INTERNATIONAL STANDARD

ISO 15668

First edition
1999-12-01

Banking — Secure file transfer (retail)

Banque — Transfert de fichier de sécurité (services aux particuliers)

Reference number
ISO 15668:1999(E)

Contents

Page

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.

International Standards are drafted in accordance with the rules given in the ISO/IEC Directives, Part 3.

Draft International Standards adopted by the technical committees are circulated to the member bodies for voting. Publication as an International Standard requires approval by at least 75 % of the member bodies casting a vote.

Attention is drawn to the possibility that some of the elements of this International Standard may be the subject of patent rights. ISO shall not be held responsible for identifying any or all such patent rights.

International Standard ISO 15668 was prepared by Technical Committee ISO/TC 68, *Banking, securities and other financial services*, Subcommittee SC 6, *Retail financial services*.

Annexes A to D of this International Standard are for information only.

Introduction

This International Standard describes how to secure file transfers in a retail banking environment. Typical use of such file transfers are those between a card accepting device and an acquirer, or between an acquirer and a card issuer.

Banking — Secure file transfer (retail)

1 Scope

In contrast to file transfers in a wholesale banking environment characterised by exchanges of large volume, between mainframes, in a relatively high-security environment ("bulk file transfers"); those in a retail banking environment are characterised by low volumes and a lower degree of reliability of environment in which downloaded devices are operated. Such devices may be, but not limited to, an electronic point of sale terminal (EPOS), an automated vending machine (AVM), an automated teller machine (ATM), or a merchant server in communication with payment gateways.

It is assumed that a pre-established relationship exists between the entities involved in the secure file transfer, especially to cover the legal and commercial aspects related to the file transfer liabilities.

This International Standard applies to the different kinds of file transfer used in retail banking environment, but does not cover transaction messages identified in ISO 8583.

The transfer may require timeliness, and requires at least one of the following security services:

— message origin authentication;

— receiver authentication;

— integrity;

— confidentiality;

— non repudiation of origin;

— non repudiation of delivery;

— auditability.

It is assumed that all data forwarded by the originator shall have been confirmed as legitimate and correct prior to the transfer.

The different types of files to be transferred could contain:

— software;

— the retail transactions which have been performed and registered, (uploading);

— technical data related to an acquirer (access parameters...), (downloading);

— application data related to an acquirer (BIN list, hot list, ...), (downloading).

Characteristics of such file transfers are the following:

a) the type of data to be transferred can be

— non-secret data (collection of retail transactions, technical data and application data); or

— secret data.

b) the number of entities to receive the data can be:

— one;

— more than one (broadcast with even thousands of receivers).

c) the communication channels can consist of one or both of the following examples:

— telecommunication: public network, private network;

d) the nature of the transfer can be:

— direct-connect, real-time transfer (also known as circuit switching); or

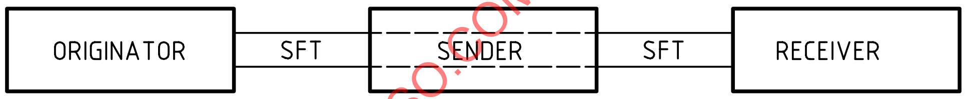
— store-and-forward transfer (also known as message switching).

NOTE This International Standard considers the security service during the transfer. Requirements to ensure that transferred files have not been altered after transfer achievement are outside the scope of this International Standard.

Permissible forms of Secure File Transfer

Transfer of Secured Files

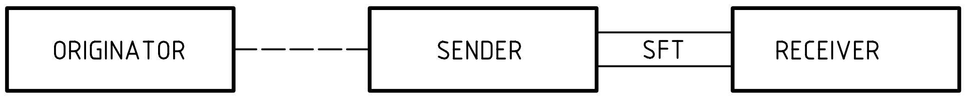
The transfer function does not provide any security services but includes only communication services. In this case the file shall be secured prior to the transfer. Security is managed by the originator and the receiver themselves. They need not trust the lower levels. There is no security added by the communication level (sender and receiver).



SFT = Secure File Transfer

Secured transfer of files

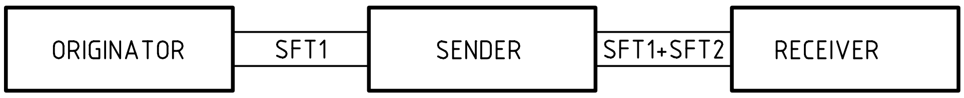
In this case, the security is taken into account only from the sender to the receiver and the originator fully trusts the sender. One example is where the originator is the sender and the security is delegated to the transfer level. This is not end to end security as there is no security added by the originator. In this case, the transfer function fully includes the security services. The file need not be secured prior to the secured transfer taking place.



Secured transfer of secured files

The security functions can be split up between the security function and the transfer function. One example is where the originator creates a file, signs it with the private signature key, and enciphers the file with a key known only by the end user (the receiver).

The concern in this example is to prevent anyone within the senders organisation from seeing the content of the originators file. However, the originator trusts its agent(s) to process the transfer and to take into account the authentication, integrity, between the sender and the receiver.



2 Normative references

The following normative documents contain provisions which, through reference in this text, constitute provisions of this International Standard. For dated references, subsequent amendments to, or revisions of, any of these publications do not apply. However, parties to agreements based on this International Standard are encouraged to investigate the possibility of applying the most recent editions of the normative documents indicated below. For undated references, the latest edition of the normative document referred to applies. Members of ISO and IEC maintain registers of currently valid International Standards.

ISO 8372:1987, *Information processing – Modes of operation for a 64-bit block cipher algorithm*.

ISO 8583:1993, *Financial transaction card originated messages – Interchange message specifications*.

ISO 8731-1:1987, *Banking – Approved algorithms for message authentication – Part 1: DEA*.

ISO 9564-1:1991, *Banking – Personal Identification Number management and security – Part 1: PIN protection principles and techniques*.

ISO/IEC 9796:1991, *Information technology – Security techniques – Digital signature scheme giving message recovery*.

ISO/IEC 9796-2:1997, *Information technology – Security techniques – Digital signature schemes giving message recovery – Part 2: Mechanisms using a hash-function*.

ISO/IEC 9797:1994, *Information technology – Security techniques – Data integrity mechanism using a cryptographic check function employing a block cipher algorithm*.

ISO/IEC 9798-1:1991, *Information technology – Security techniques – Entity authentication mechanisms – Part 1: General model*.

ISO/IEC 9798-2:1994, *Information technology – Security techniques – Entity authentication – Part 2: Mechanisms using symmetric encipherment algorithms*.

ISO/IEC 9798-3:1993, *Information technology – Security techniques – Entity authentication mechanisms – Part 3: Entity authentication using a public key algorithm*.

ISO/IEC 9798-4:1995, *Information technology – Security techniques – Entity authentication – Part 4: Mechanisms using a cryptographic check function*.

ISO 9807:1991, *Banking and related financial services – Requirements for message authentication (retail)*.

ISO/IEC 10116:1993, *Information technology – Modes of operation for an n-bit block cipher algorithm*.

ISO/IEC 10118-1:1994, *Information technology – Security techniques – Hash-functions – Part 1: General*.

ISO/IEC 10118-2:1994, *Information technology – Security techniques – Hash-functions – Part 2: Hash-functions using an n-bit block cipher algorithm*.

ISO 11568 (all parts), *Banking – Key management (retail).*

ISO/IEC 13888-2:1998, *Information technology – Security techniques – Non-repudiation – Part 2: Mechanisms using symmetric techniques*.

ISO/IEC 13888-3:1997, *Information technology – Security techniques – Non-repudiation – Part 3: Mechanisms using asymmetric techniques*.

NIST FIPS PUB 180-1, *Secure Hash Standard (Secure Hash Algorithm SHA-1)*.

3 Terms and definitions

For the purposes of this International Standard, the following terms and definitions apply.

3.1
hot list
a list of Primary Account Numbers (PAN) which are arranged according to the card issuer or its agent, not valid for transaction use

3.2
digital signature
data appended to, or a cryptographic formation of, a data unit that allows a recipient of the data unit to prove the source and integrity of the data unit and protect against forgery e.g. by the recipient

3.3
File Validation Value (FVV)
a derived value used for file validation purposes

3.4
hash code
the result of applying hash-function to data bits

3.5
hash-function
a (mathematical) function which maps values from a (possibly very) large set of values into smaller range of values, satisfying the following two properties within this International Standard:

— it is computationally infeasible to find for a given output an input which maps to this output;

— it is computationally infeasible to find for a given input a second input which maps to the same output.

3.6
application manager
the part of the software of a terminal which is in charge of verifying the secure downloading of the intended executable object

3.7
Message Authentication Code (MAC)
a code in a message between the originator and the receiver used to validate the source and part or all of the text of the message

NOTE The code is the result of an agreed calculation.

3.8
originator
the entity that creates the file that has to be transferred to the receiver and is responsible for its security

3.9
receiver
the entity that receives the file

3.10
sender
the entity that sends the file

3.11
sponsor
the entity that evaluates the risk of the file transfer

4 Principles

4.1 Message Origin Authentication

The purpose of message origin authentication is to provide assurance to the receiver that the alleged originator is the legitimate originator. It may also provide assurance to the receiver that the alleged file is the actual file, in the event that the receiver is authorised to receive certain files from only a subset of its authorised originators.

Message origin authentication may occur concurrent with the file transfer, though it may also occur before the transfer in a connection mode. If a store and forward transfer is used, the message origin authentication shall occur after the transfer, when the file is retrieved by the receiver. (Note: transfer of secured files in the model).

Techniques that provide content authentication may provide message origin authentication, but these techniques require the transfer of the entire file before the authentication can be verified. There may be situations in which it is desirable to perform message origin authentication prior to initiating the actual transfer. As an example, it may be desirable to prevent an impostor from masquerading as a legitimate file-provider and tying up a communications channel for an extended time in transferring a long file, even though the illegitimacy of the originator would eventually be detected and the transferred file rejected.

4.2 Receiver authentication

This security service authenticates the identity of the receiver prior to initiating the transfer, so that the transfer will not occur unless the receiver's identity has been verified.

Some receivers (POS terminals) are allowed to receive only some types of files. Part of the authentication process is the control by the originator of the rights for the receiver to receive some file-type.

Another possible reason for using "receiver authentication" is to prevent an unauthorised party from impersonating a legitimate receiver and tying up the originator's communications capabilities while the originator transfers a perhaps-lengthy file to the impersonator.

Receiver authentication does not prevent an unauthorised party from ascertaining the file contents (by "listening in"). Without this security service, and without the security service of confidentiality, anyone could easily impersonate the legitimate receiver and thus obtain the file. If it is essential that only the authorised receiver(s) of a file actually receive the file, then the security service of confidentiality must be used. Only in this way can it be ensured that an unauthorised party has not "listened in" on the communications channel and thus obtained a copy of the file.

Note that there is a related security service, non repudiation of delivery, that confirms, after the completion of the transfer, that the authorised party has successfully received the file.

4.3 Integrity

Accidental or unauthorised alteration of the transferred file, or, at a minimum, selected portions of the transferred file, shall be detected during and after the transfer process. Integrity services may control the entire file in a single process, or they may individually control segments of the file.

4.4 Confidentiality

When necessary, the confidentiality of the transferred files shall be ensured and shall be applied to either the whole file or those file portions that actually require confidentiality.

4.5 Non repudiation of origin

This security service provides evidence that the claimed originator actually originated the transferred file. (Without such evidence, the originator might falsely claim that the receiver created the file, or else the receiver might have created the file, and then falsely claimed that it came from the originator.)

4.6 Non repudiation of delivery

This security service provides evidence that the claimed receiver actually received the transferred file. Without such evidence, the receiver might claim non-receipt of the file. This service is achieved by mechanisms identified in annex A. The receiver sends to the originator a non repudiation token message proving that:

1) the intended receiver received the file;

2) the file contents were received unaltered.

NOTE The extent of non repudiation may be determined by domestic laws or local banking regulations.

4.7 Auditability

If required by the application, it may be necessary for the sender/originator and/or the receiver to log appropriate details of the transfer (time and date, nature of the files, volume of the files, version number...). Such logs may include failed attempts to transfer data, as well as successful attempts. In the event of attempted fraud, a log of failed transfers may assist in identifying the source of the attempt.

5 Application

5.1 Software downloading

5.1.1 Definition

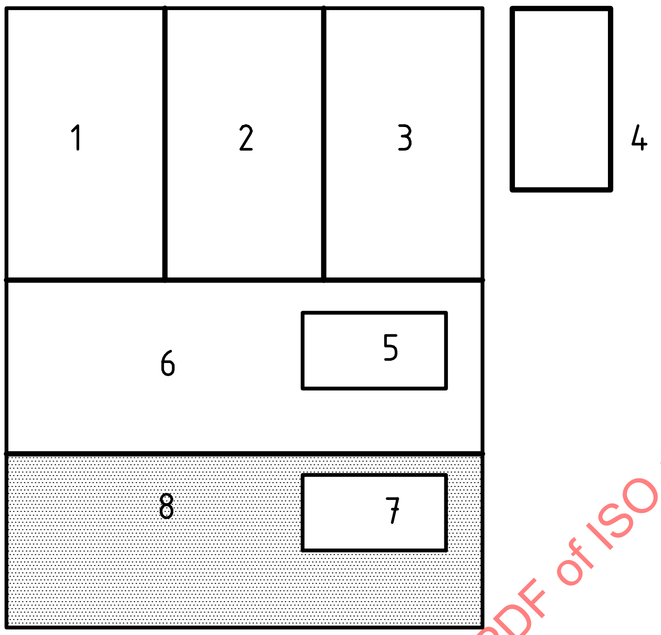
The software of a terminal is composed of distinct functional layers:

a) a trusted and fixed software, pre-loaded before secure file transfer, which is in charge of implementation of security mechanisms for downloading the application manager and its secure execution;

b) an application manager which is in charge of implementation of security mechanisms for downloading the different applications and their secure execution;

c) different applications, which consist of either executable objects or interpreted objects.

NOTE The above are representative of different layers within the device. Only layers 2 and 3 are downloadable The security of layer 1 is outside the scope of this International Standard.



Key

1 Application 1
2 Application 2
3 Application 3
4 Executable objects or interpreted objects
5 Keys
6 Application manager
7 Trusted, fixed and preloaded software (bootstrap)

Figure 1 — Representation of the software of a terminal

Security services required for software downloading are:

— mutual authentication or unilateral authentication of the originator/sender;

— integrity verification of the transferred file.

Confidentiality may be required for sensitive data when appropriate (e.g. keys).

Prior authentication of the sender by the device may not be required when the downloading is processed in a specific buffer and integrity checking is performed prior to accepting the software, achieving implicit authentication of the sender.

For each layer, the sequence of the operations shall be: mutual authentication, software transfer, integrity verification. In the case of mutual authentication failure, the software transfer shall not be performed. In the case of integrity failure, the software shall not be accepted by the device and the whole sequence shall be performed again.

NOTE It is not always necessary to authenticate the receiver. It may be sufficient for the receiver to have message origin authentication for downloaded software.

5.1.2 Mutual authentication

5.1.2.1 Implementation

Whatever entity is the initiator of the software download, mutual authentication shall be successfully performed prior to any file transfer at any layer:

— when the application manager is downloaded, it shall be protected by mutual authentication security services provided by the trusted and fixed software that has been pre-loaded;

— downloading of the application itself shall be protected by a mutual authentication service, implemented in the application manager and using dedicated keys to each authorised application.

Usually a sender may manage several applications, each of which may be transferred to devices. Part of the mutual authentication process is the verification by the sender of each receiving device and its rights to receive each application. This involves the following:

— each device shall be identified by a unique identifier;

— the sender associates a list of authorised applications to each device identifier.

5.1.2.2 Mechanism

The mutual authentication mechanism requires a 2 or 3-way message exchange and may be achieved by either a symmetric algorithm or an asymmetric algorithm. ISO/IEC 9798 specifies these security mechanisms.

When a symmetric algorithm is used, ISO/IEC 9798-2 and ISO/IEC 9798-4 apply. When an asymmetric algorithm is used, ISO/IEC 9798-3 applies.

The mutual authentication does not involve a Trusted Third Party and may be performed either by a two pass authentication using a sequence number (or time stamp), or a three pass authentication using random numbers.

For details, refer to annex A.

5.1.2.3 Key management

Key management techniques shall comply with ISO 11568.

Symmetric algorithm

The initial keys shall be securely installed in the device prior to any download. The same key may be used for both directions but it is recommended that the mutual authentication uses a unique key per device to prevent a device from masquerading as another one by changing its identification and moreover to prevent the possibility of creating a false sender in case of compromising the secret key of a device.

The keys used for mutual authentication, prior to downloading of each layer, may be different.

Asymmetric algorithm

The use of an asymmetric algorithm requires that the receiver shall have a trusted relationship with the sender and the keys used for mutual authentication are:

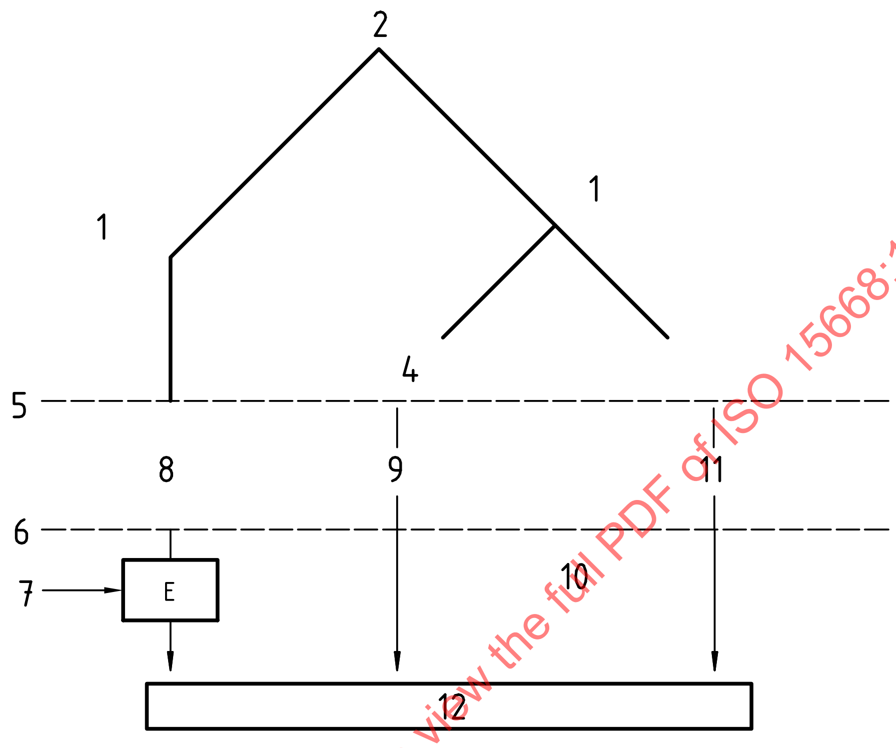
— a private key resident in the device and its associated public key which is transmitted to the sender to authenticate the terminal;

— one unique private/public key pair per layer in all devices.

The device private key may be the same for both layers, even if the senders are two different senders.

The public key used to authenticate the sender of the application manager shall be resident in each device and be protected against substitution. The public key used to authenticate the application sender shall be securely downloaded at the same time as the application manager or pre-loaded.

5.1.3 Integrity



Key

1	File	7	KEY
2	Non keyed algorithm	8	Hash code
3	Asymmetric	9	MAC
4	Symmetric	10	No further processing
5	Step 1	11	Digital signature
6	Step 2	12	File Verification Value (FVV)

Figure 2 — creation of File Verification Value

5.1.3.1 Implementation

The integrity of the transferred file shall be ensured by adding a File Verification Value (FVV) to the content of the file. The FVV of the file need only be computed once, as it is not required to be dependent on the downloading operation and on the identity of the receiving device.

After downloading of a layer and prior to activation of that layer on the device, the downloaded file FVV shall be verified by the device.

5.1.3.2 Mechanism

Depending upon the algorithm used, it involves one or more steps to create the File Verification Value (FVV), see Figure 2 and annex A.

5.1.3.3 Key management

The File Verification Value process requires the use of a secret key (symmetric algorithm) or a private key (asymmetric algorithm) or a non-keyed value (hash code). The FVV operation is performed by the device:

Symmetric algorithm

The secret keys used for integrity of each layer shall be different.

The keys used to verify the integrity of the downloaded application manager shall be securely installed prior to delivery of the device.

The keys used to verify the integrity of the downloaded application shall be:

— either securely installed prior to delivery of the device;

— or securely downloaded at the same time as the application manager.

The integrity and confidentiality of these keys shall be protected using a set of pre-installed keys.

Asymmetric algorithm

There is a unique public/private key pair per layer with the private key at the sender and the public key common to all receivers.

The public key used to verify the integrity of the downloaded application manager shall be resident in each device and protected against substitution. The public key used to verify the integrity of the downloaded application shall be securely downloaded at the same time as the application manager or pre-loaded.

Encipher HASH technique

The hash function is a two-phase operation.

The first step in creating the FVV is the computation of the hash on the whole file transferred, using a hash algorithm and without use of any (secret or otherwise) key.

The second step consists of applying a symmetric or a asymmetric algorithm to the hash, additional data and appropriate padding, to produce the FVV.

5.1.4 Confidentiality

Confidentiality may not be required for software downloading. However when confidentiality service is required for keys or data, a secret key or a private key is transferred from the sender through downloading and shall be protected by encipherment.

5.1.5 Non repudiation of delivery

After receiving the software and verifying its integrity, the device should send a secure acknowledgement to the sender, confirming the completion (successful or failed) of the reception and the integrity verification.

— The secure acknowledgement shall consist of a message identical to the authentication message sent by the device to the sender during the mutual authentication phase, using the same key, except that it shall include: the result: e.g. positive or negative;

— a time stamp.

The secure acknowledgement shall be in accordance with either ISO/IEC 13888-2 or ISO/IEC 13888-3.

On receipt of that secure acknowledgement, the sender shall verify and record it.

5.1.6 Non repudiation of origin

Non repudiation shall be implemented to ensure the integrity of the service.

When an asymmetric algorithm is used to ensure the integrity service, it provides, at the same time, non repudiation of origin, as only the originator has knowledge of the private key required to compute the signature. In this case, the device shall record the signature.

NOTE This implementation of the service is not fully performed if the signed data does not include a time stamp.

5.1.7 Auditability

To ensure non repudiation services, logging of the information described in 5.1.5 and 5.1.6 shall be performed.

5.2 Parameter downloading

5.2.1 Definition

Parameter downloading includes application data (such as BIN list, floor limits), negative file (hotlist), transfer and updating of sender references (access parameters).

The application software of the receiver shall initiate such transfers, which may involve a completely different protocol and sender from the one used previously to download the application software.

Security services for parameter transfer are under the control of the originator/sender. Usually an application software supplier through an application server provides the application, and the parameters are provided by the originator through a parameter server, however the two entities may be the same. In this case, the security mechanisms and keys shall be implemented by the application provider through the application software in accordance with the sponsor specification.

Security services required for parameter downloading are:

— mutual authentication or unilateral authentication of the originator/sender;

— integrity verification;

— confidentiality when appropriate;

— non repudiation of delivery depending upon the type of the parameters.

The sequence of the operations shall be mutual authentication, parameters transfer, integrity verification, and non repudiation of delivery when needed.

In the case of mutual authentication failure, the parameters transfer shall not be performed. In the case of integrity failure, the parameters shall not be retained on the device and the whole sequence shall be performed again.

NOTE It is not always necessary to authenticate the receiver. It may be sufficient for the receiver to have message origin authentication for the downloaded parameters.

5.2.2 Mutual authentication

The mutual authentication requirements described for application software downloading shall apply. This file transfer may be considered as an additional layer of the application downloading operation.

5.2.3 Integrity

The integrity requirements described for application software downloading shall apply. This file transfer may be considered as an additional layer of the application downloading operation.

5.2.4 Confidentiality

The confidentiality service may be required to transfer parameters, such as hot lists, floor limits, depending upon the nature of the network.

When required, implementation of this service shall fulfil local regulations.

5.2.5 Non repudiation of delivery

If required by the sponsor, the non repudiation of delivery requirements described in application software downloading shall apply.

5.2.6 Non repudiation of origin

If required by the sponsor, the non repudiation of origin requirements described in application software downloading shall apply.

5.2.7 Auditability

To ensure non repudiation services, logging of the information described in 5.1.5 and 5.1.6 shall be performed.

5.3 Retail transaction uploading

5.3.1 Definition

For this environment, the originator/sender is the terminal or the merchant server.

Retail transaction uploading consists of sending, to the acquirer server, a file containing the financial transactions of the period and additional related data such as the number of financial transactions and the total amount.

Security services required for retail transaction uploading are:

— mutual authentication;

— integrity verification of the transaction file transferred.

The sequence of the operations shall be mutual authentication, transactions file transfer, and integrity verification by the server, sending of evidence of delivery by the server. In the case of mutual authentication failure, the transfer shall not be performed. In the case of integrity failure, the server shall reject the transactions and the mutual authentication shall be performed again.

5.3.2 Mutual authentication

5.3.2.1 Implementation

Retail transaction uploading shall be protected by a mutual authentication service implemented in the application layer.

5.3.2.2 Mechanism

As for software downloading, ISO/IEC 9798 applies.

The mechanism shall be independent of the device, because the same acquirer may collect transactions from devices provided by different manufacturers.

5.3.2.3 Key management

Symmetric algorithm

As the retail transaction uploading operation is under the control of the acquirer, the keys used for mutual authentication are acquirer dependent. They shall be resident in a protected location of the device, for example a Secure Application Module (SAM) as defined in ISO 10202-4:1996, *Financial transaction cards — Security architecture of financial transaction systems using integrated circuit cards — Part 4: Secure application modules*.

The same key may be used in both directions but a unique key shall be used for each device. The key may be securely transferred as part of the parameter downloading process.

Asymmetric algorithm

The keys used for mutual authentication are:

— a private key is resident in the device and its associated public key is transmitted to the sender to authenticate the terminal;

— a unique private key in the server and the related common public key in all devices.

The public key used to authenticate the server may be securely transferred as part of the parameter downloading process and protected against substitution.

5.3.3 Integrity

5.3.3.1 Implementation

The integrity of the retail transaction file shall be ensured by adding a check-value to the content of this file. The check-value shall be computed prior to the transfer.

5.3.3.2 Mechanism

The requirements defined in 5.1.3.2 apply and for appropriate mechanisms, refer to annex A.

5.3.3.3 Key management

Symmetric algorithm

As the retail transaction uploading operation is under the control of the acquirer, the keys used for integrity verification are acquirer dependent. They shall be resident in a protected location of the device, for example a SAM as defined in ISO 10202-4.

It is recommended that the integrity verification process use a unique key for each device. The key may be securely transferred as part of the parameter downloading process.

Asymmetric algorithm

The keys used for integrity verification should be a unique private key resident in a protected location of the device, and its associated public key, which is transmitted to the server, to verify the signature.

5.3.4 Confidentiality

When required, implementation of this service shall fulfil local regulations.

If the uploaded retail transactions contain a PIN, the PIN shall be enciphered according to ISO 9564-1.

5.3.5 Non repudiation of delivery

This service is not required.

5.3.6 Non repudiation of origin

This service is not required.

5.3.7 Auditability

This service is not required.

For an overview of the security services as specified for each application, see annex D.

6 Authentication mechanisms

The authentication values are either integrated in the file transfer protocol or transmitted along with service files whose transfers are separated from the transfer of the data file.

In a protocol, which does not integrate security parameters, an authentication exchange shall take place before transferring the file.

When a communication session has not been established prior to the file transfer, and if the authentication values are not integrated in the file transfer protocol, then each file transfer shall be linked to a prior authentication relationship. For example, a cryptographic way to establish this relation is to exchange a key during the authentication process and to use this key for securing the transfer.

This need does not exist when a formal session has been established prior to the transfer provided both parties trust the integrity of the connection. When a formal protocol has been established prior to the transfer and if the authentication values are not integrated in the file transfer protocol, when several successive files are transmitted during a single session, there is no need to repeat the authentication exchanges before each new transfer.

In a protocol that integrates security parameters, one of the following techniques shall be used to authenticate the sender of a file:

— the file is signed by the sender and the digital signature is added to the file;

— the file is enciphered by a symmetric algorithm;

— a MAC is added to the file.

For approved algorithms, refer to ISO 11568.

Annex A
(informative)

Mechanism examples

A.1 Authentication mechanisms

ISO/IEC 9798-1 to ISO/IEC 9798-4, specifies entity authentication mechanisms which can be applied to secure file transfer.

For approved algorithms, refer to ISO 11568.

A.1.1 Authentication mechanisms using symmetric encipherment algorithms and not involving a Trusted Third Party

ISO/IEC 9798-2 specifies entity authentication mechanisms using symmetric encipherment algorithms. It specifies exchanges of messages providing unilateral authentication and mutual authentication.

ISO/IEC 9798-2 is based on the sharing of a common secret authentication key and an entity corroborates its identity by demonstrating its knowledge of a secret authentication key. The method by which the secret key is known to the involved party is not specified.

The mechanisms specified require the use of time variant parameters such as time stamps, sequence numbers or random numbers. They protect against replay. Using random numbers requires one more pass than using time stamps or sequence numbers.

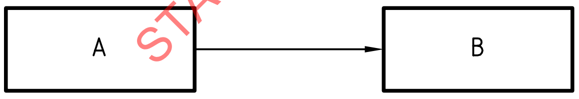
The use of text fields is optional and depends upon the application.

Unilateral authentication model

Unilateral authentication is used for transfer of retail payment data from a terminal to a host. The host authenticates the terminal.

1) One pass authentication

In this authentication mechanism the claimant A initiates the process and is authenticated by the verifier B. Uniqueness/timeliness is controlled by generating and verifying a time stamp or a sequence number. For details, see ISO/IEC 9798-2.



A sends to B the following token: Text1 // eK_{AB} (T_A or N_A // B // Text2).

K_{AB} : common secret authentication key shared by A and B.

$eK_{AB}(X)$: encipherment of data X with a symmetric algorithm using the key K_{AB}.

X // Y : concatenation of the data items X and Y.

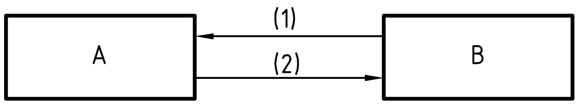
T_A : time stamp.

N_A : sequence number.

B : identifier of B (its inclusion is optional).

1) Two pass authentication

In this authentication mechanism the claimant A is authenticated by the verifier B who initiates the process. Uniqueness/timeliness is controlled by generating and verifying a random number. For details, see ISO/IEC 9798-2.



(1) B sends to A a random number: R_B and optionally a text field Text1.

(2) A sends to B the following token: Text2 // eK_{AB}(R_B // B // Text3).

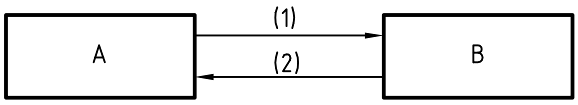
(the inclusion of B is optional).

Mutual authentication model

Mutual authentication is used for software application and secret data file transfer between a terminal and a host.

1) Two pass authentication

In this authentication mechanism uniqueness/timeliness is controlled by generating and verifying time stamps or sequence numbers. For details, see ISO/IEC 9798-2.



(1) A sends to B the following token: Text1 // eK_{AB}(T_A or N_A // B // Text2).

(2) B sends to A the following token: Text3 // eK_{AB}(T_B or N_B // A // Text4).

T_A, T_B : time stamps.

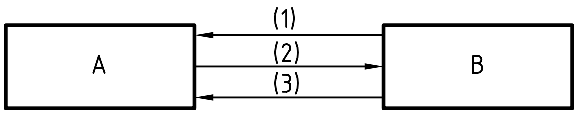
N_A, N_B : sequence numbers.

B : identifier of B (its inclusion is optional).

A : identifier of A (its inclusion is optional).

1) Three pass authentication

In this authentication mechanism uniqueness/timeliness is controlled by generating and verifying random numbers. For details, see ISO/IEC 9798-2.



(1) B sends to A a random number: R_B and optionally a text field Text1.

(2) A sends to B the following token: Text2 // eK_{AB}(R_A // R_B // B // Text3).

(the inclusion of B is optional).

(3) B sends to A the following token: Text4 // $eK_{AB}(R_B$ // R_A // Text5).

R_A, R_B : random numbers.

A.1.2 Application of ISO/IEC 9798-2 to secure file transfer

Depending upon the use of text fields within the token:

— if the enciphered text field is empty, the mechanism is used for authentication prior to the file transfer;

— if the enciphered text field equals the file to be transmitted, it corresponds to sending the enciphered file;

— if the enciphered text field is a symmetric secret key, key distribution is combined with authentication.

When using several keys for the same function, the cleartext text field may include the identifier of the key used to encipher the data or the identifier of the exchanged key or both.

A.1.3 Authentication mechanisms using asymmetric encipherment algorithms and not involving a Trusted Third Party

ISO/IEC 9798-3 specifies entity authentication mechanisms using asymmetric encipherment algorithms. It specifies exchanges of messages providing unilateral authentication and mutual authentication.

A digital signature is used to verify the identity of an entity: the entity to be authenticated corroborates its identity by using its secret key to sign specific data and therefore demonstrates its knowledge of its secret signature key. The verifying entity possesses the corresponding valid public key.

The mechanisms specified require the use of time variant parameters such as time stamps, sequence numbers or random numbers. Using random numbers requires one more pass than using time stamps or sequence numbers.

Unilateral authentication model

Unilateral authentication is used for transfer of retail payment data from a terminal to a host. The host authenticates the terminal.

1) One pass authentication

In this authentication mechanism the claimant A initiates the process and is authenticated by the verifier B. Uniqueness/timeliness is controlled by generating and verifying a time stamp or a sequence number. For details, see ISO/IEC 9798-3.

A sends to B the following token: CertA // T_A or N_A // B // Text1 // $sS_A(T_A$ or N_A // B // Text2).

CertA : A certificate (optional).

T_A : time stamp.

N_A : sequence number.

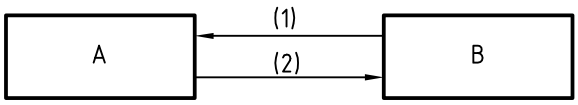
B : identifier of B.

sS_A : digital signature using A secret key.

X // Y : concatenation of the data items X and Y.

1) Two pass authentication

In this authentication mechanism the claimant A is authenticated by the verifier B who initiates the process. Uniqueness/timeliness is controlled by generating and verifying a random number. For details, see ISO/IEC 9798-3.



(1) B sends to A a random number: R_B and optionally a text field Text1.

(2) A sends to B the following token: CertA // R_A // R_B // B // Text2 // sS_A (R_A // R_B // B // Text3).

R_A, R_B : random numbers.

CertA : A certificate (optional).

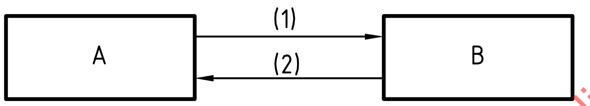
B : identifier of B.

Mutual authentication model

Mutual authentication is used for software application and secret data file transfer between a terminal and a host.

1) Two pass authentication

In this authentication mechanism uniqueness/timeliness is controlled by generating and verifying time stamps or sequence numbers. For details, see ISO/IEC 9798-3.



(1) A sends to B the following token: CertA // T_A or N_A // B // Text1 // sS_A(T_A or N_A // B // Text2).

(2) B sends to A the following token: CertB // T_B or N_B // A // Text3 // sS_B(T_B or N_B // A // Text4).

CertA : A certificate (optional).

CertB : B certificate (optional).

T_A, T_B : time stamps.

N_A, N_B : sequence numbers.

B : identifier of B.

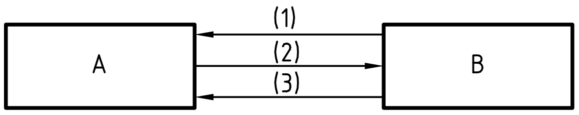
A : identifier of A.

sS_A : digital signature with A secret key.

sS_B : digital signature with B secret key.

1) Three pass authentication

In this authentication mechanism uniqueness/timeliness is controlled by generating and verifying random numbers. For details, see ISO/IEC 9798-3.



(1) B sends to A a random number: R_B and optionally a text field Text1.

(2) A sends to B the following token: CertA // R_A // R_B // B // Text2 // sS_A (R_A // R_B // B // Text3).

(the inclusion of B is optional).

(3) B sends to A the following token: CertB // R_B // R_A // A // Text4 // sS_B (R_B // R_A // A // Text5).

(the inclusion of A is optional).

R_A, R_B : random numbers.

A.1.4 Application of ISO/IEC 9798-3 to secure file transfer

Depending upon the use of text fields within the token:

- if the enciphered text field is empty, the mechanism is used for authentication prior to the file transfer;
- if the enciphered text field equals the file to be transmitted and if the digital signature allows message recovery (see clause Digital Signature), it corresponds to sending the enciphered file;
- if the enciphered text field equals the file to be transmitted and if the digital signature does not allow message recovery (see clause Digital Signature), the file is transmitted along with its signature;
- if the enciphered text field is a symmetric secret key and if the digital signature allows message recovery (see clause Digital Signature), key distribution is combined with authentication.

When using several keys for the same function, then the cleartext text field may include the identifier of the key used to sign the data or the identifier of the exchanged key or both.

A.1.5 Authentication mechanisms involving a Trusted Third Party

The mechanisms described in ISO/IEC 9798-2 and ISO/IEC 9798-3 may be adapted when using a TTP. The principle is to combine authentication with key exchange. When using symmetric encipherment algorithms, the sharing of a common secret authentication key by the two entities is not required, however each entity shares a secret key with the TTP and the entities request a session key from the TTP.

A.1.6 Authentication mechanisms using cryptographic check function

ISO/IEC 9798-4 specifies entity authentication mechanisms using a cryptographic check function. It specifies exchanges of messages providing unilateral authentication and mutual authentication.

ISO/IEC 9798-4 is based on the sharing of a common secret authentication key and an entity corroborates its identity by demonstrating its knowledge of a secret authentication key.

The method by which the secret key is known to the involved party is not specified.

The mechanisms specified require the use of time variant parameters such as time stamps, sequence numbers or random numbers. Using random numbers requires one more pass than using time stamps or sequence numbers.

NOTE The principles are the same as ISO/IEC 9798-2, the symmetric encipherment algorithm is replaced by a cryptographic check function. ISO/IEC 9798-4 provides examples of such functions.

A.2 Digital signature

A file can be signed:

— with a symmetric cryptosystem;

— with an asymmetric cryptosystem.

A.2.1 Using a symmetric cryptosystem involving a Trusted Third Party

This clause describes a scheme to suit the defined outcome of a digital signature (see definition of digital signature) using a symmetric cryptosystem.

The sender and the receiver communicate with a trusted third party. The sender shares a secret key KS with the trusted third party. The receiver shares a different secret key KR with the trusted third party. The way these keys have been established is outside the scope of this International Standard.

a) The sender enciphers the file with KS and sends it to the trusted third party;

b) The trusted third party deciphers the file with KS;

c) The trusted third party adds to the file a statement that the message has been received from the sender, and enciphers it with KR;

d) The trusted third party sends the enciphered file to the receiver;

e) The receiver deciphers the file with KR.

A.2.2 Using an asymmetric cryptosystem

Most digital signature schemes are based upon public key cryptography.

The principle is the following: the sender enciphers the file with his private key, thereby signing the file, the receiver deciphers the file with the sender's public key, thereby verifying the signature.

To prevent the sender from sending twice the same file reusing the same signature, the digital signature includes a time stamp. The date and time are attached to the file and signed along with the rest of the file.

To ensure non repudiation, the secret key is kept in a secure cryptographic device or by a trusted third party.

When combining digital signature with public-key encipherment, the signing operation is performed before the enciphering operation:

1. The sender signs the file with his private key

2. the sender enciphers the signed file with the receiver's public key and sends it to the receiver

3. the receiver deciphers the file with his private key

4. the receiver verifies the signature with the sender's public key.

Two types of digital signature schemes exist:

— giving message recovery: when the verification process reveals the message;

— not giving message recovery: when the verification process needs the message.

ISO/IEC 9796 specifies a scheme giving message recovery for digital signature of messages of limited length and using a public-key cryptosystem. It does not involve the use of a hash-function. This International Standard can be applied to secure file transfer when the file has a limited length, the mechanism is equivalent to encipherment, because the file need not to be transferred, its signature is transmitted alone and the receiver can recover the file from the digital signature.

NOTE ISO/IEC 9796-2 describes a digital signature scheme giving message recovery using a hash-function.

A.2.3 Example of algorithms allowing digital signature

DSA

The Digital Signature Algorithm (DSA) were proposed by the National Institute of Standards and technology (NIST) for use in their Digital Signature Standard (DSS). It specifies a public-key digital signature algorithm. DSA does not allow message recovery.

RSA (Rivest, Shamir, Adleman)

RSA allows message recovery.

A.2.4 Hash-functions

In practical implementations, public-key algorithms are often too inefficient to sign long files. Hash-functions can be used for reducing a file to a short imprint for input to a digital signature mechanism. The principle is the following: the sender reduces the file to a message digest called hash, enciphers the hash with his private key and sends the file and the signed hash to the receiver. The receiver produces a hash of the received file, deciphers the signed hash with the sender's public key. If the two results match, the signature is valid. An archival system can store the hashes of files with a time stamp (the time stamp of the transfer or the time stamp of the submission to the archival system or both). The receiver stores the received digital signatures and in the case of disagreement, the digital signatures can be verified again.

Examples of hash-functions:

— secure Hash Algorithm (SHA-1) is standardised by the NIST as FIPS 180-1.SHA-1 takes as input messages of arbitrary length and produces a 160-bit hash;

— hash-functions using symmetric block algorithms:

  — ISO/IEC 10118, Is entitled "Information technology – Security techniques – Hash-functions – Part 1: General – Part 2: Hash-functions using an n-bit block cipher algorithm".

  — ISO/IEC 10118-2, Specifies hash-functions using an n-bit block cipher algorithm (with the property that plaintext blocks and ciphertext blocks are n bits in length). The n-bit block cipher algorithm is not specified. Two types of hash-functions are specified. The first provides hash-codes smaller than or equal to n, the second provides hash-codes of length less than or equal to 2n.

  — ISO/IEC 10118-3, Standardises SHA-1, RIPEMD-128 and RIPEMD-160.

  — ISO/IEC 10118-2, annex A, presents a way of using the DES with n=64 and the length of the hash = 56.

A.3 Message Authentication Code

A Message Authentication Code may be used to authenticate the origin of a file and to protect the integrity of this file sent by a sender to a receiver. It is generated by the sender of the file and is transmitted along with the file.

A MAC can be applied to the whole file or to part of the file.

ISO/IEC 9797 specifies a data integrity mechanism: the method uses a key and an n-bit block cipher algorithm to calculate an m-bit cryptographic check value. The cryptographic check value is referred to as a Message Authentication Code (MAC).

ISO 9807 gives a list of algorithms approved for the calculation of a MAC, in which the algorithm described in ISO 8731-1, using the DEA in the cipher block chaining mode of operation; it is a specific case of ISO/IEC 9797 when n=64, m=32 and when DEA is used.

A.4 Cipher algorithm

A.4.1 Symmetric ciphers

ISO/IEC 10116 entitled "Modes of operation for an n-bit block cipher algorithm" specifies 4 modes of operation for an n-bit block cipher algorithm (it does not specify the n-bit block cipher algorithm):

— Electronic Code Book mode (ECB);

— Cipher Block Chaining mode (CBC);

— Cipher Feedback mode (CFB);

— Output Feedback mode (OFB),

and comments on the properties of each mode.

NOTE ISO 8372 is the same as ISO/IEC 10116 for n=64.

A.4.2 Asymmetric ciphers

RSA is an example of asymmetric ciphers.

Annex B
(informative)

An example of implementation

B.1 Description of the system

This example concerns a POS terminal composed of:

- software:
  - a secure application manager part of the operating system;
  - several retail banking applications;
  - a parameter file for each application, including a BIN list, a hot list, floor limits,...
- hardware:
  - the terminal itself (CPU, ROM, RAM,...);
  - a secure cryptographic processor;
  - a removable integrated circuit card used to identify and authenticate the merchant by his acquirer.

The different servers to which the device is connected are the following:

- the manufacturer server which downloads the application manager;
- the application supplier server which downloads the application software;
- the parameter server, under the control of the acquirer, which downloads the parameters;
- the acquirer server which receives the financial transactions files.

B.2 Cryptographic FUNCTIONS of the terminal

Notations:

- A = eK(B) : Encipherment of the data B using the symmetric key K;
- B = dK(A) : Decipherment of the cipher text A using the symmetric key K to recover the plain text B.

B.2.1 Software downloading

B.2.1.1 Integrity verification: Asymmetric algorithm

Asymmetric algorithm is used for downloading integrity verification of the software.

This involves a Public key in the terminal. Such a key does not require protection for confidentiality, but only against substitution.

This method provides protection against creation of false signature even in the case of a successful attack on the secure device of the terminal.

B.2.1.2 Mutual authentication: Symmetric algorithm

Symmetric algorithm is used for mutual authentication.

The mutual authentication requires a secret key in the terminal whatever algorithm is used.

To prevent the possibility of using a key obtained after the compromise of a terminal to simulate fake terminals, derivation of the authentication key is required for application manager downloading.

To implement a derived key system, an asymmetric algorithm would require to keep tables of all keys of all terminals, whereas symmetric key algorithm may be implemented with a unique Master key and a derivation algorithm.

B.2.1.3 Key management for Implementation of the application manager

B.2.1.3.1 Mutual Authentication

DEA1 is used as symmetric algorithm for mutual Authentication.

KAM : Mutual Authentication Masterkey used for derivation, generated by the manufacturer, installed in the manufacturer server

KAT_i : Mutual Authentication derived key generated by the manufacturer and installed in the secure cryptographic processor of the terminal prior to delivery.

TID : Terminal Identifier.

KAT_i = eKAM(TID) ; TID = dKAM(KAT_i).

B.2.1.3.2 Integrity

RSA is used as asymmetric algorithm for Integrity.

SKKI : Private key used by the manufacturer to sign the software (need not to be installed in the manufacturer server when signature does not include time stamp or sequence number).

PKKI : Public key generated by the manufacturer and installed in the secure cryptographic processor of the terminal prior to delivery.

B.2.1.4 Key management for Implementation of the Application

B.2.1.4.1 Mutual Authentication

DEA1 is used as symmetric algorithm for Authentication of the terminal.

KAA_n : Mutual Authentication key of the application n, generated by the application provider, installed in the application server. It has to be securely sent to the terminal and protected in its secure cryptographic processor. To do so, the manufacturer should generate an asymmetric key pair: $SKAA_n$, $PKAA_n$. He publishes its public key to all application suppliers in order for them to encipher KAA_n. In return, each application supplier transmits the cryptogram $ePKAA_n(KAA_n)$ to the manufacturer. The Private transport key $SKAA_n$ is installed by the manufacturer in the secure cryptographic processor of the terminal prior to delivery.

The software of the application manager maintains the list of applications to be installed in the terminal and associated cryptograms $ePKAA_n(KAA_n)$ to be used for mutual authentication. During application manager downloading operation, the list of applications and related cryptograms are transferred to the terminal.

B.2.1.4.2 Integrity

RSA is used as asymmetric algorithm to verify the integrity of the application software.

The application provider n generates an asymmetric key pair: SKA_nI, PKA_nI. He publishes its public key to the manufacturer.

SKA_nI : Private key used by the application provider n to sign its application software (need not be installed in the server when signature does not include time stamp or sequence number).

PKA_nI : Associated Public key contained in the downloaded application manager of the terminal.

The software of the terminal maintains the list of applications to be installed in the terminal and associated public keys for integrity verification.

B.2.2 Parameter downloading

The keys are related to the acquirer and may therefore be kept in a *removable* secure cryptographic device used to identify and authenticate the merchant by his acquirer.

B.2.2.1 Mutual Authentication

DEA1 is used as symmetric algorithm for Authentication of the terminal.

KAQM : Master key for mutual Authentication of the acquirer and the terminal, generated by the acquirer, installed in the parameter server.

$KAQR_k$: Mutual Authentication derived key generated by the acquirer and installed in the terminal of the Retailer k e.g. by means of a merchant ICCard.

RID : Retailer Identifier.

$KAQR_k$ = eKAQM(RID).

B.2.2.2 Integrity verification

RSA is used as asymmetric algorithm to verify the integrity of the application parameters.

SKPI : Private key used by the acquirer to sign its application parameter (needs to be installed in the server as signature does include time stamp or sequence number).

PKPI : Associated Public key contained in the merchant ICC to be used by the terminal.

B.2.3 Transaction uploading

B.2.3.1 Mutual Authentication

There is no evidence of secure requirements for authentication prior to transmission of the transaction file when integrity service and non repudiation of delivery service are implemented in accordance with this International Standard.

This example implements these two services.

B.2.3.2 Integrity verification: Symmetric algorithm

A symmetric algorithm is used for integrity verification of the transaction file.

As it is based on a derived keys method, this service provides the authentication service of the terminal by the acquirer server.

DEA1 is used as symmetric algorithm for integrity verification of the transaction file.

KTQM : Master key of the acquirer, generated by the acquirer, installed in the acquirer server.

$KTQR_k$: Integrity verification derived key generated by the acquirer and loaded in the merchant ICCard.

$KTQR_k = eKTQM(RID)$.

The use of derived keys provides protection against creation of false signature even in the case of a successful attack on the secure device of one terminal.

B.2.3.3 Non repudiation of delivery: Asymmetric algorithm

An asymmetric algorithm is used, the Private key SKPNR generated by the acquirer is resident in the acquirer server in order to produce a real time certificate when the transactions file has been successfully received and verified for integrity.

The Public key PKPNR protected against substitution is included in the parameter file sent by the acquirer and protected for integrity.

The terminal uses this certificate to authenticate the acquirer server to which the financial transaction file has been sent and is able to erase the file as soon as this certificate has been successfully verified.