends. These areas are the 'standard' payment processing areas that would exist in any normal PIN based transactions.

These components are illustrated in the picture below, with the areas in scope of the new PCI PIN on COTS standard outlined in the brown boxes.



The various aspects and connections of a SPoC solution

The transaction process is then performed as follows:

a) The customer presents their card to the SCRP. As noted above, this card must be an EMV based 'card', but may be a contact or contactless chip card, mobile payment mechanism, or other form factor.
b) The card data is read and encrypted by the SCRP, which will also perform the EMV processing of this data.
c) The customer enters their PIN into the PIN CVM application on the merchant COTS device. This PIN is encrypted on that COTS device, and sent to the SCRP; where it is either sent directly to the customer card for validation (if it is an 'offline PIN' transaction), or it is 'translated' to encryption under the cryptographic keys shared with the PIN processing backend.
d) The transaction, including the encrypted online PIN as 'translated' by the SCRP (if it is an online PIN transaction), is sent to the payment processing backend.
e) The transaction is authorised (or rejected) by the payment processing backend as per normal.

Throughout this process, the monitor system will also be constantly checking the security of the COTS device and merchant application.

Online offline, or offline online?

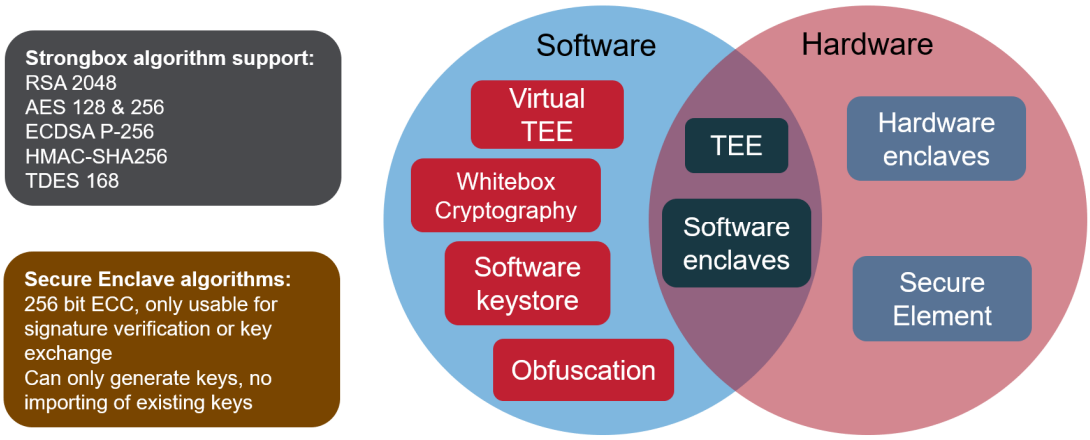
An important item to note is that the transaction ***must*** be processed online. The customer PIN cannot be accepted by the COTS application unless the merchants COTS device has connectivity through to the backend(s). This does not forbid the use of 'offline PINs' however – an offline PIN transaction may still be processed online. They're different.

It would also be possible for a PIN on COTS system to accept different authentication methods – signature, Consumer Device Cardholder Verification Method (CDCVM – such as biometrics on the

cardholders phone), etc. PIN use is not mandated, but it is the focus of the security aspects of the new standard.

A History of Violence

Consumer grade 'mobile' systems (phones, tablets, etc) have been getting more secure with each iteration, as researchers have found vulnerabilities and the OS vendors have patched and updated their systems to mitigate these. The new processors have support for some impressive security features (such as pointer authentication), and the operating systems are under a constant state of improvement in regards to security. Both major mobile operating systems now support hardware security processors ('Strongbox' for Android, and the 'Secure Enclave' for iOS), and these provide significant security protections to the cryptographic keys that applications may store and use.



Securing assets on COTS systems can include both software and hardware methods

Of course, hardware is not the only way to secure assets, and security can be provided by 'just' software, or a combination of software and hardware, as well as these dedicated security processors. The diagram above illustrates the common methods for protecting assets in a COTS system, as well as the cryptographic support provided by the hardware implementations. Software methods provide the broadest platform and cryptographic support, but often with reduced security.

Section Take-Aways:

1) PCI Software PIN on COTS allows for PIN entry on consumer grade devices.
2) SPoC compliance requires the use of an SCRP to accept and encrypt card data.
3) The security goal of SPoC is to separate the PIN and cardholder data using cryptography, but where the cardholder data is protected by a PCI PTS device rather than the PIN.
4) SPoC requires the use of a backend monitoring/attestation service that can be quite complex to implement and manage. SPoC is not a 'cheap way to accept PINs'.
5) SPoC requirements are still developing – recent changes may see the allowance of MSR transactions with a SPoC system (but without PIN entry).

PCI have publicly announced that they are working on a new standard that will cover the acceptance of contactless cards directly on consumer grade devices (COTS devices like mobile phones). At the time of writing, this standard is yet to come out, so covering the details of this is not possible right now (look for the next revision of this book once the standard does come out!). However, we can discuss the technical aspects of how this may work, and also the context of different brand standards that already exist covering this mode of acceptance right now.

Most modern mobile phones and tablets include an NFC interface – this has been common since mobile payments started gaining traction, and the inclusion of an NFC interface is now mandatory with Google certified Android phones and with all new iOS devices. The use of this has usually been aimed at mobile payments by the owner of the phone (using that interface as a payment mechanism), as a data transfer function, or for other card replacement operations such as mobile ticketing for transport. However, the nature of NFC is that it can behave both as a 'card' emulation device, but also as a contactless reader.

This means that a phone or tablet can also become a contactless payment terminal, accepting contactless cards for payment without the need for any external hardware. Put another way, with a compatible mobile phone, anyone can become a merchant and start accepting contactless payment transactions by simply downloading an app from the relevant app store of their phone provider. This opportunity has many people quite excited by the possibilities of contactless acceptance on COTS devices.

However, of course, it's not necessarily that simple. We discussed the need for EMV L1 testing for contactless payment terminals, and that the contactless volume required for that testing – the area above the terminal contactless acceptance mark where the card will be powered by the terminal RF field and perform the transaction – is around 5 cm. For most, if not all, mobile phones this volume is ***much*** smaller – often less than 1cm.

This greatly impacts the user experience during the contactless transaction, literally meaning that a card that reads correctly when facing one way will fail when rotated around 180 degrees and placed in exactly the same location (as illustrated above).

Damn it SPoC, I'm Not a Miracle Worker!

Fundamentally many of the security issues faced by a Contactless on COTS solution is similar to that of a SPoC solution – the fact that the platform is untrusted, the lack of tamper responsive systems within the hardware of COTS devices, etc. However, there are also some fundamental differences that make contactless acceptance a potentially more thorny problem to solve.

With SPoC it is possible to rely on the SCRP as a trust anchor for the system; it provides the secure hardware and approved cryptographic operations, and it allows for a physical touch point with the merchant during the on-boarding process. Having the SCRP also improves the ability to correlate potential compromise data between different COTS devices which may share the same reader, as well as providing the ability to force disable card reading if things are found to be too suspect.

None of this is available with a contactless on COTS solution that does not have any reliance on external hardware. How then is can such a solution be secured? One advantage that a contactless on COTS solution does have over SPoC is that we're not dealing with PINs, which changes the threat models quite significantly. We already accept payment cards through non-SCD devices all the time in card present payments, so maybe the real question is why do we need any additional security at all?

The end answer is somewhere is, and is expected to be, somewhere in between. I say "is, and is expected to be" because we already have standards for contactless on COTS from some of the card brands, but we do not yet have the PCI contactless on COTS standard. Therefore we're left to extrapolate from both the brand standards and the existing PCI SPoC standard. Ultimately, it's reasonable to expect that a solution will still require both a COTS app and a backend monitoring/attestation solution – with the attestation helping to mitigate against the lack of patching and security controls on the COTS systems that would otherwise be required in a computer or POS accepting contactless payments.

This book will be updated once the PCI Contactless on COTS standard (to be called 'CPoC' – Contactless Payments on COTS) is released with full details and assessment of that standard.

Take Me to Your Kernel

One of the problems faced by COTS based contactless systems is that the NFC interface is often designed to be routed through the main Operating System (often called the Rich Execution Environment, or REE). At least this is the case with Android, and that is currently the only system where sufficient access to the NFC interface is possible to implement contactless payments acceptance.

This means that even if you implement some aspects of security in a TEE or security processor, the card data is still routed through the main Operating System. Realistically, the level of security in modern OS's are sufficient to make this less of a concern – especially in light of the security inherent in EMV contactless transactions – but it is something that needs to be considered when deploying contactless on COTS systems.

Section Take-Aways:

1) Contactless interfaces on COTS devices generally provide a much smaller contactless volume for operating and reading a contactless card. This can affect the user experience.
2) The lack of secure hardware in a contactless on COTS solution creates new challenges for securing these implementations.
3) KYC for merchant on-boarding is also further complicated by the ability to become a merchant through simply downloading an app.
4) However, contactless solutions operate in a different threat landscape where PINs are not used, reducing the risk of compromise.
5) A monitoring/attestation system is still required to compensate for the lack of controls on the COTS devices used.

The implications and implementation of tokenization systems have already been discussed previously, at least from an operational point of view. Tokens can be used to reduce PCI DSS scope, or to provision mobile payment applications so that they don't have the 'real' (of 'funding') PAN being used, increasing the security of the system. How do these systems work, and what security requirements apply?

The Format is the Message

The primary need for any payment tokenization system is to create, deliver, and manage tokens for payment use. This can be done in one of two different ways; providing a token that is formatted in the same way as a PAN so that it appear to all intents and purposes to be a PAN, or to supply a payment reference token that is entirely different to a PAN and used in different ways.

The creation of tokens for payments also has two different types, when viewed at a high level; referenced token values, and calculated token values. A referenced token value is one that is simply assigned and is referenced to the funding PAN within some table or database. In this scenario the token may be randomly generated or defined by some algorithm or counter, may be information bearing in some way, may be plaintext or encrypted.

The foundational aspect of a referenced token is that to reverse the token, or to determine what that token references in terms of the funding PAN or specific transaction data, a look-up in some 'token vault' is required. This token vault may be a databased, flat file, spreadsheet, or any form of referenceable data storage mechanism that allows for a funding PAN value to be assigned to a token. There must be one entry for each token, allowing for it to be checked as needed.

A calculated token is different. Here the token is calculated from the input data, which may be just the funding PAN, or may include transaction details, merchant details, etc. Generally, a calculated token requires the use of cryptography or hash functions to secure the calculation process, which may be applied in a way to ensure that the value output from the cryptographic process actually 'looks' like a PAN – this is an application of a specific type of cryptography called 'format preserving encryption' (FPE).

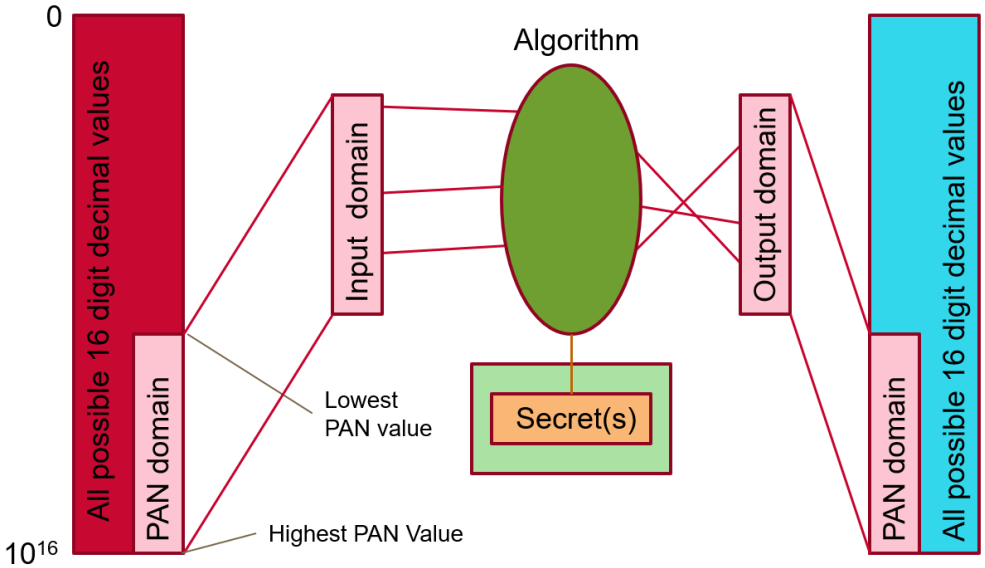


Illustration of a generic format preserving PAN token system

PCI take the concepts of a token a bit further by further dividing them into 'high value' tokens and 'low value' tokens. Here the difference is that high value tokens are essentially reusable payment card values – something that can be used for an arbitrary number of different transactions. A low value token is instead limited, generally to be used just once, and cannot be processed through a 'normal' payment card purchase process.

For PCI DSS purposes, a high value token must be treated in the same way as any normal payment card data, as from an operational point of view it is exactly the same. Low value tokens can be used to descope areas from PCI DSS assessment, where those areas only have access to the token, and have no access to detokenize or reference the token back to the funding PAN.

(Not Just) A Token Effort

The security of the tokenization system itself, and how that is assessed, depends also on the type of token and the implementation of the tokenization system. Not all tokenization systems require reversion back to the funding PAN – sometimes the ability to validate the details of the transaction without the PAN, or to be able to validate that any given PAN value was involved)but not to actually be able to reverse the token to determine what PAN was used) is enough. In these systems, security may be somewhat reduced or even considered out of scope from a storage point of view – although the systems that have access to the original funding PAN during the process of generating the token are always in scope of PCI DSS, of course.

However, many tokenization systems either allow for reversion to the funding PAN, or produce high value tokens that are in themselves targets for compromise. In these situations, the tokenization system must be protected and secured against compromise or misuse. PCI have a standard for 'Token Service Providers' (TSP) that contains additional requirements for PCI DSS assessments against such entities.

The scope here is for entities that produce high value tokens that are essentially replacement PAN values that can be processed and managed normally through a merchant payment system, even when that system is not 'aware' it is processing a token. Scope for the PCI TSP requirements does not include merchant owned tokenization systems, only those run by third parties for other entities, as discussed in the mobile payment section of this book.

The requirements in the PCI TSP standard are focused on supplementing the existing PCI DSS requirements for those that affect the 'Token Data Environment' as follows:

- TSP 1: Document and validate PCI DSS scope
- TSP 2: Secure TDE systems and Network
- TSP 3: Protect and manage cryptographic keys
- TSP 4: Restrict access to the TDE by business need to know
- TSP 5: Identify and authenticate all access to the TDE systems
- TSP 6: Restrict physical access to the TDE
- TSP 7: Monitor all access to the TDE
- TSP 8: Maintain an Information Security Policy

From the above, you can see that this does not deviate from PCI DSS very much at all and much of the standard is focused on ensuring that the TDE is segregated from other parts of the network and system, and that sufficient controls are in place for access to the TDE and TDE systems such as HSMs.

But if these requirements are only scoped for a TSP, what are other entities to do when it comes to securing their tokenization systems? Best practice is to follow the TSP requirements as much as you can, and PCI also have some best practice references to help people understand the security controls that may be most suitable. Overall, the primary rule is segregation of the token system from the card data systems, and ensuring that any de-tokenisation process is secured.

This is where cryptographic tokens tend to win over referenced tokens – with cryptography you can often isolate the token system down to a single HSM which has been validated through other programs such as PCI HSM or FIPS140-2 (although do take care in understanding the scope of any existing approval to make sure you know what you're getting!). A referenced system requires storage of the token and PAN data in some referenceable form, and securing that storage can be difficult.

Section Take-Aways:

1) A token is used to replace details about a transaction, generally to remove the need to store and process the 'funding' PAN value used in that transaction.
2) Tokens can be generated as a reference value or calculated using a defined algorithm (usually involving cryptography).
3) Not all tokens must be reversable back to the funding PAN value.
4) A token may appear like a 'normal' PAN so that it can be processed with systems that are not designed for use with token values. Alternatively tokens may be values or data that are quite different to a PAN value.
5) A 'high value token' may be used for on-going purchases. Often such tokens are formatted as PAN substitutes, but this may not always be the case. High values tokens remain in scope of PCI DSS assessments.
6) A 'low value token' cannot be used for other transactions.

The card production requirements are divided into two parts, addressing separately the physical and logical requirements. These requirements are notoriously strict, with the physical requirements going into perhaps painful detail on what is and is not allowed, up to and including if you can have a toilet in your card production area!

(The answer is no, by the way, unless there is an overriding law about toilet accessibility and then you need video monitoring of the toilet entrance at all times – refer requirement 3.3.5.b)

The physical requirements also cover the security of card stock as it is received, processed, and shipped back out. Specific security areas are defined, including a High Security Area (HSA) where the actual card stock is personalized and stored during that process.

Logical requirements cover data classification requirements, and how specific classifications of data should be secured in terms of network security, key management, remote access, etc.

So, who what types of organizations are in scope for the card production requirements? Well, companies that produce payment cards, of course, traditionally called 'Personalization Bureaus' as they take generic card stock and 'personalize' this to the individual Issuing bank and customer. But what about companies that are managing the digitization process for mobile payments? In the PCI Card Production Logical Security Requirements, in the section on scoping, it notes:

> This document describes the logical security requirements required of entities that:
>
> - Perform cloud-based or secure element (SE) provisioning services;
> - Manage over-the-air (OTA) personalization, lifecycle management, and preparation of personalization data; or
> - Manage associated cryptographic keys.

Which therefore does include in scope companies involved in the digitization process for payment cards used in mobile payment implementations.

Section Take-Aways:

1) The card production standards contain both physical and logical requirements.
2) Both the physical and logical requirements are quite strict and proscriptive.
3) Card production requirements apply to personalization Bureaus that work with physical cards, and token service providers that issue digital cards for mobile payments.

In this book we've discussed just some of the many different ways in which payment can be processed during both card present and card not present transactions, and the standards that apply to those types of payments. In the next revision and update of this book, we'll dive a bit deeper into some of the standards we've already covered, as well as looking at some of the new areas of payments such as Open Banking and EMV Secure Remote Commerce (SRC). By the end of this year (2019) we'll likely have the new PCI CPOC contactless standard, as well as more detail on the programs around PCI PIN and PCI Software Security to discuss.

So, more remote payment and more card-present payment.

However, these concepts of defining a transaction based on the physical location of the customer card are become outdated; not only is the term 'card' itself an obsolete term, the idea that the physical presence of such a thing is important is simply false. Is a Host Card Emulated (HCE) system 'card present', when the data is obtained from a remote cloud system? Is an online payment 'card not present' if the customer is using their mobile phone web browser which has the ability to access and use their stored card details?

As we move forward into the next chapters of payments, we face a reality where we may only care about 'customer present' or 'customer not present' payments, and even then that may not mater too much. The existing cannons of payments – e.g. that PINs must only be processed in Secure Cryptographic Devices – are being challenged and knocked down.

I don't expect the payment industry to stop existing, but I do expect that it will disappear; from something that is obvious, to something that is invisible to most. For a long time, we've expected things to revolve around how we have built and designed things dedicated to payments. Dedicated payment terminals were designed to accept dedicated payment cards, to be processed with dedicated payment software, and sent to dedicated payment processing systems. What we're seeing right now is a change, perhaps for the better, in which payments is becoming commoditized – rather than its own vertical, payments is becoming a horizontal that underlies all other commerce which is probably what it should have been all along.

In many ways, payments has stood still for a long time. That time is over, we're moving again and gaining speed fast. Understanding both where we are now, and what's may be coming into the future is more important than ever for everyone in the payment industry as we take the journey to what comes next.

All aboard.

Andrew Jamieson

Format 0

The format 0 PIN block is the most widely used PIN block format.

0	N	P_1	P_2	P_3	P_4	P_5/F	P_6/F	P_7/F	P_8/F	P_9/F	P_{10}/F	P_{11}/F	P_{12}/F	F	F

XOR

0	0	0	0	A_1	A_2	A_3	A_4	A_5	A_6	A_7	A_8	A_9	A_{10}	A_{11}	A_{12}

Encrypt to produce encrypted PIN block

Where:
N = Number of PIN digits
P = PIN digits 1 – 12 (minimum of 4 digits in any PIN block)
F = Filler value of hexadecimal 'F' (binary '1111')
P_x/F = PIN value or filler if no PIN value is provided at that location
A = Last

Format 1

Format 1 PIN blocks are deprecated for new use, and cannot be translated to from other PIN block formats, due to security concerns over the lack of the customer PAN in the format.

1	N	P_1	P_2	P_3	P_4	P_5/T_1	P_6/T_2	P_7/T_3	P_8/T_4	P_9/T_5	P_{10}/T_6	P_{11}/T_7	P_{12}/T_8	T_9	T_{10}

Encrypt to produce encrypted PIN block

Where:
N = Number of PIN digits
P = PIN digits 1 – 12 (minimum of 4 digits in any PIN block)
T_x = Transaction digit of hexadecimal '0' – 'F'.
Transaction digits should be uniquely tied to the transaction in which the PIN was entered.
P_x/T_x = PIN value or transaction digit if no PIN value is provided at that location

Format 2

Format 2 PIN blocks must be used only for offline PIN verification, where the PIN is transmitted to the customer card.

2	N	P_1	P_2	P_3	P_4	P_5/F	P_6/F	P_7/F	P_8/F	P_9/F	P_{10}/F	P_{11}/F	P_{12}/F	F	F

Format 2 is not encrypted

Where:
N = Number of PIN digits
P = PIN digits 1 – 12 (minimum of 4 digits in any PIN block)
F = Filler value of hexadecimal 'F' (binary '1111')
P_x/F = PIN value or filler if no PIN value is provided at that location

Format 3

Format 3 PIN blocks are the most secure of the TDES PIN blocks, including both the PAN and random data, but also are the least used.

3	N	P_1	P_2	P_3	P_4	P_5/R_1	P_6/R_2	P_7/R_3	P_8/R_4	P_9/R_5	P_{10}/R_6	P_{11}/R_7	P_{12}/R_8	R_9	R_{10}

XOR

0	0	0	0	A_1	A_2	A_3	A_4	A_5	A_6	A_7	A_8	A_9	A_{10}	A_{11}	A_{12}

Encrypt to produce encrypted PIN block

Where:
N = Number of PIN digits
P = PIN digits 1 – 12 (minimum of 4 digits in any PIN block)
R_x = Random value of hexadecimal 'A' – 'F'
P_x/R_x = PIN value or random value if no PIN value is provided at that location

Format 4

Format 4 PIN blocks are the most recent PIN block format, using AES encryption rather than TDES (although other 128 bit block ciphers *may* be used). They also require two AES encryptions; one of the intermediate PIN block value, the encrypted output of which is then XOR'd with the PAN block prior to the final AES encryption to create the encrypted PIN block. From v5.1 of PCI PTS, all POI devices must support the use of format 4 PIN blocks.

4	N	P_1	P_2	P_3	P_4	P_5/F	P_6/F	P_7/F	P_8/F	P_9/F	P_{10}/F	P_{11}/F	P_{12}/F	F	F	R_1	R_2	R_3	R_4	R_5	R_6	R_7	R_8	R_9	R_{10}	R_{11}	R_{12}	R_{13}	R_{14}	R_{15}	R_{16}

Encrypt to Encrypted Intermediate PIN Block

XOR Encrypted Block with

M	A_1	A_2	A_3	A_4	A_5	A_6	A_7	A_8	A_9	A_{10}	A_{11}	A_{12}	$A_{13}/0$	$A_{14}/0$	$A_{15}/0$	$A_{16}/0$	$A_{18}/0$	$A_{18}/0$	$A_{19}/0$	0	0	0	0	0	0	0	0	0	0	0	0

Encrypt to produce encrypted PIN block

Where:
N = Number of PIN digits
M = Number of PAN digits minus 12 (so, M=0 for 12 PAN digits, M=7 for 19 PAN digits)
P = PIN digits 1 to 12 (minimum of 4 digits in any PIN block)
R_x = Random value of hexadecimal '0' – 'F'
P_x/F = PIN value or hexadecimal 'A' if no PIN value is provided at that location
F = Hexadecimal 'A'

EMV Encrypted PIN Format

The EMV encrypted PIN block format is used for offline PIN verification where the card supports transport of the PIN in an encrypted format. It is based on an extension to the format 2 PIN block, adding random data from the terminal to ensure that even if the same PIN and card are used more than once, the encrypted PIN blocks will appear different.

H	PB	UN	RP

Encrypt with EMV card PIN encryption public key

Where:
H = Block header of hexadecimal '7F'
PB = Format 2 PIN block (previously calculated as above)
UN = EMV card unpredictable number
RP = Random padding as applied by the payment terminal to the end of the block

PIN Block Translation Rules

The table below summarizes the requirements for PIN translation from one PIN block format to another. This is primarily used in a HSM for translation during transmission of the PIN, but may also be used when transmitting the PIN from one device to another in another context – such as between a PIN entry device, and an external card reader.

Incoming PIN block format (Translation from)	Output PIN block format (Translation to)				
	Format 0	**Format 1**	**Format 2**	**Format 3**	**Format 4**
Format 0	PANx → PANx PANx → PANy PANt → PANf	Not Permitted	For Offline PIN	Permitted	Permitted
Format 1	Permitted	Tx → Tx	For Offline PIN	Permitted	Permitted
Format 2	Not Permitted	Not Permitted	N/A	Not Permitted	Not Permitted
Format 3	PANx → PANx PANx → PANy PANt → PANf	Not Permitted	For Offline PIN	Permitted	Permitted
Format 4	PANx → PANx PANx → PANy PANt → PANf	Not Permitted	For Offline PIN	Permitted	Permitted

Where:

PANx → PANx = Translation between different cryptographic keys, keeping the same PAN

PANx → PANy = Translation changing the PAN value, may use different or the same cryptographic keys (This is only permitted with the HSM in a sensitive state for card Issuance purposes)

PANt → PANf = Translation between tokenized PAN to funding PAN, may use different or the same keys

Tx → Tx = Translation between different cryptographic keys is permitted, but transaction digits within the PIN block should not change

For Offline PIN = Permitted to translate an encrypted PIN to format 2 within a card reading device for direct transmission of the PIN block to the EMV card for offline PIN verification only

Permitted = Translation between these formats is permitted in all scenarios

Not Permitted = Translation between these formats is not permitted in any scenario

N/A = Translation from format 2 to format 2 is technically permitted, but should not be implemented as it provides no value (format 2 may only be used with offline PIN verification to the payment card)

Unique secret / private keys
- per purpose & per device
- Same public key in many devices is OK

Use triple DES, AES, RSA, or ECC
- TDES requires unique key per transaction for many standards, and is being deprecated, consider AES for new implementations

A key must have equal or greater bit-strength than the key(s) it protects
(key bit-strength equivalency noted in the rows across the following table)

Algorithm	TDES	RSA	ECC	DSA	AES
Minimum key bit size	112	2048	160	1024/160	
Minimum key bit size	168	2048	224	2048/224	
Minimum key bit size		3072	256	3072/256	128
Minimum key bit size		7680	384	7680/384	192
Minimum key bit size		15360	512	15360/512	256

Do not variant across hierarchy levels
- Variants should be considered deprecated for new solutions
- Use One Way Functions such as CMAC for key derivation where possible

Use dual control & split knowledge for key loading and handling of secret and private keys
- Split knowledge is not required for public keys

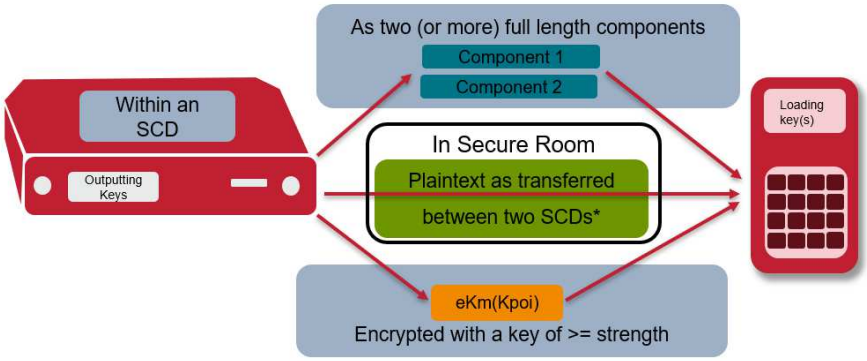
Dual Control: An action must be authorized by two or more individuals to prevent malicious action(s)

Split Knowledge: No one person can ever know any part (even one bit) of a secret or private key

Keys must be generated using a random or pseudo-random process
- Never generate working keys directly from passwords
- Shared secrets used to derive keys must be treated like keys and meet key entropy, length, and security requirements (such as approved forms)

Unattended and SCR(P) devices cannot be loaded with plaintext secret or private keys
- Avoid all plaintext secret/private key loading where possible – use key agreement or RKI

Approved key forms:



Module 1: Core Requirements

Section A - Physical Requirements

A1: Tamper Detection Mechanisms
A2: Robustness Under Changing Environmental and Operational Conditions
A3: Protection of Sensitive Functions or Information
A4: Monitoring During PIN Entry
A5: Determining Keys Analysis
A6: Physical Security of Display Prompts
A7: Visual Observation Deterrents
A8: Magnetic Stripe Reader
A9: Component Protections against Removal
A10: Audible Tones During PIN Entry

Section C – Online PIN Requirements

C1: Key Substitution

Section D – Offline PIN Requirements

D1: Penetration Protection
D2: ICC Reader Slot Visibility
D3: ICC Reader Construction (wires)
D4: PIN Protection During Transmission Between Device and ICC Reader

Section B – Logical Requirements

B1: Self Test
B2: Logical Anomalies

B3: Firmware Certification
B4: Firmware Updates
B4.1: Software Authenticity
B4.2: Signing
B5: Differentiation of Entered PIN
B6: Clearing of Internal Buffers
B7: Protection of Sensitive Services
B8: Sensitive Service Limits
B9: Random Numbers
B10: Exhaustive PIN Determination
B11: Key Management
B12: Encryption Algorithm Test
B13: Encryption or Decryption of Arbitrary Data
B14: Clear-Text Key Security
B15: Transaction Controls
B16: Logical Management of Display Prompts
B17: Application Separation

B18: Minimal Configuration
B19: Component Integration Documentation
B20: Security Policy

Module 2: POS Terminal Integration Requirements

Section E - Integration Requirements

E1: Target of Evaluation (TOE) Identification
E2.1: Integration of PIN Entry Functions
E2.2: Overlay Attack Protection
E3.1: Integration Vulnerabilities
E3.2: Protection Against Card Trapping
E3.3: PIN Entry Interface Segregation
E3.4: User Interface Consistency
E3.5: Control of any Numeric Interface
E4.1: Protection Against Removal
E4.2: Unauthorised Removal – Integration Documentation
E4.3: Unauthorised Removal – Embedded Devices

Module 3: Open Protocols Requirements

Section F&G - Discovery and Definition of Protocols and Interfaces

F1: Identification of Interfaces
G1: Vendor Vulnerability Assessment Procedures
G2: Vulnerability Assessment of all Interfaces
G3: Vulnerability Disclosure

Section H – Vendor Guidance

H1: Security Guidance for the Protocols and Interfaces
H2: Default Configuration of the Interfaces
H3: Key Management Security Guidance

Section I – Operational Testing

I1: Use of Secure Protocols
I2: Secure Protocols to Provide Data Confidentiality
I3: Secure Protocols to Provide Data Integrity
I5: Secure Protocols to Provide Exception Handling and Replay Detection
I6: Session Management

Section H – Vendor Guidance

J1: Security Guidance for Configuration Management
J2: Security Maintenance
J3: Security Guidance for Terminal Updates
J4: Securing Configuration and Software Updates

Module 4: Secure Reading and Exchange of Data

K1: Account Data Processing
K1.1: Account Data Protection
K2: Account Data Protection – Integration
K3: Determining Keys Analysis
K4: Encryption Mechanisms
K5: Remote Key Distribution
K6: Data Origin Authentication
K7: Unique Secret and Private Keys Per Device
K8: Encryption/Decryption of Arbitrary Data within a Device
K9: Remote Access
K10: Firmware Certification
K11.1: Software Authenticity
K11.2: Software Guidance
K12: Firmware Updates
K13: Logical Anomalies
K14: Open Protocols and Services
K15: Output of Clear-text Account Data
K15.1: Protection of Clear-text Data
K15.2: Clearing of Internal Buffers
K16: Surrogate PAN values
K16.1: Salt Generation
K16.2: Salt Storage
K17: Key Management
K18: Exhaustive PAN Determination
K19: Robustness under Changing ENvironmetal and Operational Conditions
K20: Application Separation
K21: Minimal Configuration
K22: Protection of Sensitive Services
K23: Sensitive Service Limits
K24: Secure Enablement Tokens

Module 5: Device Management Security Requirements

L1: Change Control
L2: Certified Firmware Control
L3: Device Component Control
L4: Production Firmware Control
L5: Post-Production Storage
L6: Secret Information
L7: Component Control during Design and Development
L8: Repair and Inspection Control
M1: Shipping Tamper-Protection Documentation
M2: Device-Accountability Transfer Procedures
M3: Shipping Security Procedures
M4: Development-Security Documentation
M5: Authenticity of POI Security-Related Components
M6: Authenticity of POI Security-Related Components for Key-Loading Facility
M7: Unique Visible Identifier
M8: Operational Management Manual

Example of a 0200 financial advice message from a terminal to an acquiring host. Each field is separately highlighted, with the message type shown in blue, initial bitmap shown in orange, and the subsequent fields noted in that bitmap shown with either a box or grey highlight.

0200323006012 0C00201003000000000002000818033854000050100818143818051010151134C101004
9110000012345 6F3354321012345678 90D1412201084060352F3131313131313131313233343536373839303132333435 01809F02060000000020009F0306000000000000820258009F360200029F26085A2D0D38BD
3FDDE69F2701809F34034103029F1E0832323232323232329F0D05FC50A000009F0E0500000000009F0F05
F870A498009F10120212A5000F040000567800000000000000FF9F090200029F3303E0F0C89F1A0200369F3
501229505000000800 05F2A0200369A031008189C01009F3704176BF0819F0607A00000000410105F28020
0569F0702FF004F07A0000000041010 89ABC87D00000000

This message is broken up into its constituent parts below.

Message Type: 0200
Bitmap: 3230060120C00201

Bit 3: 003000 [Processing code]
Bit 4: 000000002000 [Transaction Amount]
Bit 7: 0818033854 [Time of transmission]
Bit 11: 000050 [System Trace Audit Number (STAN – unique transaction ID)]
Bit 12: 100818143818 [Time of transaction]
Bit 22: 051010151134C101 [Entry mode]
Bit 23: 0049 [Card sequence number]
Bit 32: Length: 11
00000579944F [Acquiring institution identification code]
Bit 35: Length: 33
5432101234567890D1412201084060352F [Track 2 data]
Bit 41: 3131313131313131 [Card Acceptor Terminal Identification Code - CATID]
Bit 42: 313233343536373839303132333435 [Merchant ID - CAID]
Bit 55: Length: 0180
01809F02060000000020009F0306000000000000820258009F360200029F26085A2D0D38BD3FDD
E69F2701809F34034103029F1E0832323232323232329F0D05FC50A000009F0E0500000000009F0F05F870A
498009F10120212A5000F040000567800000000000000FF9F090200029F3303E0F0C89F1A0200369F350122
950500000080005F2A0200369A031008189C01009F3704176BF0819F0607A00000000410105F280200569F
0702FF004F07A0000000041010
Bit 64: 89ABC87D00000000 [Transaction Message Authentication Code (MAC)]

UL.com

05/2019