INTERNATIONAL STANDARD

ISO 11568-2

First edition
1994-12-01

Banking — Key management (retail) — Part 2: Key management techniques for symmetric ciphers

Banque — Gestion de clés (services aux particuliers) —

Partie 2: Techniques de gestion de clés pour les algorithmes cryptographiques symétriques

Reference number
ISO 11568-2:1994(E)

Contents

Foreword

ISO (the International Organization for Standardization) is a worldwide federation of national standards bodies (ISO member bodies). The work of preparing International Standards is normally carried out through ISO technical committees. Each member body interested in a subject for which a technical committee has been established has the right to be represented on that committee. International organizations, governmental and non-governmental, in liaison with ISO, also take part in the work. ISO collaborates closely with the International Electrotechnical Commission (IEC) on all matters of electrotechnical standardization.

Draft International Standards adopted by the technical committees are circulated to the member bodies for approval before their acceptance as International Standards by the ISO Council. They are approved in accordance with ISO procedures requiring at least 75 % approval by the member bodies voting.

International Standard ISO 11568-2 was prepared by Technical Committee ISO/TC 68, *Banking and related financial services.* Subcommittee SC 6, *Financial transaction cards, related media and operations.*

ISO 11568 consists of the following parts, under the general title *Banking — Key management (retail) :*

— *Part 1 : Introduction to key management*

— *Part 2 : Key management techniques for symmetric ciphers*

— *Part 3 : Key life cycle for symmetric ciphers*

— *Part 4 : Key management techniques for asymmetric ciphers*

— *Part 5 : Key life cycle for asymmetric ciphers*

— *Part 6 : Key management schemes*

Annexes A, B and C form an integral part of this part of ISO 11568. Annexes D, E and F are for information only.

Introduction

ISO 11568 is one of a series of standards describing procedures for the secure management of cryptographic keys used to protect messages in a retail banking environment, for instance, messages between an acquirer and a card acceptor, or an acquirer and a card issuer. Key management of keys used in an Integrated Circuit Card (ICC) environment is not covered by ISO 11568 but will be addressed in another ISO standard.

Whereas key management in a wholesale banking environment is characterized by the exchange of keys in a relatively high-security environment, this standard addresses the key management requirements that are applicable in the accessible domain of retail banking services. Typical of such services are point-of-sale/point-of-service (POS) debit and credit authorizations and automated teller machines (ATM) transactions.

ISO 11568-2 describes key management techniques which, when used in combination, provide the key management services identified in ISO 11568-1. These services are :

— key separation

— key substitution prevention

— key identification

— key synchronization

— key integrity

— key confidentiality

— key compromise detection

The key management services and the corresponding key management techniques are cross referenced in clause 6.

Banking — Key management (retail) —

Part 2:
Key management techniques for symmetric ciphers

1 Scope

This part of ISO 11568 specifies techniques for the protection of the cryptographic keys for symmetric ciphers used in a retail banking environment.

It is applicable to any organization which is responsible for implementing procedures for the protection of keys during the life cycle. The techniques described in this part enable compliance with the principles described in ISO 11568-1.

The techniques described are applicable to any symmetric *n*-bit block cipher algorithm.

The notation used in this part of ISO 11568 is given in annex A.

Algorithms approved for use with the techniques described in this part of ISO 11568 are given in annex B.

2 Normative references

The following standards contain provisions which, through referenced in this text, constitute provisions of this part of ISO 11568. At the time of publication, the conditions indicated were valid. All standards are subject to revision, and parties to agreements based upon this part of ISO 11568 are encouraged to investigate the possibility of applying the most recent edition of the standards indicated below. Members of IEC and ISO maintain registers of currently valid International Standards.

ISO 8908:1993, *Banking and related fiancial services — Vocabulary and data elements.*

ISO/IEC 10116:1991, *Information processing — Modes of operation for an n-bit block cipher algorithm.*

ISO 11568-1:1994, *Banking — Key management (retail) — Part 1: Introduction to key management.*

ISO 11568-3:1994, *Banking — Key management (retail) — Part 3: Key life cycle for symmetric ciphers.*

ANSI X3.92, *Data Encryption Algorithm.*

3 Definitions

For the purposes of this part of ISO 11568, the definitions given in ISO 8908 and the following definitions apply.

3.1 cipher : A pair of operations which effect transformations between plaintext and ciphertext under the control of a parameter called a key. The encipherment operation transforms data (plaintext) into an unintelligible form (ciphertext). The decipherment operation restores the original text.

3.2 counter : An incrementing count used between two parties, for example, to control successive key distributions under a particular key encipherment key.

3.3 data integrity : The property that data has not been altered or destroyed in an unauthorized manner.

3.4 data key : A cryptographic key used for the encipherment, decipherment or authentication of data.

3.5 exclusive-or : see modulo-2 addition.

3.6 hexadecimal digit : A single character in the range 0-9, A-F (upper case), representing a four bit string.

3.7 key component : One of at least two randomly or pseudo-randomly generated parameters having the characteristics (e.g. format, randomness) of a cryptographic key that is combined with one or more like parameters, e.g. by means of modulo-2 addition, to form a cryptographic key.

3.8 key offset; offset : The result of adding a counter to a cryptographic key using modulo-2 addition.

3.9 key space : The set of all possible keys used within a cipher.

3.10 Message Authentication Code (MAC) : A code in a message between a originator and recipient used to validate the source and part or all of the text of a message. The code is the result of an agreed calculation.

3.11 modulo-2 addition : exclusive-or; XOR : A binary addition with no carry, giving the following values :

$0 + 0 = 0$

$0 + 1 = 1$

$1 + 0 = 1$

$1 + 1 = 0$

3.12 *n*-bit block cipher : A block cipher algorithm with the property that plaintext blocks and ciphertext blocks are *n*-bits in length.

3.13 notarization : A method of modifying a key encipherment key in order to authenticate the identities of the originator and the ultimate recipient.

3.14 originator : The party that is responsible for originating a cryptographic message.

3.15 pseudo-random : A process that is statistically random and essentially unpredictable although generated by an algorithmic process.

3.16 recipient : The party that is responsible for receiving a cryptographic message.

3.17 secure cryptographic device : A device that provides security storage for secret information such as keys and provides security services based on this secret information.

4 General environment for key management techniques

The techniques which may be used to provide the key management services are described in clause 5. This clause describes the environment within which those techniques operate and introduces some fundamental concepts and operations which are common to several techniques.

4.1 Functionality of a secure cryptographic device

The most fundamental cryptographic operations for a symmetric block cipher are to encipher and decipher a block of data using a supplied secret key. For multiple blocks of data, these operations might use a mode of operation of the cipher as described in ISO/IEC 10116. At this level, no meaning is given to the data, and no particular significance is given to the keys.

Typically, in order to provide the required protection for keys and other sensitive information, a secure cryptographic device must provide a higher level functional interface, whereby each operation includes several of the fundamental cryptographic operations using some combination of keys and data obtained from the interface or from an intermediate result. These complex cryptographic operations are known as functions, and each one operates only on data and keys of the appropriate type.

4.1.1 Data types

Application level cryptography assigns meaning to data, and data with differing meanings need to be manipulated and protected in different ways by the secure cryptographic device. Data with a specific meaning constitutes a data type. The secure cryptographic device ensures that it is not possible to manipulate a data type in an inappropriate manner.

For example, a PIN is a data type which must remain secret, whereas other transaction data may constitute a data type which requires authentication but not secrecy.

A cryptographic key may be regarded as a special data type. A secure cryptographic device ensures that a key can exist only in the forms permitted by ISO 11568-3.

4.1.2 Key types

A key is categorized according to the type of data on which it operates and the manner in which it operates. The secure cryptographic device ensures that key separation is maintained, so that a key cannot be used with an inappropriate data type or in an inappropriate manner.

For example, a PIN encipherment key is a key type which is used only to encipher PINs, whereas a key encipherment key (KEK) is a key type which is used only to encipher other keys. Also a KEK may require typing such that it operates only on one type of key. For example, one type of KEK may encipher a PIN encipherment key, while another enciphers a MAC key.

4.1.3 Cryptographic functions

The set of functions supported by the secure cryptographic device directly reflects the cryptographic requirements of the application. It might include such functions as : encipher a PIN, verify an enciphered PIN, generate a MAC, or generate an enciphered random key.

The design of the secure cryptographic device is such that no individual function may be used to obtain unauthorized sensitive information. Additionally, no combination of functions exists which may result in such data being obtained. Such a design is referred to as being logically secure.

A secure cryptographic device may be required to manage keys of several types. Cryptographic keys used in such a system may be held securely outside of the cryptographic device by being stored in an enciphered formby using KEKs which either exist only within the cryptographic device, or are enciphered under a higher level KEK.

One technique of providing key separation is to use a different KEK type for the encipherment of each type of key. When this technique is used, and an enciphered key is passed to the secure cryptographic device, the key is deciphered using the KEK type appropriate for the expected key type. If this key is an incorrect type, and thus is enciphered under some other KEK type associated with some other key type, the decipherment produces a meaningless key value.

4.2 Double length keys

The secret key used with a symmetric block cipher may at some future time be subjected to an exhaustive key search attack, whereby each key in the key space is tested in turn until the correct key is discovered. The attack requires knowledge of an amount of ciphertext and the corresponding (known or suspected) plaintext. Each test involves deciphering the ciphertext using a test key value and comparing the result with the corresponding plaintext.

The average time required to discover a key by exhaustive search is directly proportional to the size of the key space of the cipher.

An effective measure against this form of attack is the use of cryptographic operations which utilize a double length key. A double length key is denoted by an asterisk preceding the alphabetic characters of the key name, e.g. *K. A double length key may also be referred to as a key pair as it is formed by the concatenation of two single length keys, the left key and the right key, which are denoted using the suffixes 'l' and 'r' respectively. Hence ;

*K = Kl || Kr

The notation (*)K may be used where the context allows a single or double length key. Abbreviations used in this part of this ISO 11568 are described in annex C.

The encipherment and decipherment of a single block using a double length key are defined as shown in figure 1.

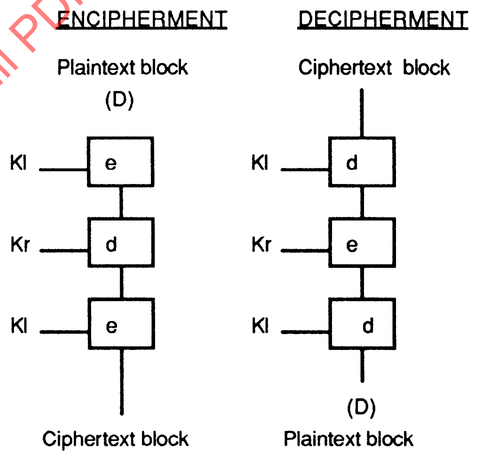

Figure 1 — Use of double length keys

This 'triple encipherment' operation is often written as 'ede' and the inverse operation of 'triple decipherment' is written as 'ded', that is;

ede*K(D) = eKl(dKr(eKl(D)))
ded*K(D) = dKl(eKr(dKl(D)))

Alternatively, the operators 'e' and 'd' may implicitly denote multiple encipherment and decipherment respectively when used with a double length key, that is;

e*K(D) = ede*K(D)
d*K(D) = ded*K(D)

These definitions for the encipherment and decipherment operations using a double length key have the characteristic that a double length key comprising a left (l) and right (r) key with equal values is equivalent to a single length key with that value.

4.3 Key generation

The key management principles in ISO 11568-1 require that keys shall be generated using a process which ensures that it is not possible to predict any key or determine that certain keys within the key space are more probable than others.

In order to conform with this principle, keys and key components shall be generated using a random or pseudo-random process. The pseudo-random key generation process may be either non repeatable or repeatable.

4.3.1 Non repeatable key generation

This process may involve a non deterministic value such as the output of a random number generator, or may be a pseudo-random process.

An example of a pseudo-random process for generating a key, "Kx", is as follows, where K is a secret cryptographic key reserved for key generation, V is a secret seed value and DT is a date-time vector updated on each key generation :

$$Kx = eK\,(eK\,(DT) \oplus V)$$

and generate a new V as follows :

$$V = eK\,(Kx \oplus eK\,(DT))$$

4.3.2 Repeatable key generation

It is sometimes convenient to generate one or more keys, perhaps thousands, from a single key using a repeatable process. Such a process allows for any of the resultant keys to be regenerated, as required, in any location which possesses the seed key and appropriate generation data, and facilitates significant reductions in the number of keys which require manual management, storage or distribution.

The generation process shall be such that if the initial key is unpredictable within the key space (as required by the key management principles), then so is each resultant key.

The procedure may be used iteratively, as a key generated from one initial key may subsequently be used as a initial key to generate others.

The generation process shall be non reversible, such that disclosure of a generated key discloses neither the initial key nor any other generated key. An example of such a process is the encipherment of a non secret value using the initial key.

4.4 Key calculation

It is possible to obtain a number of keys from a single key using a reversible process. An example of such a process is the modulo-2 addition of the key and a non secret value.

Key calculation has the qualities of speed and simplicity, but disclosure of one key calculated in this manner discloses the original key and all other keys calculated from it.

4.5 Key hierarchies

A key hierarchy is a conceptual structure in which the confidentiality of certain keys is dependent upon the confidentiality of other keys. By definition, disclosure of a key at one level of the key hierarchy shall not disclose any key at a higher level.

Key encipherment introduces a key hierarchy whereby a key encipherment key (KEK) is considered to be at a higher level than a key which it enciphers. The most simple is a two level hierarchy, whereby the working keys are enciphered by KEKs which are themselves stored in a cryptographic device. In a three level hierarchy, these KEKs are also managed in an enciphered form using a higher level KEK. The concept may be extended to four or more layers.

Similarly, when an initial key or key generating key (KGK) participates in the generation of other keys using a deterministic process, a hierarchy may result whereby the KGK is considered to be at a higher level than the generated keys.

Keys at the higher levels of the key hierarchy tend to be long-term keys and may protect a large number of other keys from disclosure, therefore they should be double length keys.

5 Techniques for the provision of key management services

This clause describes the techniques which may be used, individually or in combination, to provide the key management services introduced in ISO 11568-1. Some techniques provide multiple key management services. A cross reference between the key management services and the techniques is given in table 1, and an example of a typical symmetric cipher key management scheme is shown in annex D.

The selected techniques shall be implemented in a secure cryptographic device. The functionality of the cryptographic device ensures that the implementation of a technique is such that the intended purpose of the technique is achieved.

5.1 Key encipherment

Key encipherment is a technique whereby one key is enciphered using another key. The resulting enciphered key may then be managed securely outside of the protected environment of a cryptographic device. A key used to perform such encipherment is called a key encipherment key (KEK).

Keys may need to exist outside of a cryptographic device, for the following reasons :

a) where a key is required to be transported securely to a remote cryptographic device using a non secure channel ;

b) where the key storage requirement of the node exceeds the facilities provided by the cryptographic device.

KEKs tend to be long term keys and may protect a large number of other keys from disclosure. Therefore the KEKs should be double length keys.

Encipherment of a single key using a double length key is as described in 4.2 with the single key being the plaintext data block.

Encipherment of a double length key using a double length key is performed by independently enciphering the left and right keys, i.e.

*KEK(*K) = e*KEK(Kl) || e*KEK(Kr)

No key should be enciphered using a key of a lesser length. This implies that a double length key used with a particular cipher should not be enciphered using a single length key of that cipher.

Although key encipherment ensures key confidentiality is maintained, other techniques may need to be employed in association with the key encipherment in order to ensure adequate key separation and to prevent key substitution.

5.2 Key variants

Key variants is a technique for obtaining a set of keys from a single key, with each resulting key having a different key type.

This technique provides key separation while eliminating the need to manage a separate, unrelated key of each required type. Each variant key is calculated from the original key and one constant from a set of non secret constants, as illustrated in figure 2, using a repeatable process (f). The process of repeatable key calculation is described in 4.4.

A constant having a unique value in the set of constants shall be allocated to each key type to be calculated from the original key using the key variants technique.

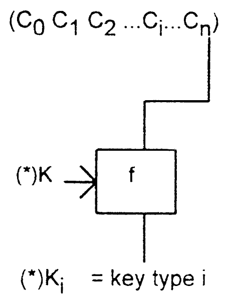

Figure 2 — Variant key calculation

A variant key calculated using a reversible process shall exist only in the cryptographic device which contains the original key.

The key variants technique is applicable at all levels of a key hierarchy. A single key may be used to calculate a set of KEKs of different types, i.e. each KEK is to be used to encipher a different key type. Alternatively, a single key may generate a set of working keys of different types.

NOTE 1 For additional explanation on key variants and control vectors, refer to annex E.

5.3 Key derivation

Key derivation is a technique by which a (potentially large) number of keys are generated ("derived") from a single initial key and non secret variable data, with each resulting key being the initial key for a different secure cryptographic device — typically the PIN pad of an EFTPOS terminal. The initial key is called a "derivation key" and each key generated from it is called a "derived key". Key derivation provides key separation by generating a (statistically) unique key for each cryptographic device, without the need to manage a large number of separate, unrelated keys. This eliminates the need to store each initial key (presumably in enciphered form) at the acquirer or receiving node, as when a key is needed for subsequent use it is derived again from the derivation key and appropriate derivation data.

The derived key generation procedure utilizes a non reversible process, as illustrated in figure 3, using the derivation key and data which uniquely identifies the target cryptographic device.

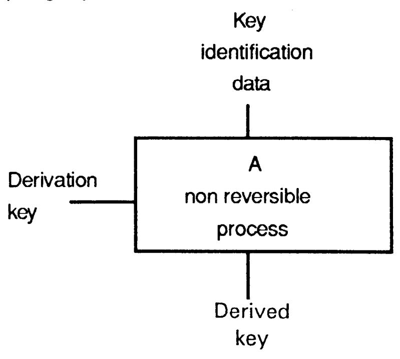

Figure 3 — Generation of a derived key

The use of a non reversible process ensures that the compromise of a derived key does not disclose the derivation key or any other derived keys. However, compromise of a derivation key discloses all keys derived from it.

NOTE 2 The procedure of key derivation as defined above excludes a key variant (see 5.2) from being considered as a derived key.

5.4 Key transformation

Key transformation is a technique whereby a secure cryptographic device - typically the PIN pad of an EFTPOS terminal — effects key replacement by generating one or more future keys from its current key and then erasing all trace of the current key.

Key transformation lessens the impact of key compromise by enabling the cryptographic device to replace its key(s) at frequent intervals (e.g. after every transaction) without the need for a key distribution process.

Key transformation is accomplished by using the current key as the initial key for a non reversible key generation process, as illustrated in figure 4. The process also involves other data relevant to the implementation which may be device or transaction related data, or may be a key replacement counter.

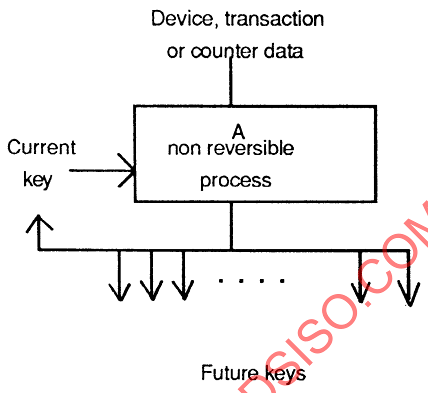

Figure 4 — Generation of future keys

The use of a non reversible process ensures that the compromise of a cryptographic device using transformed keys does not disclose any key previously used by the device, even given the knowledge of all relevant data which has ever existed outside of the device other than within another cryptographic device. Equipment in cryptographic communication with a device which utilizes key transformation determines the key in use by the device at any time by either :

a) replicating the key transformation process in synchronization with the device, and storing the resultant key(s) for subsequent use, or

b) receiving the current value of the key replacement counter, which is included in the transaction data transmitted from the device, and generating the current key from the counter and the initial key. This procedure involves performing all those key transformations which lead directly from the initial key to the current key. Typically, the number of transformations required is small (e.g. 10), even though the key replacement counter increments to a large (e.g. 1 million) value.

5.5 Key offsetting

Key offsetting is a technique for calculating a new KEK from an initial key each time a new enciphered key is to be transmitted to a receiving node.

The technique prevents the substitution of a previous (but replaced) key for the current key exchanged between communicating parties.

A counter, which is incremented each time a replacement key is required, is combined with the initial KEK using a repeatable process, (e.g. modulo-2 addition). The resulting key is the KEK with which the replacement key is enciphered, as illustrated in figure 5.

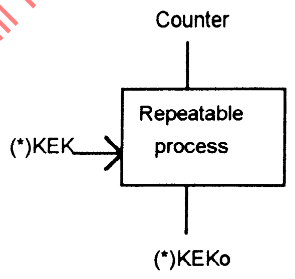

Figure 5 — Calculation of a KEK offset (KEKo)

Typically, the current value of the counter is transmitted to the receiving node along with the enciphered key.

The counter is typically the same length as a key, and for a double length key is independently combined with the left and right keys. The initial key shall itself be replaced before the counter resets.

Key offsetting is fully described in ISO 8732.

5.6 Key notarization

Key notarization is a technique whereby the identities of the communicating parties are incorporated into the KEK(s) before key encipherment occurs.

Key notarization prevents key substitution. Should an enciphered key intended for use between other parties be substituted for the correct enciphered key, the decipherment of the substituted key would produce a garbled result.

Key notarization is fully described in ISO 8732.

5.7 Key tagging

Key tagging is a technique whereby a key existing outside of a cryptographic device has an associated tag which identifies the key type. A key and its tag are bound together in a manner which prevents undetectable modification of the key tag.

Key tagging provides key separation. It is used in combination with key encipherment, and allows multiple key types to be enciphered using a single KEK.

A key tag consists of a distinct but otherwise arbitrary constant. A different key tag shall be allocated to each key type to be tagged. When a key is generated in the secure cryptographic device, it is bound together with the tag appropriate to the key type as part of the key encipherment process, as illustrated in figure 6. The tagged key may then be stored or distributed outside of a cryptographic device.

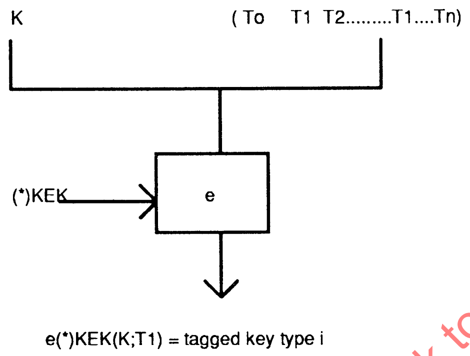
K
(To T1 T2.........T1....Tn)
(*)KEK
e
e(*)KEK(K;T1) = tagged key type i

Figure 6 — Tagged key generation

When a tagged key is passed into a secure cryptographic device for use in a specific function, it is deciphered and the key tag is recovered and checked within the device, as illustrated in figure 7. If the key tag reveals that the type is not appropriate to the requested function, the function shall be aborted.

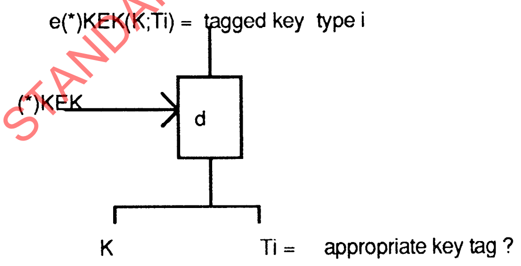

Figure 7 — Tagged key use

The manner in which the key and its tag are bound is dependent on whether or not their combined length exceeds the block length of the cipher used for key encipherment. If the block length is not exceeded, the key bits and tag bits may be concatenated or interleaved and the resultant block enciphered. If the block length is exceeded, the key and tag occupy separate blocks for encipherment. In this case, a chaining mode of encipherment (see ISO 10116) shall be used in order to bind the key and its tag together.

5.8 Key verification

A key verification code (KVC) is a value cryptographically related to the key and some additional non-secret information. The KVC is used at some later date to prove one or more of the following conditions have been met :

a) A key has been correctly entered into a cryptographic device ;

b) A key has been correctly received over a communications channel ;

c) A key has not been altered ;

d) An instance of a key remains in synchronization.

A common use of KVCs is to find where keys have been altered or corrupted in a network. Any key failing the KVC test shall be suspected of having been corrupted.

An example of a common KVC implementation is to encipher a fixed commonly known, non secret constant (e.g. a 16 hexadecimal value), under the key in question (see figure 8), then truncate the resulting ciphertext (e.g. six digits).

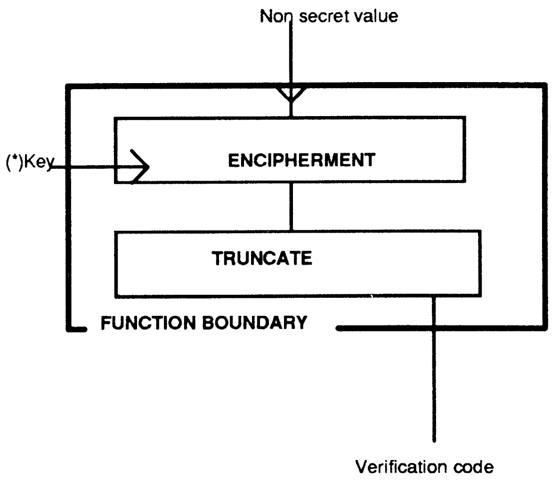

Figure 8 — Example of KVC function

The KVC is transported either electronically or by other means to the location where verification will take place and is entered into the cryptographic processor. The key that it is verifying may be sent to this location by the same or by a different route. The same cryptographic function is then performed on the received key and the output ofthis is compared to the key verification code.

The known, non secret value should be supplied by the cryptographic device that generates the KVC, and the KVC should be substantially truncated (e.g. to six digits). Otherwise the KVC-produced capability could be misused to encipher known plaintext.

When a double length key is used, the key verification code may be obtained by replacing the "encipher" operation in figure 8 with an "encipher/decipher/encipher" operation.

5.9 Key identification

Key identification enables the transaction recipient to determine the appropriate key(s) associated with the transaction. Two methods are available.

5.9.1 Explicit key identification

Explicit key identification is the inclusion in a transaction message of information that identifies the key(s) used to encipher or authenticate data in that message. The technique is especially useful when many cryptographic devices, all with unique keys, communicate with a single facility. This is because it enables this facility to determine the key(s) to be used with any given message it receives.

5.9.2 Implicit key identification

With implicit key identification, the key used with a message is determined from other message-related information such as the terminal identifier or the communications line over which the message is received.

5.10 Controls and audit

It is not always possible or feasible to prevent security compromises. However, the effects of a compromise can often be averted if the compromise is detected. It is especially important

a) to detect the disclosure of a key ; and

b) to detect the substitution of a disclosed key for a legitimate key.

Key disclosure can be detected by :

a) Audits and controls imposed on those individuals who manage keys and/or cryptographic devices, to detect possible collusion between such people that might have resulted in key disclosure ;

b) Periodic inspection of and control over interfaces through which unenciphered keys or key components are transferred, to detect 'bugs' or 'taps' that might have resulted in the disclosure of transferred keying material ;

c) Careful control and auditing of cryptographic devices that contain keys, to detect any lost or stolen devices. Automated and manual techniques to audit and control cryptographic devices are specified in this part of ISO 11568. If it is known or suspected that a cryptographic device has been lost or stolen, the keys contained in the device are considered to have been disclosed.

Key substitution (of a disclosed key) can be detected by :

a) When explicit key identification is used, verifying that a just-received key identifier has the expected value ;

b) Maintaining a 'negative file' of keys whose disclosure is known or suspected, and at appropriate occasions performing audits of presumed valid keys to ensure that none is listed on the negative file. When explicit key identification is used, the key identifier of the disclosed key can be used to identify the key in the 'negative file' entry. Otherwise the enciphered version of the key itself can be used to identify the disclosed key in the 'negative file' entry ;

c) Performing periodic audits of presumed-valid keys to ensure that no key is associated with more than one cryptographic device (the audit can use enciphered keys, or key identifiers). This detects the replication of a disclosed key.

Unauthorized key modification, deletion, and insertion, as well as some instances of key substitution, that have occurred in association with a cryptographic device (e.g. a device that enciphers) are detected when the corresponding cryptographic device that performs the inverse operation (e.g. deciphers) produces a garbled or otherwise invalid result.

The above audits and controls also serve to deter, and thus prevent, some security compromises. Furthermore, an audit of a cryptographic device's functionality can prevent key misuse, by ensuring that the device is capable of performing only authorized functions.

6 Key management services cross reference

The purpose of the matrix given in table 1 is to provide a quick cross reference between the security services defined in ISO 11568-1 and the techniques as detailed in clause 5 in this part of ISO 11568. It should be noted however, that the validity of the service/technique will ultimately depend upon the specific user implementation.

Table 1

Techniques \ Services	Separation	Substitution prevention	Identification	Synchronization (availability)	Integrity	Confidentiality	Compromise detection
Encipherment						5.1	
Variants	5.2						
Key derivation	5.3	5.3		5.3			
Key transformation	5.4	5.4		5.4			
Offsetting		5.5					
Notarization		5.6					
Tagging	5.7						
Key verification		5.8		5.8	5.8		
Key identification			5.9				5.9
Audit/control							5.10

Annex A

(normative)

Notation used in this part of ISO 11568

A.1 Operators

Operators are represented by the following :

e	encipherment
d	decipherment
||	concatenation
⊕	modulo-2 addition

A.2 Suffix letters for keys

Suffix letters are defined as follows :

o	key has been key offset
l	key is the left key of a key pair
r	key is the right key of a key pair
S	keys being sent
R	keys being received

Otherwise, the letter is unassigned and may be used for any purpose.

A.3 Specific keys and key pairs

Keys may be single keys or may be used as key pairs. A single key is expressed as :

K = Key

whereas :

*KEK = KEKl || KEKr

is a key pair consisting of two quantities, part "l" (left) and part "r" (right). An asterisk "*" preceding the character sequence is used to specify that the key is a key pair.

Annex B

(normative)

Approved algorithms for symmetric key management

The techniques for key management described in this part of ISO 11568 involve cryptographic operations. The following algorithm(s) and modes of operation have been approved for use in conjunction with these techniques. The procedure for algorithm approval is described in ISO 11568-1:1994, annex A.

[1] ANSI X3.92, *Data Encryption Algorithm.*

[2] ISO/IEC 10116:1991, *Information processing — Modes of operation of an n-bit block cipher algorithm.*

Annex C

(normative)

Abbreviations

The following abbreviations are used in this part of ISO 11568.

Abbreviation	Meaning	Description
C	Constant value	A value that once agreed will not change during the operation being performed
CV	Control vector	Information necessary to provide control of key separation and key usage
d	Decipherment	A decipherment operation
D	Plaintext data block	Intelligible text that has meaning and can be read
DEA	Data encryption algorithm	see ANSI X3.92
ded	Decipher-encipher-decipher	A multiple operation used during decipherment of a single block using a double length key
ede	Encipher-decipher-encipher	A multiple operation used during encipherment of a single block using a double length key
e	Encipherment	An encipherment operation
f	Function	The interaction of data, cryptographic key, and a cipher operation
K	Cryptographic key	—
KD	Data key	A key used to encipher/decipher data
KEK	Key encipherment key	A cryptographic key used for the encipherment and decipherment of cryptographic keys
KGK	Key generating key	A cryptographic key used to ensure an unpredictable bit stream when generating other cryptographic keys (also known as a 'initial key')
KM	Master key	A cryptographic key used to encipher and decipher key encipherment keys
KMAC	MAC key	A key used for Message Authentication Code purposes
KPE	PIN encipherment key	A key used to encipher PINs
KVC	Key verification code	A code produced by enciphering a value for later comparison to verify a key
MAC	Message authentication code	—
PIN	Personal identification number	—
T	Time	A time parameter used as a variable input to a process or function
XOR	Modulo-2 addition	Binary addition without carry

Annex D

(informative)

Symmetric cipher examples

Figure D.1 illustrates the various keys which might be used with symmetric ciphers in a typical retail banking environment. The keys are those used at node B, and each such key is labelled "S" (send) or "R" (receive) from the perspective of node B. Table D.1 indicates which functions are valid and which are invalid for each of these node B keys.

Node B must be implemented in such a way that a key can only be used for its intended purpose. Thus this node must implement key separation, by one or more of the above techniques, to prevent a key of one type from being used instead of a key of another type, for example to prevent a MAC key from being used as a PIN encipherment key. The node must also implement one or more of the above techniques to prevent key substitution, to prevent a key which is intended for use with one party (e.g. node A) from being used with another party (e.g. node C), for example, to prevent "KDR A→B" from being used for "KDR C→B".

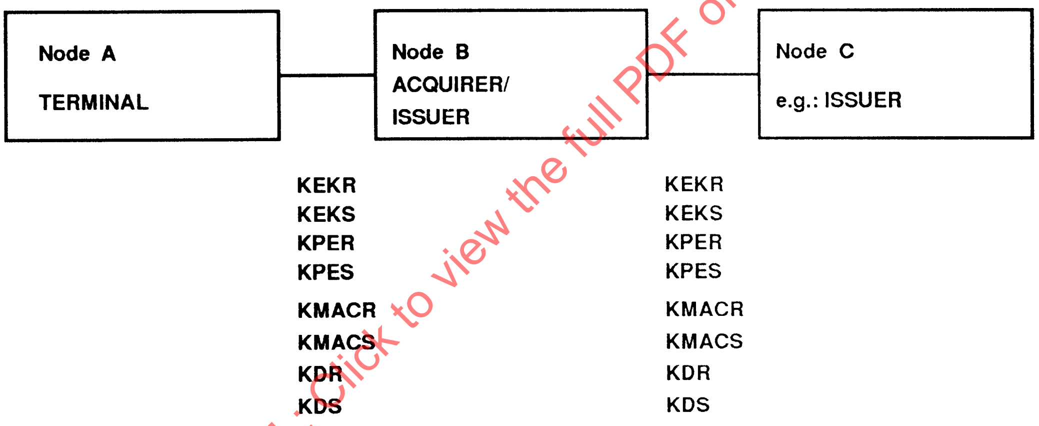

Figure D.1 — Example of a symmetric cipher key management system

Table D.1 — Examples of key usage at node B

FUNCTION								
KEY ENCIPHERMENT			PIN		MAC		Data	
	e	d	Translate	Validate	Generate	Validate	e	d
KEKR A→B		V 2						
KEKR C→B		V 2						
KEKS B→A	V 2							
KEKS B→C	V 2							
KPER A→B			V 3	V 1				
KPER C→B			N 5	V 4				
KPES B→A			N 5					
KPES B→C			V 3					
KMACR A→B						V 6		
KMACR C→B						V 6		
KMACS B→A					V 6			
KMACS B→C					V 6			
KDR A→B								V 7
KDR C→B								V 7
KDS B→A							V 7	
KDS B→C							V 7	

Legend:

I - Invalid

V - Valid

NOTES

1 The only functions relating to PINs that should be permitted are PIN translate (i.e. translated from keys from node A to node C) and PIN validate (in the case of node B being an issuer or Issuers' agent of transactions to the Issuer).

2 Principles to prevent substitution should ensure that only keys of the proper types are transferred using these functions.

3 Keys are input to crypto facility together to give the translate function.

4 Transactions arriving at node B, who is an issuer for transactions originating at (or past) node C.

5 PINs will never be sent to the terminal (node A).

6 Combined MAC translate function.

7 Combined data translate function.

Annex E
(informative)

Key variants and control vectors

E.1 Key variants

A secure cryptographic device may manage a large number of keys of several types which are stored in an enciphered form outside of the device, using KEKs which are stored internally. To provide key separation, a different KEK is used for the encipherment of keys of each type.

In this example, the set of KEKs used by the cryptographic device can be variants of a single initial key. This key may be the only key stored in the cryptographic device. When a function needs to obtain a specific KEK in order to decipher a supplied enciphered key, it selects the constant value appropriate to the key type, uses it and the stored seed key to calculate the KEK.

E.2 Control vectors

Control vectors provide separation and control of key usage for the purpose of ensuring keys are used only for their intended purpose. The control vector is a data element that contains, in encoded form, key related information necessary to control key processing. To ensure that key usage information is cryptographically linked to the key, the key is enciphered with a key encipherment key (KEK) via a process which combines the KEK and the control vector to produce a variant key under which the key is enciphered. Upon recovery, the KEK is again combined with the control vector to produce the same variant key, which is then used to decipher the enciphered key. Since the decipherment of the key occurs entirely within a secure cryptographic device, use of a secret KEK provides a process that the user cannot perform independently. The control vector restricts the process to one 'type' of key while maintaining key secrecy.

The key related information is incorporated into the control vector by assigning fields of the control vector to assume control of various key 'attributes'. For example, the type of key is restricted by encoding certain bits to represent a data key, PIN encipherment key, key encipherment key. Other bits control in what functions this key may participate, e.g., encipherment, decipherment, PIN verification. Additional bits control how a key may be distributed on a network or how it can be used once it is received. As with any key separation technique, both parties must participate in the rules of the key separation technique.

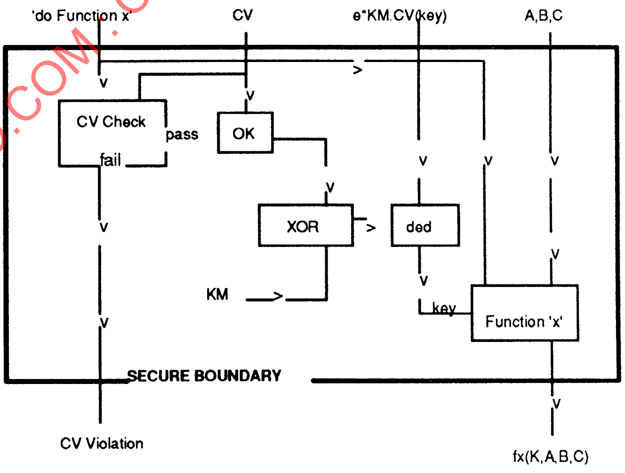

Figure E.1 — Example of Control Vectors for 'function x'

Shown in figure E.1 is an example of how control vectors are used. A request "to do" function x (one of perhaps many functions) is made to the cryptographic device. The control vector and the enciphered form of the key (e*KM.CV(Key)) and additional parameters and data A,B,C required to do function x is supplied. In this case the 'key' is protected under the master key (KM). It could also be protected under a KEK which in turn is enciphered under the KM using a control vector (CV) appropriate for the requested function. If the CV is inconsistent with the requested function, the operation is terminated. If the CV is consistent, the operation continues with the CV being XOR'd with KM and this variant of KM being used to recover the 'key' to complete function x. If a CV is supplied that passes the CV check but the enciphered form of the key is not permitted to participate in function x, the recovered 'key' will be meaningless and the results of function x useless.