Chapter 17
Hardware Security Modules

Stathis Mavrovouniotis and Mick Ganley

Abstract Hardware Security Modules/(HSMs), also known as Tamper Resistant Security Modules (TRSMs), are devices dedicated to performing cryptographic functions such as data encryption/decryption, certificate management and calculation of specific values such as card verification values (CVVs) or Personal Identification Numbers (PINs). What these devices offer is tamper response, the capability to detect any attacks on their surface and securely delete the sensitive content stored in their memory. Such devices are manufactured to meet specific criteria [e.g. Federal Information Processing Standard (FIPS)] and must be appropriately managed throughout their whole lifecycle. Together with encryption algorithms, cryptographic functions and vendor provided functionalities, they host one or more cryptographic keys that respond to automated or manual commands. Physical security and key management are essential in order to protect the confidentiality and integrity of the keys and these requirements are properly described in various standards. Due to the specific functionality of HSMs, there have been many published attacks via the command interface, which reinforces the need for adequate controls, both physical and logical, around these devices.

17.1 Introduction

The first question that needs to be addressed is what is meant by a Hardware Security Module (HSM)? In order for a device to be classified as an HSM, it must belong to

S. Mavrovouniotis (✉)
20 Pandrosou Str, P. Faliro, 17564 Athens, Greece
e-mail: mavrovouniotis@gmail.com

M. Ganley
Information Security Group, Smart Card Centre, Royal Holloway, University of London, London, United Kingdom
e-mail: mick.ganley@rhul.ac.uk

the family of Tamper Resistant Security Modules (TRSM) or Secure Cryptographic Devices (SCD), which are physically secure devices and/or tamper responsive, meaning that any attempt at penetration of the device will cause immediate erasure of all secret information stored in the memory of that device [1]. An HSM is any hardware device, with some level of tamper-resistance,[1] which is used for cryptographic processing. Of course, this rather broad definition would include smart cards and other devices that are discussed elsewhere in this book. It would also include, for example, devices used at the network level to provide high-speed encryption, devices for issuing/signing certificates for a Certification Authority (CA), devices using for time stamping, etc. Another good example is that of retail Point of Sales terminals (POS terminals) used for processing "Chip and PIN" transactions, which have a security core that is frequently referred to as an HSM.

The HSM itself is either a peripheral device to the host computer or bus-connected. Nowadays, most peripherally connected HSMs communicate with the host machine via Ethernet or fibre cable, but in the past a variety of communication protocols were usually supported; customers would choose the protocol that best suited the required transaction throughput and available budget. As well as having a port to allow communication with the host computer, HSMs usually support a variety of other input/output methods, for example smart card reader, key pad, a dedicated management port, printer port, a CD/DVD drive to allow software loading, or a console cable in order to perform the HSM management or the key ceremonies. In simple terms, therefore, an HSM has many of the characteristics of a PC, the main differences being the limited functionality and the physical and logical security of the device, which will be described later in the chapter.

17.2 HSM Usage

Clearly an HSM can be used in any situation where high-grade cryptographic security is required. An HSM is most commonly a hardware device or a Payment card industry (PCI) card that responds to commands sent to it by an application, via a vendor-specified application programming interface (API). It is generally straightforward to modify the API to meet customer requirements; this may be done by the customer or, more usually, by the HSM vendor. HSM software/firmware is digitally signed, either directly or indirectly, using a vendor private key and verified using the corresponding public key installed in the HSM as part of the manufacturing process. Examples of the use of HSMs include the protection of personal data (e.g. health records, databases, etc), bulk encryption (e.g. satellite broadcasting) and trusted third-party services (certificate authorities, signature authorities, etc).

A typical 3-tier architecture containing an HSM is depicted in 17.1.

[1] The term "tamper resistant", in this context includes "tamper-evident" and "tamper-detective" that will often appear in this chapter and which can be used interchangeably, as well as "tamper responsive" which refers to the reaction of the device in a tamper attack.

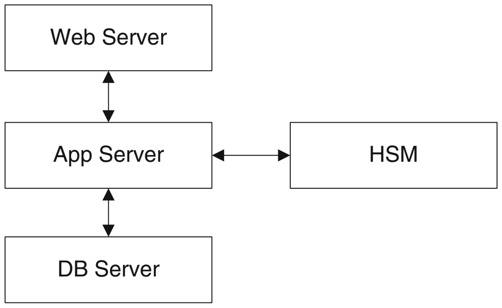

Fig. 17.1 A typical 3-tier architecture including an HSM

A message for the application server would be sent to HSM for cryptographic processing, prior to being sent to the next tier of the process. In a case of encryption of financial record, the data would be received by the web server and would then be passed to the application server. The application server would then, with the aid of the HSM, encrypt the data to pass on to the DataBase server to be stored.

In a case of a financial transaction, the application server would pass the financial data to be verified to the HSM, which would calculate, on the fly, a cryptographic value [(such as a Card Verification Value (CVV)] with the corresponding key Card Verification Key (CVK) and compare it with the value provided in the transaction, in order to approve or decline the transaction. As stated earlier, we will focus on HSMs used in the financial sector. Such an example is their use in "Chip and PIN" payment cards, such as a debit or credit cards. The two principal areas where HSMs are used with such cards are:

- data preparation, card personalisation and Personal Identification Number (PIN) mailer printing, as part of the issuing process;
- transaction processing.

In terms of card personalisation, a variety of secret or sensitive values need to be generated and loaded onto the card; these include a number of cryptographic keys (symmetric keys used during transaction processing and asymmetric keys, together with certificates, used for authentication purposes), as well as cryptographic values, such as a PIN and CVVs (CVV / CVV2 / iCVV). These values are typically generated with the use of specific cryptographic keys PIN Verification Key (PVK) for PIN, CVK for CVV values—we will call them functional keys in general) during the data preparation process (the data generation creates the demographic data) and then transmitted to the card personalisation system for loading onto the card. The PIN itself may need to be transmitted to another organisation for printing on some form of PIN mailer. The protection of such data, during generation, transmission and loading/printing, is provided by HSMs.

During transaction processing, at least in the online case, an HSM is used by the card issuer to ensure the integrity of transaction messages. In the particular case of the card being used at an Automatic Teller Machine (ATM) to withdraw cash, the PIN is encrypted by the ATM and sent to the issuer for verification, possibly via an acquiring organisation. The acquirer would use an HSM for PIN translation and message integrity purposes.

Obviously, the overall handling of Chip and PIN cards is far more complicated than the rather brief description provided above. Each part of the issuing process and each part of the acquiring process requires specific keys for specific cryptographic functions, and an HSM is present in each step of this process. We will discuss key usage later in the chapter, but hopefully the above serves to illustrate HSM usage within this single financial application. Continuing with the above example, it is clear that HSMs must support a variety of cryptographic mechanisms, such as:

- cryptographic algorithms, such as DES (although no longer accepted), 3-DES, AES, RSA, SHA-1, SHA-256;
- cryptographic key management, including key generation, key derivation, key distribution, key storage, etc;
- data encryption, in particular PIN encryption techniques;
- data integrity, including Message Authentication Code (MAC) generation and verification, digital signature generation and verification;
- CVV generation and verification;
- PIN generation and verification, PIN translation;
- chip cryptographic keys generation and transmission, cryptographic values generation and verification.

A typical HSM used solely in financial applications will support all of the above functionality including many PIN generation/verification, PIN block formats, key management techniques, MAC algorithms and encryption modes. The sheer number of commands that are supported by an HSM, including many that are not actually used by the application(s), raises a number of security concerns that we will investigate later in this chapter.

Whilst it may be tempting simply to provide a range of "primitive" functions for an HSM and allow application developers to build more complex functions from these primitives, this would have an adverse effect on performance and could have serious implications for the security of command processing. Security evaluation of such a solution would also be difficult, as discussed later in this chapter. Instead, an HSM API will typically include many rather complicated functions, often with a range of options. For example, in the Chip and PIN application discussed earlier, a single HSM command used by an acquirer could involve PIN translation, including a format change, data decryption using one key and then re-encryption using a different key (even possibly a different mode of encryption) and MAC verification followed by the generation of a new MAC.

The typical format of a command for an HSM used in a payment transaction would be:

- Command header
  - Command code
  - Command data

Table 17.1 Sample HSM command and responses messages

Command message (with comments)	
Header	Typically for use by the calling application
Command code	Unique command identifier
Source PIN encryption key	Encrypted using a local key
Destination PIN encryption key	Encrypted using a local key
Source MAC key	Encrypted using a local key
Destination MAC key	Encrypted using a local key
Source PIN block format	e.g. ISO 9564 format 0 [2]
Destination PIN block format	May be the same as the source format
Source MAC mode	e.g. ISO 9797 algorithm 3 [3]
Destination MAC mode	May be the same as the source format
PIN block data	(Optional) depends on PIN block formats
PIN block	Encrypted with the source PIN key
Message data	Transaction data
Source MAC	MAC on message data, using source MAC key
Command trailer	(Optional)
Response message (with comments)	
Header	Typically for use by the calling application
Response code	Unique response identifier
Error code	e.g. 00 = no errors
PIN block	Encrypted with the destination PIN key
Destination MAC	MAC on message data, using destination MAC key
Response trailer	(Optional)

- Command trailer

The corresponding response message is:

- Response header
  - Response code
  - Error code
  - Response data
- Response trailer

So, for example, an acquirer command that involves a PIN translation and a MAC verification and new MAC generation is described in Table 17.1:

In the above command, errors may occur in a variety of ways, for example:

- key parity errors, if using DES or 3-DES (stored keys may have become corrupted);
- an invalid PIN block format or MAC mode;
- a PIN block error (e.g. format of the plaintext PIN block does not match the specified format);
- MAC verification failure;
- command syntax error.

In the event of an error being detected the HSM should return an appropriate error code and continue to the next transaction. As already mentioned, a crucial function of an HSM is the protection of secret data, in particular cryptographic keys and PINs. In a following section, we consider HSM key management in more detail.

17.3 HSM Physical Security

The purpose of an HSM is to provide high-grade cryptographic security and a crucial aspect of this security is the physical security of the device. It must be emphasised, however, that this is only one aspect of HSM security—attacks via the HSM's API and procedural aspects of HSM security are equally important. Indeed, it could be argued that an HSM's physical security is the easy part; a physical attack is likely to be detected quickly, whereas a logical or procedural attack might never be detected!

An HSM's primary defence against physical attack is based around the concept of a tamper-resistant core, which is an HSM sub-system that contains all the sensitive components. As the most common approach, the security core will provide battery-backed volatile memory for the storage of plaintext cryptographic keys (such as the Host Master Key (HMK), discussed later) and all cryptographic processing will be performed within the core system. The tamper-resistant features of the core sub-system mean that should attack on the core be detected then the contents of the secure memory will be immediately deleted ("zeroised"). Typically, HSM software/firmware is stored in a combination of ROM and E2PROM and so will not be deleted if the device is tampered. The ANSI X9.24-1 standard [4] mandates the following:

> An HSM must have features that resist successful tampering, which includes penetration without zeroisation of security parameters, unauthorised modification of the HSM's internal operation or insertion of tapping mechanisms or non-intrusive eavesdropping methods to determine, record or modify secret data; such features must include one or more of the following:

- tamper-detection mechanisms, which must be active regardless of the HSM's power state;
- physical barriers to make successful tampering infeasible;
- sufficient resistance to tampering, so that successful tampering requires an extended period of time (absence of an HSM from its authorised location should be noticed before the tampered device is returned to resume cryptographic operations);
- the HSM's construction is such that successful tampering will cause visible damage to the device that is likely to be noticed after the device has been returned to its authorised location but before cryptographic operations are resumed—i.e. a tamper-evident feature.

Immaterial of the use of an HSM, because of its nature, the physical controls around it are very strict. This would most commonly mean the HSM is located within a high security area, probably locked inside a secure cabinet, under dual physical controls (each cabinet door would require two controls—for example, two keys, a key and a combination or a key and a biometric) or a similar and equally effective approach. Consequently, an attacker would find it very difficult to remove an HSM from its normal location without detection—unless he is an employee with physical access and rights to do so anyway—this is why the dual control should be enforced and actually the second person should not by default be entitled to be around the specific cabinet. However, the same may not be true for other types of HSM (e.g. the security core of a retail PIN pad). In general, therefore, the primary defence of an HSM against physical attack is the tamper-detection circuitry, which must zeroise secure memory as soon as an attack is detected.

Attacks that must be defended against include:

- drilling or otherwise penetrating the security core;
- low temperature attacks;
- attacks involving variations in voltage or current;
- power analysis or timing attacks (members of a class of attacks known as "side channel" attacks).

Typical defences against such attacks include wrapping the entire security core in some form of fine-grained electronic mesh and then encasing the core in epoxy resin. An attacker attempting to penetrate the resin is likely to break the mesh. If the mesh is broken or damaged in some way then the zeroisation circuitry is immediately triggered.

Other HSM defences include the use of physical locks, micro-switches, light-sensitive diodes, mercury tilt-switches, temperature sensors and sensors to detect variations in voltage and current. Side channel attacks are unlikely to be successful unless the attacker is able to penetrate the core (in which case a side channel attack is probably unnecessary!) and in any case HSM vendors usually build in defences against such attacks. Important note: Some of these controls can be enabled or disabled, so it must be stressed that they can protect the HSM only when activated. An HSM is considered as an HSM only if it has these controls activated!

Many early HSMs used by the financial industry had only rudimentary tamper-detection mechanisms, often no more than a couple of micro-switches to detect when an HSM's casing was opened. This could be easily by-passed by an attacker and so in such circumstances the physical and access security of the computer centre environment became the principal defence against an attacker. Nowadays, HSM security is usually evaluated against standard requirements and we now move on to consider such evaluations.

17.4 HSM Security Evaluation and Approvals

Although there are a number of standards detailing security requirements for cryptographic modules, for example ISO 13491 [5, 6], most HSMs used in the financial sector are evaluated against the requirements of the Federal Information Processing Standard (FIPS) 140-2 [7]. Devices used in some government applications may also need to be evaluated against the Common Criteria requirements (e.g. [8]).

More recently, a PCI standard for HSM security has been published (PCI-HSM, see [9]) and we will briefly consider this standard at the end of this section. For the time being, however, we will concentrate on FIPS 140-2. The FIPS 140-2 standard ("Security Requirements for Cryptographic Modules") specifies security requirements in 11 different areas and covers 4 different security levels, with level 1 being the lowest and level 4 being the highest. Each level builds on the previous level. The following table, copied from the FIPS 140-2 standard summarises the requirements for the different levels:

The term Operational environment refers to the management of the software, firmware and/or hardware components required for the module to operate. The abbreviations PP and EALx refer to the Common Criteria Protection Profile and Evaluation Assurance Level x, respectively (see [8]).

There is little purpose to be served in a detailed discussion of FIPS 140-2 in this chapter, but the following points are noted:

- devices approved to FIPS 140-2 level 1 or level 2 provide limited protection for cryptographic keys and other sensitive data; such devices are not appropriate for many financial applications; indeed, the example given in the standard of a device that could achieve level 1 approval is a PC encryption board;
- HSMs used in financial applications are typically approved to level 3 or 4; note however that for particularly sensitive applications the physical security requirements of level 3 may not be acceptable except in secure environments;
- there is a large "gap" between the level 3 and level 4 requirements, in particular the requirement for a formal model for design assurance; some HSMs meet the level 4 requirements in many areas and yet only receive approval to level 3;
- FIPS 140-2 evaluation does not consider side-channel attacks, such as power analysis, nor does it include command manipulation attacks, based on the device's API; this latter topic has already been discussed briefly and will be considered further later in this chapter. Level 3 is the approved level by all payment schemes security requirements.

Level 4 approval is hard to achieve and currently (early 2012) very few products have been approved to this level:

Those products with certificate #235 and lower were evaluated against the earlier FIPS 140-1 standard. A complete list of all FIPS 140 approved products can be found at [10]. The list gives an overall security level for each approved product, but also includes those areas where the overall level has been exceeded.

As previously mentioned, the PCI-HSM standard [9] has recently appeared and lists its security requirements in 4 categories (Tables 17.2 and 17.3):

- A: Physical security
- B: Logical Security
- C: Device Security during Manufacture

Table 17.2 Summary of FIPS 140-2 security requirements

	Level 1	Level 2	Level 3	Level 4
Cryptographic module specification	Specification of cryptographic module, cryptographic boundary, approved algorithms and approved modes of operation; description of cryptographic module, including all hardware, software and firmware components; statement of module security policy			
Cryptographic module ports and interfaces	Required and optional interfaces; specification of all interfaces and of all input and output paths		Data ports for unprotected critical security parameters logically or physically separated from other data ports	
Roles, services and authentication	Logical separation of required and optional roles and services	Role-based or identity-based operator authentication	Identity-based operator authentication	
Finite state model	Specification of finite state model; required states and optional states; state transition diagram and specification of state transitions			
Physical security	Production grade equipment	Locks or tamper-evidence	Tamper detection and response for covers and doors	Tamper detection and response envelope; EFP or EFT[a]
Operational environment	Single operator; executable code; approved integrity technique	Referenced PPs evaluated at EAL2 with specified discretionary access control mechanisms and auditing	Referenced PPs plus trusted path evaluated at EAL3 plus security policy modelling	Referenced PPs plus trusted path evaluated at EAL4
Cryptographic key management	Key management mechanisms: random number and key generation, key establishment, key distribution, key entry/output, key storage and key zeroisation			
	Secret and private keys established using manual methods may be entered or output in plaintext form		Secret and private keys established using manual methods may be entered or output encrypted or with split knowledge procedures	
EMI/EMC[b]	47 CFR FCC[c] part 15, subpart B, class A (business use) applicable FCC requirements (for radio)		47 CFR FCC part 15, subpart B, class B (home use)	

Table 17.2 Summary of FIPS 140-2 security requirements

	Level 1	Level 2	Level 3	Level 4
Self-tests	Power-up tests: cryptographic algorithm tests, software/firmware integrity tests, critical function tests; conditional tests			
Design assurance	Configuration management (CM); secure installation and generation; design and policy correspondence; guidance documents	CM system; secure distribution; functional specification	High-level language implementation	Formal model; detailed explanations (informal proofs); preconditions and postconditions
Mitigation of other attacks	Specification of mitigation of attacks for which no testable requirements are currently available			

[a]Environmental failure protection and environmental failure testing
[b]Code of Federal Regulations (CFR) and Federal Communication Commission (FCC)
[c]Electromagnetic Interference/ Electromagnetic Compatibility

Table 17.3 Products approved to FIPS 140-2 overall level 4

Certificate #	Vendor	Product
1505	IBM	IBM 4765 cryptographic coprocessor security module
1340, 956, 235, 146, 123, 112	AEP	Networks advanced configurable cryptographic environment (ACCE) various versions
1174, 930	Hewlett Packard	Atalla cryptographic sub-system (ACS)
661, 524	IBM	IBM eServer cryptographic coprocessor security module
541	AEP	Networks AEP enterprise CM
118	IBM	IBM eServer zSeries 900 CMOS cryptographic coprocessor
116	IBM	IBM 4758-002 PCI cryptographic coprocessor (miniboot layers 0 and 1)
115	Thales	Secure generic sub-system (SGSS)
40	IBM	IBM S/390 CMOS cryptographic coprocessor
5	IBM	IBM 4758 PCI Cryptographic coprocessor (miniboot layers 0 and 1)

- D: Device Security between Manufacture and Initial Key Loading

Many of the requirements for physical security are derived from the level 3 FIPS 140-2 requirements, although requirement A2 includes some side-channel attacks, such as power analysis. The logical requirements are generally more strict than the corresponding FIPS 140-2 requirements, in particular the key management requirements. Of particular interest, however, is requirement B9, which states:

"The HSM's functionality shall not be influenced by logical anomalies such as (but not limited to) unexpected command sequences, unknown commands, commands in a wrong device mode and supplying wrong parameters or data which could result in the HSM outputting the clear-text PIN or other sensitive information."

We will return to this topic later in this chapter.

Currently, only three products are listed on the PCI web site [11] as having been approved against the PCI-HSM requirements, namely the, HP Atalla Ax160, Thales payShield 9000 and Tokheim Crypto HSM+ devices.

17.5 HSM Management

Under normal operating conditions, HSMs are intended to work without any manual intervention. However, there are many HSM activities that require some form of human input, for example:

- HSM installation and initialisation, including generation of the highest level local key (the HMK, if used);
- define users and corresponding authorisations;

- generation, import/export and installation of other keys;
- configuration; for example, communication parameters and security policy;
- state changes, such as putting the HSM off-line to the host or requiring special authorisation for sensitive functions (e.g. key loading);
- enabling and disabling of commands;
- enabling and disabling of PIN block formats;
- audit functions;
- diagnostics and problem solving;
- other tasks, such as those relating to real-time clock management (e.g. "set time");
- firmware and other software (e.g. licence files) updates.

Such activities have in the past required direct access to the HSM, via a dedicated management port, while some HSM vendors now offer a "remote" solution for managing HSMs. This has many advantages in terms of personnel and time, especially when trying to manage a large number of geographically dispersed HSMs. Remote access requires additional security mechanisms to be in place, in particular mutual authentication between operators and HSMs and confidentiality of transmitted data.

Regardless of the actual mechanism employed, all management activities must be governed by detailed and rigorously enforced procedures. Security incidents are far more likely to occur because of poor management than an attacker somehow compromising a physically secure HSM located in a data centre.

An HSM's security policy can be configured to cover items such as:

- number of HMK components—as described later in the chapter;
- minimum number of components for plaintext key entry;
- enabled commands and PIN block formats;
- denying the use of single-length DES;
- minimum PIN length—for the incoming PIN from a transaction;
- various key export/import options (e.g. ANSI X9.17 not permitted);
- types of keys that can be exported or imported;
- permitted number of key check value characters;
- permitting clear PINs to be input or output, for example when PIN translation is performed;
- data encryption and decryption options;
- audit options;
- authorisation options.

The above configuration options are reasonably standard across most HSMs used for the processing of financial transactions, but vendors will typically offer a range of other configuration possibilities, for example:

- preventing a single-length DES key masquerading as a double- or triple-length key;
- encrypted decimalisation tables, i.e. tables used to map hexadecimal characters to decimal digits;
- weak PIN checking;

- minimum HMAC length for verification;
- specific restrictions on individual commands.

One area of concern for HSM management relates to the entry of plaintext key components which are combined to form a secret value, such as a cryptographic key. Such components are typically received from a partner organisation, are in paper form and must be entered by (trusted) security officers into the HSM, inside which they are combined to form the clear key, which is then output encrypted under the HMK. The issue is the entry of the components, which is usually done via some form of terminal, such as a PC, which leads to the possibility of the components being captured during entry (e.g. via a key logger or some form of device connected to the communication line to the HSM). In the past, such concerns have been mitigated by strict procedural controls but nowadays the payment industry is demanding that key components be entered via a more secure mechanism. For instance, requirement 13 of the PCI-PIN security requirements [1] states:

"The mechanisms used to load keys, such as terminals, external PIN pads, key guns, or similar devices and methods are protected to prevent any type of monitoring that could result in the unauthorized disclosure of any component."

One possibility would be the use of a secure retail PIN Entry Device (PED), approved against the PCI-PED security requirements [12]. HSM vendors are now actively seeking ways to meet this particular requirement. We will discuss the key management procedures in more detail in the next chapter.

17.6 Key Management

As mentioned before, different keys are used for different cryptographic processes and the whole of the proper functioning of the cryptographic model relies on the protection and proper use of the keys, which is the main principle of cryptography, as mentioned in all cryptography related publications. An HSM is essentially a cryptographic engine and it serves no useful purpose if secret or private cryptographic keys are exposed to an attacker during command processing. Hence, such keys must never appear in plain form outside the secure confines of the HSM. There is one exception to this rule, namely that if a key is required to appear in plain form outside the HSM then it must be in the form of two or more components and strict procedures must be followed to enforce the principles of dual control and split knowledge—ensure that the components cannot exist in the hands of one individual at any point in time. PCI-PIN Security Requirements [1] together with payment schemes standards, provide specific requirements about how the safety of the participating keys is preserved, during all phases of a key management lifecycle. In order to address this part of the chapter, we split the keys into three main categories as below:

1. Storage keys: keys such as the HMK, which is used to encrypt other keys when stored.

2. Transport keys: keys that are used to encrypt keys during key exchange, e.g. a Key Encrypting Key (KEK).
3. Functional keys: keys used to perform specific cryptographic functions and generate respective cryptographic values, such as PVKs, CVKs, chip authentication keys, PIN block encryption keys, etc.

There are two principal methods for protecting keys used by an HSM:

Method 1. Store all keys inside the secure memory of the HSM; in this case, when sending a command to the HSM a pointer to the key to be used must be included in the command message. This technique has one significant drawback, namely that if the HSM is tampered and loses its keys then all the keys must be reloaded into the HSM. In addition, if multiple HSMs are used (for reasons of throughput and/or redundancy) then all the keys must be loaded into each HSM.

Method 2. A single key, which we have already termed a "HMK" is loaded into the HSM and all other keys are encrypted with the HMK and stored in some form of key database accessible to host applications; this database can exist either on the database server of a 3-tier model, or as a file within a mainframe system.

Of course, if the HSM loses its HMK then the key must be reloaded into the HSM, but unlike method 1 this is the only key that needs to be reloaded. The drawback in method 2 is that should the HMK be somehow compromised then all the other keys in the system are potentially compromised as well. For this reason, strict procedures must be in place to protect the HMK, including plaintext storage of the key in component form under dual control and split knowledge. The HMK must be a "strong" key, for example a triple-length 3-DES key or an AES-256 key. It goes without saying that, regardless of the HSM key management technique, all keys that are stored inside the HSM should be backed-up. Interestingly enough, there is no mandate for backing up keys, only that if the keys are to be backed-up the same controls used for the production keys should be also enforced on the backup keys (e.g. requirement 27 of the PCI-PIN standard [1]).

Remark on terminology: Method 2 is the most commonly used HSM key management technique. This means that in general:

- function specific keys must be transferred encrypted under a transport key, such as a KEK;
- function specific keys and transport keys must be stored encrypted under a HMK, summarised in the Table 17.4:

Protecting the confidentiality of keys is one issue that is addressed by the methods described above, but it is equally important to protect the integrity of such keys. In particular, it must not be possible for an attacker to modify a key or to use a key for a purpose for which it is not intended. The requirements that "keys must be used only for their sole intended purpose" and that "cryptographic keys ever present and used for any function must be unique (except by chance) to that device" are basic

Table 17.4 Matrix of different types of keys and their storage/exchange

Function/key	Storage keys	Transport keys	Functional keys
Storage	In components	Under HMK	Under HMK
Exchange	Not applicable	In components	Under KEK

principles for protecting the keys and are two very important requirements of PCI standard for PIN Security requirements [1, requirements 19 and 20].

The key management is performed by a team of custodians, chosen and managed in a way that the principles of dual control and split knowledge are met. The role of custodian is crucial—they have access, although controlled, to all cryptographic material, together with physical access to the HSMs. Thus, the custodians must be appropriately trained, the key management procedures must be very well documented, and audit trails must exist and be maintained for every activity relating to key management: key generation, import/export, key storage/retrieval, key back up, key replacement or destruction and arguably most importantly key compromise. Once these basic principles are met and the HSMs as well as the keys are appropriately protected, the chances of key compromise are minimal.

If an attacker were to try and attack the keys themselves he would be looking for the following [13]:

- production keys used in the test environment, allowing the technical support staff to attack the key structure;
- PINs not protected by a secure PIN block, allowing "dictionary" attacks;
- failure to use approved cryptographic devices for PIN processing;
- cryptographic keys non-random, non-unique and never change;
- hard copy keys in the clear or in clear-text halves;
- few, if any, procedures documented; and,
- no audit trails or logs maintained.

We give two simple examples to illustrate the importance of these requirements:

Example 1 Suppose a double-length 3-DES key is encrypted using (some variant of) the HMK in Electronic Codebook (ECB) mode—this is a very common encryption mode and it is analysed in relevant bibliography. The attacker could replace the second half of the encrypted key with the left half of the key, so that the modified key is really a single-length DES key masquerading as a double-length key. The HSM could be used with the modified key to generate sufficient data to allow a brute-force attack on the left half of the key. This could then be used to expose the right half of the original key. This attack can be prevented by a variety of techniques. For example, the HSM could be configured to prevent such a key masquerade, by checking that all parts of a double or triple-length key are different. We will discuss a more generic technique shortly. □

Example 2 Suppose a key is designated as a PIN encryption key (so, in particular, there is no HSM function that allows the key to be used to decrypt a PIN block).

Header (16 ASCII characters)	Optional Header Blocks (ASCII characters, variable length	Encrypted Key Data (variable length, ASCII encoded)	Key Block Authenticator (8 ASCII characters)

Fig. 17.2 TR-31 key block

If the attacker can change the key usage so that it appears to the HSM as a (generic) data encryption/decryption key then the key could be used to decrypt PIN blocks. One method, whereby, it may be possible to change key usage is via a combination of key export and key import. Until recently, HSM vendors used proprietary methods for local key management but were generally forced to use a "lowest common denominator" approach for exchanging keys among HSMs of different vendors. This approach usually involved exporting a key, encrypted under some higher-level KEK, using the ANSI X9.17 standard [14, now withdrawn), and in most cases when the key was encrypted under the KEK the original key usage could no longer be determined. Consequently, the attacker could easily import the key back into the HSM system as a different key type. HSM vendors have long recognised the importance of key usage and have employed a variety of techniques to ensure that a key is only used for its intended purpose. For example, different key types can be encrypted under different variants of the HMK, in some cases different variants are also used for different key parts. IBM HSMs use Control Vectors to define exactly how keys can be used by the HSM. Whilst such techniques can provide some level of protection for keys, the two examples above illustrate their limitations. □

The ANSI X9.24-1 standard [4] for retail financial key management mandates that keys must, amongst other things:

- be protected against disclosure and misuse;

and that 3-DES keys must:

- exist in a "key bundle";
- be secret and randomly or pseudo-randomly generated;
- have integrity, so that each key in the bundle cannot be altered in an unauthorised manner;
- be used as specified by the particular mode;
- be considered as a fixed quantity, in that it is not possible to manipulate part of the key, and that the key cannot be "unbundled".

Although the standard relates primarily to 3-DES keys, clearly the same requirements make sense for any secret or private key. The ANSI TR-31 standard [15] specifies a technique for meeting the requirements of X9.24-1 via the use of "key blocks". Although TR-31 is specifically for key distribution, it has been adopted and refined by some HSM vendors as a method of protecting local keys when encrypted under the HMK (Fig. 17.2).

The Key Data is encrypted using a variant of the KEK used to protect the key block, in Cipher Block Chaining (CBC) mode, with bytes 0–7 of the Header as the Initial Vector (IV). The Key Block Authenticator is a MAC over the rest of the key

Table 17.5 TR-31 key block header

Byte(s)	Field	Comments
0	Version ID	Value = A; current version
1–4	Key block length	Total length of key block
5–6	Key usage	e.g. key encryption, data encryption
7	Algorithm	e.g. DES, 3-DES, AES
8	Mode of use	e.g. encrypt only
9–10	Key version number	e.g. version of key in the key block or used to indicate that the key is a key component
11	Exportability	e.g. no export permitted
12–13	Number of optional blocks	Number of optional header blocks
14–15	Reserved for future use	Value 00

block data, calculated using a different variant of the KEK. The key block Header governs the use of the key contained within the key block and has the following structure (Table 17.5):

As mentioned, the TR-31 key block mechanism has been adopted and refined by some HSM vendors for local protection of keys using the HMK. Whilst the local use of key blocks will greatly improve the security of HSMs against the type of "key manipulation" attack described earlier, vendors are constantly battling against the need to maintain backwards compatibility for legacy systems and so the benefits of key blocks are nullified to a certain extent. There will still be potential problems involving key manipulation until all HSM vendors have introduced key blocks and legacy systems have been upgraded. This will be hopefully enforced in the next versions of PCI standards and payment schemes security requirements.

In conclusion, if the keys are administered in a proper way, and the HSMs are physically protected, an attacker, as the last resort, will focus on the attacks for command manipulation, which are addressed in the next section.

17.7 Command Manipulation Attacks

The final topic discussed in this chapter is that of HSM attacks based on the HSM's API, which we will designate "command manipulation attacks". This subject has already been mentioned a number of times and it is worth recalling requirement B9 of the PCI-HSM standard: "The HSM's functionality shall not be influenced by logical anomalies such as (but not limited to) unexpected command sequences, unknown commands, commands in a wrong device mode and supplying wrong parameters or data which could result in the HSM outputting the clear-text PIN or other sensitive information."

Two rather simple attacks have already been outlined, namely the use of a single-length DES key masquerading as a double-length key and changing key usage via a combination of key export and key import.

In the latter example, a PIN encryption key had its use changed to that of a data encryption/decryption key to allow PIN blocks to be decrypted. This same attack could also be used to decrypt keys in some earlier models of HSMs. A recent paper by Bortolozzo et al. [16] details a variant of this attack on a number of commercially available devices that support the PKCS#11 API [17].

Before describing some more sophisticated command manipulation attacks, it is worth asking the question as to whether such attacks are feasible in real-life. For example, some assumptions about the attacker need to be made:

- the attacker has detailed knowledge of the HSM and its command structure;
- the attacker is in a position to send commands to a "live" HSM;
- the attacker has access to live data, including keys (encrypted under the HMK) and transaction data;
- the attacker has knowledge of "standard" algorithms but does not generally have knowledge of proprietary techniques used by the HSM;
- the attacker may have physical access to the HSM but is not in a position to carry out sensitive management functions.

Financial institutions argue that these are not realistic assumptions and that physical restrictions, procedural arrangements, host configuration settings and comprehensive audit trails make impossible such types of attack. Whilst this may be true for some organisations, there is absolutely no guarantee that all organisations using an HSM have such stringent security regimes. It is probably the case that an "outsider" would find it extremely difficult to attack the system via command manipulation, hence the likely attacker would almost certainly be an "insider", probably with a number of system privileges.

The logging of HSM transactions in audit trails is potentially a major deterrent to an attacker, but somebody with detailed system knowledge may be able to get round that. If the attacker can directly access the HSM's host port then there may be no audit trail anyway. Finally, one problem with most audit trails is that they contain so much information that nobody bothers to look at them, at least not until it is too late!

So, the above assumptions are probably "not unreasonable" and a number of command manipulation attacks could perhaps be carried out by a privileged "insider". If the reader thinks this is an unduly pessimistic view of the security of HSM systems in financial institutions then he or she should read the article "Why Cryptosystems Fail" by Ross Anderson [18] or Anderson's book "Security Engineering" [19]. Many papers on command manipulation attacks have been written in recent years, with the most comprehensive treatment being given in Jolyon Clulow's MSc thesis [20]. One of the techniques described in this thesis is that of finding a single-length DES key via a "parallel search" technique, initially proposed in [21]. Here, by obtaining the same plaintext encrypted under many different keys, an exhaustive search to compromise one of the keys (but you cannot specify which one) can be speeded up significantly. For example, if 2k single-length DES keys all encrypt the same block of data then an exhaustive key search would expect to find one of the keys after an average of 255-k attempts. This technique forms the basis for a well-publicised attack [22] on the IBM 4758 HSM, summarised below.

Remark 1 The IBM 4758 HSM has been approved to FIPS 140-1 level 4 (see Table 17.3). The attack does not invalidate this approval, as the FIPS 140 security requirements do not cover API attacks. Although the published attack was specifically against the IBM 4758 HSM, which has long since been withdrawn, the basic idea is still applicable to possible attacks against modern devices. □

Step 1. Use the parallel search technique to obtain the value of a single-length key, KDATA. This requires the use of the Encrypt function to generate the known plaintext/ciphertext pairs.

Step 2. Use the parallel search technique again, to obtain the value of a double-length KEK, which allows key export (KEKEXPORTER). The trick that allows this step is to make both halves of the KEK the same. This time, the corresponding plaintext/ciphertext pairs are obtained by exporting the known key, KDATA.

Step 3. Export all keys using the known key KEKEXPORTER and decrypt them at leisure.

One interesting aspect of this attack was the development of a DES search engine, based on an FPGA at a cost of less than $1,000, to carry out the searches in steps 1 and 2 in, approximately, 24 h for each stage.

A number of papers have been written on attacks on the IBM 3624 PIN verification technique (for example, [20, 23, 24]). This is a standard PIN generation algorithm and is implemented in most HSMs used in card payment systems.

One major benefit of this technique is that there is no need for the card issuer to maintain a database of (encrypted) PINs. Instead, a Customer PIN can always be regenerated or verified from input values, namely Account Related Data (ARD, for example the card's Primary Account Number), a PVK, a Decimalisation Table and an Offset. Specifically, the ARD, PVK and Decimalisation Tables are used to generate a Derived PIN, which is then combined with the Offset, to form the Customer PIN. This algorithm is described in Fig. 17.3. Note that the Derived PIN is a transitory value only—it never appears outside the HSM. In general, HSMs that support this PIN generation method have two specific commands:

- verify a Customer PIN;
- generate an Offset for a given Customer PIN.

The second command is to allow a customer to change his or her PIN. Note that the Derived PIN does not change, so that if a customer changes a PIN then the Offset must change to compensate.

One possible attack on this method (using the second command) is to run the command whilst making successive changes to the Decimalisation Table. Whenever the generated Offset is different from the correct value, the attacker can deduce one or more digits of the Derived PIN, from which the corresponding digits of the Customer

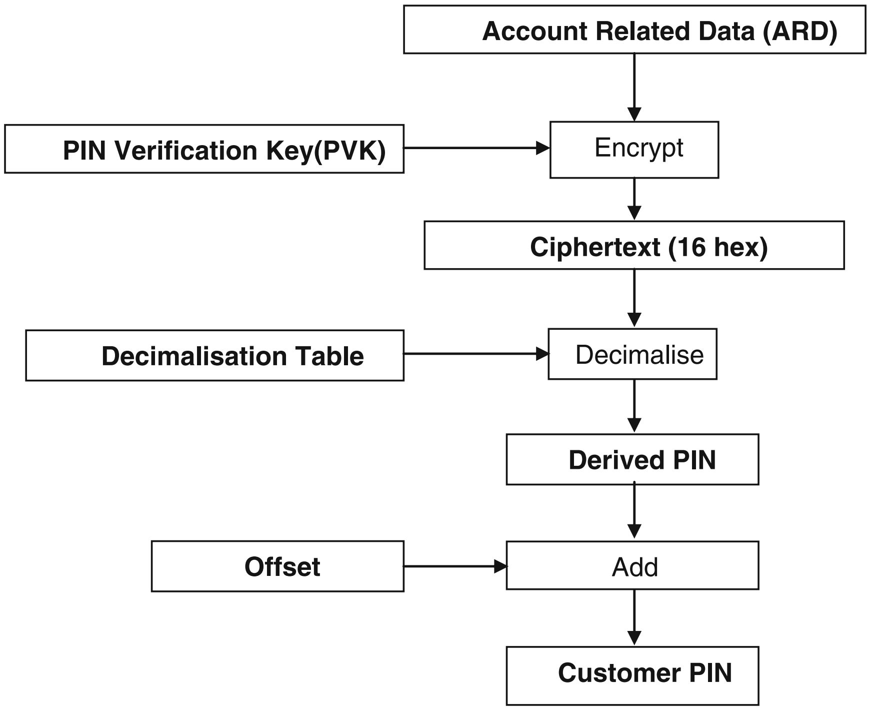

Fig. 17.3 IBM 3624 PIN generation algorithm

PIN can be calculated. This attack requires a maximum of 15 calls to the HSM. A slightly less efficient attack, which uses only the PIN verify command, involves modifying the Decimalisation Table, as above, to ascertain the PIN digits (but not their positions), via returned error codes, and then repeat the process this time also modifying the Offset. On average, this attack will reveal a Customer PIN in about 20 calls to the HSM.

In a third attack, it may be possible to compromise PINs for other customers via the use of insecure PIN blocks, when just one Customer PIN is known. For example, if the PIN for Customer X is known then the PIN for Customer Y could be translated to a PIN block format that does not involve an account number (e.g. ISO 9564, format 1) and then translated back to a format that involves the ARD of Customer X. By using the command to generate an Offset the PIN for Customer Y can be calculated. This attack requires only 3 calls to the HSM. The first two attacks described above can be easily defeated by using an encrypted Decimalisation Table, but this will not defend against the third attack.

One especially clever attack first described in Clulow's thesis (not involving the IBM 3624 algorithm) uses a combination of PIN verify and PIN translate commands to compromise PINs; essentially PINs are "flip-flopped" between different PIN block formats and error codes returned by the PIN verify command can be used to determine the PIN digits. The details of the attack are rather complicated and are not given here, but the interested reader should consult [20].

The above gives only a flavour of the types of command manipulation attacks that are possible. The crucial point is that many of these attacks are quite ingenious and use only “standard” HSM commands, so HSM vendors must be very careful when trying to satisfy requirement B9 of the PCI-HSM standard [9].

What, then, can be done to mitigate such attacks? The following suggestions would at least be a good starting point in addressing the problem:

Enabling and disabling functions: The golden rule should be that only those HSM features that are actually required should be enabled; this includes HSM commands, PIN block formats, PIN algorithms, etc; all other features should be disabled. In particular, enabling the generation of plain text PINs is a major risk, and is the only HSM related risk that is mentioned in a preventive measures paper, published by USSS and FBI [25].

Security policy: The HSM’s security policy should be configured as “tightly” as possible, subject to the requirements of applications calling the HSM.

Key blocks: HSM vendors should introduce key blocks as soon as possible and customers should ensure that they are using this feature.

Formal methods of analysis: Some formal approaches to the analysis of HSM command sets have been defined (for example, the previously mentioned paper by Bortolozzo et al. [16]), although the results have been rather “patchy”. HSM vendors should think about collaborating in the development of some sort of tool to enable formal analysis.

Vigilance: HSM vendors and HSM users should monitor academic papers describing new command manipulation attacks and (if necessary) modify HSMs as soon as possible to defend against such attacks. In addition, regular analysis of HSM command sets should be conducted by vendors, especially following major new releases or significant customisations.

Procedures: HSM users must ensure that all HSM-relevant procedures are strictly enforced, in particular that no unauthorised access to an HSM’s host port is possible and that audit logs of HSM transactions are regularly monitored.

Whilst the above defences cannot guarantee immunity to command manipulation attacks, they would certainly make the attacker’s life a lot more difficult.

17.8 Conclusions

In this chapter we have explained what we mean by an HSM, given some usages for HSMs (focusing primarily on the financial sector) and described the key management regime supported by many HSMs. Here we have also described two very simple attacks on the HSM’s API that exploit weaknesses in the key management structure.

We then moved on to discuss the physical security of HSMs and HSM security evaluation, in particular against the requirements of FIPS 140-2. One specific requirement of the PCI-HSM standard was also highlighted, essentially that an HSM API should be immune to command manipulation attacks. Following a short discussion on HSM management, we moved back to the topic of API attacks and outlined a variety of (known) attacks that demonstrate the difficulty of meeting the PCI-HSM requirement. We concluded the discussion on command manipulation attacks by suggesting a variety of defences that could be used to reduce the likelihood of such attacks being successful.

References

1. "Payment card industry PIN Security Requirements", version 1.0, September 2011.
2. ISO 9564–1, "Financial services - Personal Identification Number (PIN) management and security - Part 1: Basic principles and requirements for PINs in card-based systems", 2011.
3. ISO 9797–1, "Information technology - Security techniques - Message Authentication Codes (MACs) - Part 1: Mechanisms using a block cipher", 2011.
4. ANSI X9.24-1, "Retail Financial Services Symmetric Key management, Part 1: Using Symmetric Techniques", 2009.
5. ISO 13491–1, "Banking - Secure cryptographic devices (retail), Part 1: Concepts, requirements and evaluation methods", 2007.
6. ISO 13491–2, "Banking - Secure cryptographic devices (retail), Part 2: Security compliance checklists for devices used in financial transactions", 2005.
7. FIPS 140–2, "Security Requirements for Cryptographic Modules", 2001, with some updates in December 2002.
8. "Common Criteria for Information Technology Security Evaluation", see http://www.commoncriteriaportal.org/.
9. "Payment card industry (PCI) Hardware Security Module (HSM) Security Requirements", version 1.0, April 2009.
10. http://csrc.nist.gov/groups/STM/cmvp/documents/140-1/140val-all.htm.
11. https://www.PCIsecuritystandards.org/approved_companies_providers/approved_pin_transaction_security.php.
12. "Payment card industry (PCI): POS PIN Entry Device, Security Requirements", version 2.1, January 2009.
13. "PIN Security Program: Auditor's Guide", version 2, January 2008, see http://usa.visa.com/download/merchants/visa_pin_security_program_auditors_guide.pdf.
14. ANSI X9.17, "Financial institution key management (wholesale)", 1985.
15. ANSI X9 TR-31, "Interoperable Secure Key Exchange Key Block Specification for Symmetric Algorithms", 2010.
16. M. Bartolozzo, R. Focardi, M. Centenaro & G. Steel, "Attacking and Fixing PKCS#11 Security Tokens", ACM Conference on Computer and Communications, Security, 2010, pp. 260–269.
17. PKCS#11, "Cryptographic Token Interface Standard", version 2.20, RSA Laboratories, June 2004.
18. R. Anderson, "Why cryptosystems fail", Proceedings of the 1993 ACM Conference in Computer and Communications Security, pp. 215–227. See also, http://www.cl.cam.ac.uk/users/rja14/wcf.html.
19. R. Anderson, "Security Engineering", (2nd Edition), Wiley, 2008.
20. J. Clulow, "The Design and Analysis of Cryptographic Application Programming Interfaces for Security Devices", version 4.0, M.Sc. Thesis at University of Natal, Durban, South Africa, dated 17 January 2003.
21. Y. Desmedt, F. Hoornaert & J.J. Quisquater, "Several Exhaustive Key Search Machines and DES", EUROCRYPT 86, 1986, pp 17–19.

22. R. Clayton & M. Bond, "Experience Using a Low-Cost FPGA Design to Crack DES Keys", presented at the CHES 2002 Workshop Francisco, 1st August. (http://www.cl.cam.ac.uk/rnc1/descrack/DEScracker.pdf).
23. M. Bond & P. Zieliński, "Decimalisation Table Attacks for PIN Cracking", University of Cambridge Computer Laboratory, Technical Report 560, dated February 2003. (http://www.cl.cam.ac.uk/TechReports/UCAM-CL-TR-560.pdf).
24. R. Anderson & M. Bond, "Protocol Analysis, Composability and Computation"; see http://www.cl.cam.ac.uk/rja14/Papers/bond-anderson.pdf.
25. Joint USSS/FBI Advisory February 2009, see http://usa.visa.com/download/merchants/20090212-usss_fbi_advisory.pdf.