



PCT

159159

1 CHIP CARD COMPRISING MEANS FOR MANAGING A VIRTUAL MEMORY,
2 ASSOCIATED COMMUNICATION METHOD AND PROTOCOL
3

4 The present invention relates to a chip card comprising
5 means for managing a virtual memory.

6 For about twenty years, the chip card has played a vital
7 role in everyday life. The banking field is primarily interested
8 in microcircuit cards; their main advantage is to reduce fraud.
9 Pay TV and radiotelephone companies use them as means for
10 generating keys that are used to encipher and decipher encrypted
11 transmissions. To guarantee security, it was necessary to create
12 a new integrated circuit architecture. Cards of the electronic
13 purse type contain a sum of electronic money; other, so-called
14 frequent-customer cards, offer their holders financial rewards.

15 Nowadays, devices related to microcircuit cards, and more
16 particularly microprocessor cards, are usable in an increasingly
17 large number of applications. At first, the operating system of
18 the cards, i.e., the program located in ROM, could only handle
19 one application. The operating system is written during the
20 production of the microcircuit. By increasing the size of the
21 program memory (ROM) and of the nonvolatile programmable memory
22 (EPROM and EEPROM, and today FeRAM), the operating system can
23 execute more functions. But the number of these functions is
24 still limited by the size of the ROM. Moreover, the addition of
25 another function into the ROM involves producing a new mask; this
26 production is very expensive and is really only profitable when a
27 large quantity of cards is involved.

28 One way to increase the number of these functions without
29 touching the ROM memory consists of writing the executable
30 program, and the data that allows it to run, into the



G000002

1 programmable memory. It is thus possible to add additional
2 functions to an operating system that initially has only a set
3 number of functions. The patent application FR-A-2.748.134
4 describes a means for loading the program into the programmable
5 memory. But the programmable memory is of limited size; once it
6 is filled with a program, it is not possible to add functions.
7 Moreover, the storage of this program takes away from the memory
8 space intended for data in the programmable memory. The preceding
9 method is used to correct certain defects in the program located
10 in ROM or to add some other functions. If a card must run a
11 program of very large size, the method described in this document
12 can prove to be inadequate.

13
14 The object of the present invention is to solve this problem
15 by proposing a method for loading and unloading the programmable
16 memory as a function of the need for the programs and/or the
17 applicative data, for an information processing device
18 constituted by a card. Thus, it becomes possible for the latter
19 to run very diverse applications, such as: electronic purse,
20 banking application, GSM telephony or the health care application
21 currently being tested in France. With the aid of the present
22 invention, the applications just listed are virtually inside the
23 card. The owner of the card has loaded them in advance; thus, the
24 card is configured according to his specific needs.

25 The present invention also makes it possible to solve
26 another problem. A user may need to simultaneously open the same
27 application twice. The running of this application in an
28 information processing device such as a card lasts a certain
29 amount of time. To speed up the process, it is advantageous to be

1 able to start a second running of the application before the end
2 of the first. Thus, the same program is running twice, at the
3 same time.

4 This object is achieved by the fact that the card is
5 equipped with an operating system comprising at least three
6 functions:

- 7 - Loading of applicative information.
- 8 - Unloading of applicative information.
- 9 - Execution of applicative information.

10 To acquire a new application, the card receives applicative
11 information in its programmable memory and checks this data.

12 When a command for running an application is received by a
13 reader cooperating with the card, the operating system of the
14 card analyzes the contents of its memory and determines whether
15 it is necessary to use the network to unload part of its memory,
16 and/or reload previously unloaded applicative information.

17 During the reloading of applicative information, the
18 operating system of the card verifies that the information loaded
19 has been validated by it in the past. This information is then
20 processed.

21 The network may be considered to be an extension of the
22 programmable memory of the card; the latter sends it what it
23 cannot keep in its own memory. It verifies, during the reloading,
24 that the information received from the network is in fact that
25 which it had sent previously. The ROM of the card must have a
26 mechanism for managing the programmable memory that allows it to
27 load and run an unlimited number of applications. From then on,
28 the sizes of the ROMs and programmable memories of the card are
29 no longer a limitation on the number of executable applications,

1 and there is no need to produce a new mask when applications are
2 added.

3 In summary, the invention relates to a chip card comprising
4 information processing means and main information storage means,
5 characterized in that the processing means comprise:

6 - means for detecting, during the operation of the card,
7 that the main storage means contain a quantity of information
8 such that the execution of an operation is not possible;

9 - means for selecting, in the main storage means, a set of
10 information to be unloaded, the unloading of which can release
11 enough space in the main storage means to allow the execution of
12 said operation;

13 - means for unloading the set of information to be unloaded
14 into secondary storage means, in the event that said secondary
15 storage means do not contain this set of information to be
16 unloaded.

17 The invention also relates to the associated method. Lastly,
18 it relates to a protocol for communication between a chip card
19 and a chip card reader, the card comprising information
20 processing means and main information storage means,
21 characterized in that it comprises the steps consisting in that:

22 - the reader transmits to the card an order to execute an
23 operation;

24 - the card searches to see if it has enough space in the ...
25 main storage means to execute this operation;

26 - if so, the card executes this operation, then transmits an
27 execution report to the reader;

28 - if not, the card selects in the main storage means a set
29 of information to be unloaded, the unloading of which can release

1 enough space in the main storage means to allow the execution of
2 this operation, then the card unloads the set of information to
3 be unloaded into secondary storage means by transmitting an
4 unloading order to the reader, in the event that these secondary
5 storage means do not contain this set of information to be
6 unloaded, then it executes said operation, then finally,
7 transmits an execution report to the reader.

8
9 Other details and advantages of the present invention will
10 appear in the following description of several preferred but non-
11 limiting embodiments, in reference to the appended drawings, in
12 which:

13 Fig. 1 represents an information processing network used by
14 the invention;

15 Fig. 2 represents an information processing device, used in
16 Fig. 1 and cooperating with a chip card;

17 Fig. 3 represents a variant of Fig. 2, wherein the
18 information processing device integrates the functionalities of
19 the chip card;

20 Fig. 4 is a variant of Fig. 2, wherein the information
21 processing device is equipped with a device for reading an
22 optical stripe; and

23 Fig. 5 represents a variant of Fig. 3.

24
25 In Fig. 1, a terminal 20 capable of reading a chip card, or
26 a terminal 22 integrating chip card functionalities, cooperates
27 with remote data banks 23 through 25 and is linked to them
28 through a data communication network 26. The data communication
29 network 26 can be a telephone network, the Internet network, or

1 any other data communication network. Each data bank comprises a
2 central data processing unit that manages a memory. According to
3 the invention, and as indicated above, the card 21 or the
4 terminal 22 can, when they detect that the loading of a new
5 application into the latter is not possible because of a lack of
6 memory space, decide to unload another application into one of
7 the data banks 23 through 25. This unloading releases enough
8 memory space to receive the new application. If the card 21 or
9 the terminal 22 subsequently needs the unloaded application, it
10 can send the corresponding data bank a command to reload the
11 application after having, if necessary, released the memory space
12 again by unloading an application.

13 The configuration of the terminal 20 and the card 21 is
14 indicated in Fig. 2. The terminal comprises, in an intrinsically
15 known way, a microprocessor 2, to which are connected a ROM 3 and
16 a RAM 4, means 5 for cooperating, with or without physical
17 contact, with the chip card 21, and a transmission interface 7
18 that allows the terminal to communicate with the data
19 communication network 26 of Fig. 1. The terminal 20 can also be
20 equipped with storage means such as diskettes or disks that may
21 or may not be removable, entry means (such as a keyboard and/or a
22 pointing device of the mouse type) and display means, these
23 various means not being represented in Fig. 2.

24 The terminal can be constituted by any data processing
25 device installed in a private or public site and capable of
26 providing means for managing information or delivering various
27 goods or services, this device being permanently installed or
28 portable. It can also be a dedicated telecommunications device.

29 In addition, the card 21 carries a chip that includes

1 information processing means 9, a nonvolatile memory 10, a
2 volatile working memory RAM 14, and means 13 for cooperating with
3 the terminal 20. This chip is designed to define, in the memory
4 10, a secret area 11 in which information, once recorded, is
5 inaccessible from outside the chip and is accessible only to the
6 processing means 9, and an accessible area 12 that is made
7 accessible from outside the chip by the microprocessor 9 for
8 reading and/or writing information. Each area of the nonvolatile
9 memory 10 can comprise a non-modifiable ROM part and a modifiable
10 part that is an EPROM, EEPROM or is constituted by a RAM of the
11 "flash" type or FRAM (the latter being a ferromagnetic RAM),
12 i.e., having the characteristics of an EEPROM with access times
13 identical to those of a conventional RAM.

14 For the chip, it is possible to use a self-programmable
15 microprocessor with a nonvolatile memory, as described in US
16 patent No. 4.382.279 in the name of the Applicant. As indicated
17 in column 1, lines 13-25 of this patent, the self-programmable
18 feature of the chip corresponds to the possibility for a program
19 fi located in a ROM to change another program fj located in a
20 programmable memory into a program gj. In a variant, the
21 microprocessor of the chip is replaced - or at least supplemented
22 - by logic circuits installed in a semiconductor chip. In effect,
23 such circuits are capable of performing calculations, including
24 authentication and signature calculations, because of the
25 hardwired, rather than microprogrammed, logic. They can be of the
26 ASIC type (for "Application Specific Integrated Circuit".) As an
27 example of an ASIC, it is possible to cite the SIEMENS component
28 marketed under the reference number SLE 4436 and the SGS-THOMSON
29 component marketed under the reference number ST 1335.

1 Advantageously, the chip will be designed in monolithic form.

2 A variant of Fig. 2 is illustrated in Fig. 3, wherein the
3 terminal 22 of Fig. 1 comprises, in addition to the elements of
4 the terminal 20, those of the card 21 disposed in a module 15,
5 the elements common to both Figs. 2 and 3 having the same
6 references. However, the cooperating means 5, 13 of Fig. 2 are
7 replaced by a permanent link between the microprocessor 2 and the
8 microprocessor 9.

9 A variant of Fig. 3 is represented in Fig. 5. In this case,
10 the terminal 50 comprises only one microprocessor 51 or the
11 equivalent, connected to a RAM 52 and to a nonvolatile memory 53.
12 The nonvolatile memory 53 comprises an area 54 made accessible
13 from outside the terminal by the microprocessor 51, and a secret
14 area 55 accessible only to the microprocessor 51. The
15 microprocessor 51 has the self-programmable characteristic of the
16 microprocessor 9 described in connection with Fig. 2. Finally,
17 the terminal 50 has a transmission interface 56 that allows it to
18 communicate with the data communication network 26 of Fig. 1.

19 The following description refers, in a non-limiting way, to
20 the embodiment of Fig. 2, and the terminal 20 is called a
21 "reader" because of its function of reading the card 21.

22 The memories of the card are organized in the following way:
23 a memory of the ROM type, a working memory of the RAM type, and a
24 programmable nonvolatile memory of the EEPROM or FLASH type. As
25 represented in Table 1, the ROM contains a basic operating system
26 area comprising a minimum of sub-programs or routines such as the
27 input/output and memory read/write routines and an area for the
28 operating system of a virtual memory, this virtual memory being
29 constituted by the memory of the data banks 23 through 25. The

1 basic operating system and the operating system of the virtual
2 memory together form what will hereinafter be called the
3 "operating system of the card."

4 The operating system of the virtual memory is preferably
5 capable of handling at least nine commands. At least four
6 commands are sent by the reader to the card:

- 7 - Loading of applications into the card.
- 8 - Running of the previously loaded applications in the card.
- 9 - Erasure of applications in card.
- 10 - Checking for presence of applications in card.

11 Five other commands are sent by the card to the reader:

- 12 - Unloading of applications to the network.
- 13 - Reloading of applications from the network.
- 14 - Suspension of the loading process.
- 15 - Resumption of the loading process.
- 16 - Erasure of applications in the network.

17 In one particular embodiment, the operating system of the virtual
18 memory filters and transmits to the program of the application
19 loaded in programmable memory all the orders received from
20 outside that must be processed by this program.

21
22 In the present text, the term "information" generally
23 designates any executable program or non-executable data. The
24 term "application" designates a particular program for
25 implementing an application of a provider of services or
26 products, and associated application data.

27 Again according to table 1, the programmable memory
28 comprises at least three areas:

- 29 - a first, so-called "system data" area containing a code

1 "C" identifying the card;

2 - a second, so-called "management data" area containing
3 management data of the applications, i.e., a signature key called
4 "SWAP," specific to each card, one or more encryption keys linked
5 as necessary to application providers or to particular
6 applications, and a table called "TAB-APPLI", and

7 - a third, so-called "loading" area used to receive the
8 information of applications, i.e., of the executable program,
9 and/or the data necessary to the running of this program.

10 Initially, the card can be given to its holder with a
11 loading area and a table TAB_APPLI that are empty. At least the
12 key SWAP is located in the secret area 11 of the nonvolatile
13 memory 10 of the card.

Application information loading area
Management data area (SWAP, TAB_APPLI, etc.)
System data area (code C, etc.)
Area for the operating system of the virtual memory (ROM)
Basic operating system area (ROM)

Table 1

The table TAB-APPLI contains the information corresponding to the applications available in the card, whether these applications are physically contained in the card, or whether they are contained in the card virtually, having been unloaded into the network. It has the following structure:

Code of the application	Storage address	Number of bytes	Signature of the information	Load/Unload
I	ADR-I	1	SGN-I	Loaded
J	ADR-J	m	SGN-J	Unloaded
K	ADR-K	n	SGN-K	Loaded

Table 2: TAB_APPLI

The table TAB_APPLI comprises as many rows as there are applications made available by the card, and for each row, five columns. A first column defines an identification code I, J, K of the application. A second column defines a storage address ADR-I,

1 ADR-J, ADR-K starting at which the application is stored in the
2 card. A third column defines a number of bytes representing the
3 quantity of information of the application. A fourth column
4 defines a signature that applies to the all of the bytes of the
5 application, calculated using an algorithm and the key SWAP of
6 the card as a secret key. The algorithm used can be a symmetrical
7 algorithm such as D.E.S. (Data Encryption Standard), or an
8 asymmetrical algorithm such as R.S.A. (for the authors Rivest,
9 Shamir and Adelman); advantageously, however, it suffices to use
10 a simpler function, such as a hash function like MD5 or SHA, or a
11 function such as the "exclusive OR" since, within the framework
12 of the invention, the signature does not leave the card and is
13 therefore protected. Finally, a fifth column defines whether the
14 state of the application in question is "loaded" into the card or
15 "unloaded" into a data bank.

16 First, a card holder or an application provider wants to
17 load into the card a first application having an identification
18 code "K." The execution of a load command can be conditioned by a
19 successful authentication of the cardholder or application
20 provider. The authentication mechanism, which itself is well
21 known, consists, for the holder or application provider, of
22 supplying the card with information that allows it to ensure that
23 it is dialoging with an authorized interlocutor.

24 The load command contains a loading order, the code C of the
25 card, the code K of the application and the number of bytes n of
26 information corresponding to this application, which results in
27 the following command format:
28

1	Loading Order	Card C	Appli K	number n
2				

3 Once the command has been received by the card, the
4 operating system of the card verifies whether the code C sent is
5 actually the same as that recorded in the system data area. If
6 not, the card returns an error message to the network. If so, the
7 information of the application is in fact intended for this card;
8 the operating system of the card then reads the table TAB_APPLI
9 in the management data area to determine whether or not this is
10 an initial load. Initially, TAB_APPLI does not contain
11 information on the application K; if that is not the case, the
12 card responds to the reader with the message "application already
13 loaded"; if it is the case, this is an initial load. The
14 operating system of the card determines whether the n bytes can
15 be accommodated in its memory; if so, it calculates the start
16 address "ADR-K" of a first block of n bytes available in the
17 loading area. If not, it returns the message "insufficient
18 memory". Finally, the card indicates to the reader that it can
19 send the n bytes of the application, using the response
20 "OK_Load". The reader then sends the n bytes of the application.

21 Once the information of the application has been stored in
22 programmable memory, the operating system of the card calculates
23 the signature "SGN-K" of this information. It then re-enters into
24 the table TAB_APPLI the application code K, the storage address
25 ADR-K, the number of bytes n, and the signature SGN-K. Once this
26 operation has been performed, the "Load/Unload" indicator is set
27 to "Loaded." The updating of the table TAB_APPLI being finished,
28 the operating system of the card can then send a report, through
29 the reader, to the card holder or the application provider,

indicating that the loading of the application has been performed correctly. The table TAB_APPLI then has the following structure

Code of the application	Storage address	Number of bytes	Signature of the information	Load/Unload
K	ADR-K	n	SGN-K	Loaded

Table 3: TAB_APPLI

According to a first variant, just after the loading, the operating system of the card can start the executable program contained in the applicative information, i.e., in the information of the application. This makes it possible to initialize the applicative information. For example, in the case of an electronic purse application, the first execution of the program makes it possible to set the balance of the purse written into the memory at 0 francs. According to a second variant, the executable program is started when a first command is sent by the reader to the card, calling the application in question. Simply put, the execution start address of the application is "ADR-K", but it is possible to use an indirect addressing; the designated address is then, in a way that is known in the microprocessor field, the content of the memory marked [ADR-K] that contains the execution address.

The reader sends the card commands, specifying the type of application; for example, this type can be coded into the first of the five bytes of a command, in accordance with the ISO 7816-3 standard; in this standard, this byte is called "CLA." The

operating system of the virtual memory of the card checks the commands that the reader sends it and determines the code of the application corresponding to the command. Then, it reads in the table TAB_APPLI whether the code has been written; if that is the case, the card can run the application K. If it is not the case, the card cannot run the application K; it responds by sending an error message. If the code K is written into TAB_APPLI, the value of the "Load/Unload" indicator is then tested. If it is set on "Loaded," the applicative information is in fact present in the programmable memory of the card. In that case, the operating system of the card passes control to a program of the application located at the address ADR-K or [ADR-K]. We will see below what happens when the programmable memory of the card does not contain the applicative information, because it has already been unloaded.

Let us now assume that the card holder or application provider wants his card to contain the information of a second application, for example marked "J". This is possible by loading the applicative information "J" into the programmable memory of the card. Just as before, the card holder or application provider is authenticated by presenting a secret followed by the following command for loading applicative information:

Loading order	Card C	Appli J	number m
---------------	--------	---------	----------

It is similar to the preceding command related to the loading of the application K; in this case, the number of bytes of the application is m.

The operating system of the card verifies the code C and

searches for the first block of m bytes available in the programmable memory. Let us assume that the programmable memory cannot physically contain the two blocks of applicative information constituted by the application K and the application J simultaneously, but that it can contain the application J if it unloads all or part of the application K. The card informs the reader that it is suspending the process for loading the application J by means of a specific command sent to the reader, and then decides to unload the application K into a data bank, which will be considered as the virtual memory of the card. This unloading will release the memory space for loading the application J.

The unloading then consists of transferring into one of the data banks 23 through 25 of the network, intended specifically for the current card, the applicative information specific to this card. Because of the signature calculation performed during the loading, the card is assured of being able to control the integrity and the authenticity of its own information during a subsequent reloading. Moreover, the fact that the signature calculation has already been performed during the initial load optimizes the run time of the load command. The card sends the card reader the following command:

Order to unload to the network	Card C	Appli K	number n	n bytes of information
-----------------------------------	--------	---------	----------	---------------------------

This command comprises, as the loading command, the code C of the card, the code K of the application to be unloaded, and the number of bytes n of information of the application; it also

1 comprises the contents of these same n bytes of information,
2 transmitted to the reader at the same time as the unloading
3 order. In the event that the unloading of the application occurs
4 when part of it has already been executed, context information,
5 which makes it possible to subsequently resume the running of the
6 application at the point where it was interrupted, is either
7 stored in the programmable memory of the card, or added to the n
8 bytes of information of the application and unloaded into the
9 network at the same time as the latter.

10 It is possible to indicate a destination identifier in the
11 form of a network address. Advantageously, the network has a
12 mapping table that associates each card with the address of the
13 data bank that is specifically intended for it. This makes it
14 possible to prevent the card from having to store this address or
15 this identifier, and to gather in the same data bank all the
16 information unloaded from the same card.

17 The reader receives the command, but recognizes that it is
18 intended for the network: it then sends it to the data bank to
19 which it is addressed. If the network has several data banks, the
20 choice can be made as a function of the code C of the card. The
21 data bank receives the n bytes of applicative information and
22 returns to the card, via the reader, an acknowledgement of
23 successful reception indicating that the storage has actually
24 taken place. The card then modifies the table TAB_APPLI by
25 setting the Load/Unload indicator to "Unloaded." The memory space
26 occupied up to that time by the applicative information of the
27 application K becomes available. The operation for loading the
28 application J can then resume and the card sends the reader a
29 command to resume the loading process; the loading operation

takes place in a way that is identical to that of K. The operating system of the card determines the storage address ADR-J of the m bytes of the application J and indicates to the reader through a message "OK_Load" that it can send the m bytes of applicative information.

The reader sends the m bytes of applicative information that are written starting at the address "ADR-J." Once the information of the application J is stored in programmable memory, the operating system of the card calculates a signature for it by performing a cryptographic calculation using the key SWAP. Finally, the operating system updates the table TAB_APPLI by writing the code J, the values ADR-J, m and SGN-J, and updates the "Load/Unload" indicator, setting it to "Loaded." The operating system can then send the reader a report indicating that the loading has been correctly performed.

The table TAB_APPLI then has the following values:

Code of the application	Storage address	Number of bytes	Signature of the data	Load/Unload
K	ADR-K	n	SGN-K	Unloaded
J	ADR-J	m	SGN-J	Loaded

Table 4: TAB_APPLI

Once the updating of the table TAB_APPLI is finished, the operating system of the card then start the application J in the same way that it started the application K and the card executes the run command sent to it by the reader.

1 If the card holder or application provider connects his card
 2 to a reader and wants to run the application K again, the
 3 operating system of the card analyzes the contents of the table
 4 TAB_APPLI to determine whether this application is accessible
 5 with this card. In the present case, the application K is
 6 recorded in TAB_APPLI, but it has been unloaded into the network.
 7 Another application is in memory, i.e., J, and it occupies m
 8 bytes. The operating system then tests to see whether the
 9 application K, which occupies n bytes in memory can be loaded
 10 into the remaining available memory. As assumed above, the
 11 response to this test is negative. The operating system then
 12 decides to unload the current application J in order to be able
 13 to reload the application K.

14 The command, sent by the card, for unloading J into the
 15 network is:

Order to unload into the network	Card C	Appli J	number m	m bytes of information
-------------------------------------	--------	---------	----------	---------------------------

20 Once the operation has been performed, the indicator of the
 21 loading of the application J into TAB_APPLI is set to the
 22 "Unloaded" position. The memory space now being available, the
 23 operating system sends the reader a command to reload the
 24 application K from the network. This command has the following ...
 25 format:

Order to reload from the network	Card C	Appli K	number n
-------------------------------------	--------	---------	----------

1 The reader receives the command and sends it to the data
2 bank associated with the card C. The data bank that has the
3 information of the card C receives the command and searches in
4 the file of this card for the n bytes of applicative information
5 relative to the application K. The data bank generates the
6 following message, which is the response to the card's last
7 command. This response is transmitted to the card via the reader:

9 Card C	10 Appli K	11 number n	12 n bytes of data
----------------	------------------	-------------------	--------------------------

13 The operating system of the card can verify whether the
14 codes C, K and the value n received are in fact identical to
15 those of the unloading command sent previously. If they match,
16 the command continues with the reception of the n bytes of data,
17 which are written starting at the address ADR-K into the loading
18 area, this address being read for this purpose by the operating
19 system in the table TAB_APPLI or retrieved from reloaded context
20 information. At the same time, the operating system calculates
21 the signature of the n bytes written through a cryptographic
22 calculation using the value of the key SWAP. The recalculated
23 signature is then compared to the value written into the table
24 TAB_APPLI. If the data received from the network is not identical
25 to that previously unloaded, the two signature values will not be
26 equal. There is therefore a doubt as to the authenticity or
27 integrity of the information received. The loaded information
28 cannot be processed. The card sends the reader an error message
29 indicating an erroneous reception of information during the last
loading operation, and the impossibility of running the
application K; the operating system does not set the load

indicator to the "loaded" position; if necessary, it may erase the contents of the application K.

If, on the other hand, the two signature values are equal, the information received does correspond to that of the application K previously loaded into the card. Once these checks are done, the operating system of the card updates the table TAB_APPLI by setting the loading indicator of the application K to the "Loaded" position.

The table TAB_APPLI then has the following values:

Code of the application	Storage address	number of bytes	Signature of the data	Load/Unload
K	ADR_K	n	SGN-K	Loaded
J	ADR_J	m	SGN-K	Unloaded

Table 5: TAB_APPLI

Once the update of the table TAB_APPLI is finished, the operating system starts the application K as before, and the card can execute the last applicative type command sent by the reader.

As described above, when the card receives a command to load an application not currently stored, the operating system of the card tests the available space in memory. If there is enough space, the loading can be done without unloading the application currently in memory. There are then two applications in the card. The table TAB_APPLI therefore has the following configuration:

Code of the application	Storage address	number of bytes	Signature of the data	Load/Unload
K	ADR-K	n	SGN-K	Loaded
I	ADR-I	1	SGN-I	Loaded
J	ADR-J	m	SGN-J	Unloaded

Table 6: TAB_APPLI

In this example, two applications I and K coexist in the card; they are directly executable. A third application J is accessible by means of this card, but it must be reloaded from the network. The nonvolatile memories of the card contain the following information:

ADR-K	ADR-I	
Program of the application K	Program of the application I	
		Available
Data of the application K	Data of the application I	
Management data (key SWAP, TAB_APPLI, etc.)		
System data (code C, etc.)		
Operating system of the virtual memory (ROM)		
Basic operating system (ROM)		

Table 7

This table corresponds to the above-mentioned Table 1, in which the loading area breaks down as follows: it is clear that the area for loading the applicative information comprises three sub-areas: an area receiving the information of the application K, an area receiving the information of the application I, and a residual available area whose size is less than m.

In light of this example, the characteristics of the invention can be better understood. The card is equipped with a minimum operating system that makes it possible to manage the memory space, to load or unload applications, to sign the applicative information to be unloaded into the network, to verify the applicative information unloaded into and received

1 from the network by comparing the signatures, and to start
2 applications loaded into the memory. The signature makes it
3 possible to verify that the applicative information stored in the
4 data bank has been previously loaded into this card. The reader
5 is equipped with a program that recognizes the commands for
6 unloading and reloading the card and means for transmitting these
7 commands to the network. Finally, the network is equipped with
8 data banks, the memory of which banks can be considered to be an
9 extension of the programmable memory of the card.

10 As seen in the preamble, the writing of routines into the
11 programmable memory in order to modify the running of the program
12 in ROM can only be done by persons who know this program. The
13 jumps to these routines and their returns into the program in ROM
14 require precise knowledge of the addresses, the input and output
15 parameters of these routines, the utilization ratio of the
16 working memory, etc. The present invention solves this problem by
17 avoiding using these routines, and consequently revealing the
18 specifications of these routines while allowing the running of
19 numerous applications. The applicative programs are executed
20 using the program in ROM as little as possible. The designer of
21 this program can indicate the entry points to certain so-called
22 elementary routines: receiving bytes, sending bytes, writing n
23 bytes into programmable memory, cryptographic calculation, etc.

24 A first improvement of the invention consists of encrypting
25 the applicative information in order to protect it during its
26 various transfers between the information processing device
27 intended to receive applications (such as the card 21 or the
28 terminal 22 of Fig. 1) and the network, and during their storage
29 outside the card 21 or the terminal 22.

1 A first application encryption relates to the initial load
2 of the application by an application provider and uses a basic
3 secret key held by the information processing device and the
4 application provider located in the network; in the event that
5 the information processing device is a card, its reader does not
6 know the basic key. Advantageously, each application is encrypted
7 with a specific diversified key obtained from the basic key and
8 from a diversifier constituted by a specific parameter of the
9 application, for example its code K or its storage address ADR-K
10 in the programmable memory. This diversifier can be stored in the
11 table TAB_APPLI so that the operating system can easily retrieve
12 it during load/unload commands.

13 During the initial load of the application by the
14 application provider into the information processing device 21 or
15 22, this provider calculates the diversified key associated with
16 this application and encrypts the application using the latter
17 before sending it into the network; upon reception, the
18 information processing device calculates the diversified key
19 associated with this application and decrypts it with this key
20 before storing it in the loading area of the programmable memory.

21 A second encryption of the application relates to the loads
22 and unloads performed by the information processing device 21,
23 22. During an unloading of the application by the information
24 processing device 21, 22 to a data bank, the application is again
25 encrypted by this device. The encryption key used should not be
26 shared by the information processing device with any interlocutor
27 other than the application provider, no matter which key
28 generated by the information processing device is suitable, since
29 it is this device, and this device alone, that will perform the

subsequent decryption.

Advantageously, the card can use the method described by the document US-A-4,907,270, the object of which is to provide a process for ensuring the authenticity and integrity of an encrypted message.

The encryption described above makes it possible to prevent applicative information from being discovered by a hacker, and prevents the fraudulent copying of the applicative programs.

In addition to the commands described above, it is possible to provide two additional commands: a command for erasing applications and a command for checking for the presence of applications on the card.

The command for erasing applications consists, for the card holder or the application provider, of sending the card a command intended to delete the applications that are no longer used; its format is the following:

Application erasure order	Card C	Appli K	number n
------------------------------	--------	---------	----------

It comprises an application erasure order, the code C of the card in question, the code K of the application, and possibly the number n of bytes of information of the application. If the application in question is loaded into the card, the operating system of the card frees up the memory space reserved up to now for the application K. If, on the other hand, the application K has been unloaded into a data bank, the card sends the latter an erasure order that has the same format as the one above. Finally, once the erasure order has been carried out, the operating system

erases the row of the table TAB_APPLI related to this application.

The command for checking for the presence of applications on the card can take two different forms. The first form of the command allows the card holder or application provider to ask the card if it has a particular application; its format is the following:

Order to check for the presence of applications	Card C	Appli K	number n
---	--------	---------	----------

It comprises an order to check for the presence of applications, the code C of the card in question, the code K of the application, and possibly the number n of bytes of information of the application.

The second form of the command allows the card holder or application provider to ask the card for all of the rows of its table TAB_APPLI, except of course for the signatures and possibly the number n of bytes and the load indicator. The format of the command is the following:

Order to check for the presence of applications	Card C
---	--------

A second improvement of the invention consists of not starting the unloading of an application to the network until it is necessary. If, at the moment the memory must be released, the loaded application has not been modified and the network already has the same applicative information of this application, it is

not necessary to unload this information. The object of the second improvement is to avoid storing the same applicative information values several times in the network.

To implement this improvement, it is necessary to modify the table TAB_APPLI; here is the new structure:

Code of the application	Storage address	number of bytes	Signature of the information	Load/Unload	Modification
K	ADR-K	n	SGN-K	Loaded/Unloaded	YES/NO

Table 8: TAB_APPLI

A sixth column has been added to the table, which contains an indicator named "Modification" that can have two values: Yes or No. During the initial load of an application, the indicator is set to "Yes"; this value indicates that it is necessary to unload the applicative information to the network in order to release the corresponding memory space. On the other hand, after a command to reload from the network, the indicator is set to "No"; this value indicates that the applicative information stored in programmable memory of the information processing device (card 21 or terminal 22 of Fig. 1) are identical to those stored in the data bank of the network. As long as the indicator remains at "No," the operating system of the information processing device does not execute a command for unloading the application; it simply sets the loading indicator to the

1 "Unloaded" position so that another application can take its
2 place in memory. The indicator is set to "Yes" when the
3 applicative information is modified; consequently, the signature
4 value is no longer exact; it must be recalculated during the
5 unloading.

6 This modification can occur in at least two cases. The first
7 case is an update of the applicative program, either in order to
8 add to its performance by adding supplementary functions or to
9 correct a problem. The second case occurs frequently when, in the
10 programmable memory of the information processing device 21 or
11 22, data are mixed with the application program. For example, an
12 electronic purse application simultaneously contains both the
13 software for managing debits and credits and data such as the
14 balance. With each utilization, this value generally changes, and
15 therefore the indicator "Modification" is nearly always in the
16 "Yes " position.

17 The latter example leads to a third improvement of the
18 present invention. It is clear that in the applicative
19 information, the executable program and the applicative data
20 values capable of evolving often exist simultaneously. The means
21 described in the third improvement described below actually make
22 it possible to separate the two types of information. The
23 information processing device then chooses to unload to the
24 network only the information it has actually modified.

25 In order to achieve this third improvement, it is necessary
26 to improve the organization of the nonvolatile memories, which
27 can be diagrammed in the following way:

1	Program of the	
2	application	
3	(programmable memory)	
4	Data of the application	Changing data,
5		sequence 1
6		Changing data,
7		sequence 2
8	Management data (key SWAP, TAB_APPLI, etc.)	
9	in programmable memory	
10	System type data in programmable memory	
11	(code C, etc.)	
12	Operating system of the virtual memory	
13	(ROM)	
14	Basic operating system	
15	(ROM)	

Table 9

Table 9 differs from Table 1 above by the structure of its loading area of the programmable memory, which appears as follows:

- a block related to the application as such and comprising two sub-blocks of information:
 - a block related to the executable program of the application, marked "program of the application";
 - a block related to the changing (non-executable) data of the application, marked "data of the application";
- a certain number of changing (non-executable) data blocks

corresponding to particular executions of the executable program; these executions will hereinafter be called "sequences." By definition, the data of a sequence is temporary, i.e., it is used only during this sequence, and not during previous or subsequent sequences. This is what distinguishes them from the above-mentioned "data of the application," which are used during all the sequences. In Table 9, two data blocks of sequences are represented, called "changing data, sequence 1" and "changing data, sequence 2." The role of these different blocks of information will be explained in the example below.

In order to achieve this third improvement, the table TAB_APPLI is modified; it has the following structure:

Applicati oncode/ Sequence number	Information related to the executable program and to the data of the application				Information related to the changing data of the sequences marked "i"			
	stora ge addre ss	number of bytes	signat ure	Loaded / Unload ed	Stora ge addre ss	number of bytes	signatur e	Loaded / Unload ed
P/1	ADR- Cod- P	p-cod	SGN- cod- P	Loaded	ADR- Dat- P/1	p- dat	SGN-dat- P/1	Loaded
P/2	ADR- Cod- P	p-cod	SGN- cod- P	Loaded	ADR- Dat- P/2	p- dat	SGN-dat- P/2	Loaded ...
J/1	ADR- Cod- J	j-cod	SGN- cod- J	Loaded	ADR- Dat- J/1	j- dat	SGN-dat- J/1	Loaded

J/2	ADR- Cod- J	j-cod	SGN- cod- J	Loaded	ADR- Dat- J/2	j- dat	SGN-dat- J/2	Unload ed
-----	-------------------	-------	-------------------	--------	---------------------	-----------	-----------------	--------------

Table 10: TAB_APPLI

Compared to the above-mentioned table TAB_APPLI 2, this table has the following differences. The first column specifies, in addition to the application code, the number "i" of the sequence in question. The information is processed in two groups: those related to the executable program and the data of the application, and those related to the changing data of the sequences. For each group of information, there are the following four columns of the table TAB_APPLI 2: storage address, number of bytes, signature, load indicator. Each row of the table corresponds to a data sequence P/1 or P/2, both related to an application P, or a sequence J/1 or J/2, both related to another application J. In different cells of the table, the code of the application is mentioned in order to indicate that the value in question is related to a given application, for example:

- ADR-Cod-P: storage address related to the application P
- j-cod: number of bytes related to the application J.

Furthermore, the symbol "Cod" indicates that the value in question is related to information of the "application" type (program or data of the first group), while "Dat" indicates that the value in question is related to information of the "sequence" type (data of the second group); for example:

- SGN-cod-P: signature of information (program or data) related

1 to the application P

2 • SGN-dat-J/2: signature of data related to sequence No. 2 of
3 the application J.

4 An example will better describe the problem posed and the
5 way to solve it using the present invention.

6 The information processing device (card 21 in this case) has
7 just received an initial load command for the application P: a
8 payment application of the electronic purse type. The applicative
9 information stored in programmable memory is the executable
10 program and the data related to the application; there is not yet
11 any changing data corresponding to a sequence. This information
12 comprises n-Cod bytes stored starting at an address ADR-Cod-P.
13 The load indicator is set to "Loaded." In addition to the
14 information related to the executable program and to the data of
15 the application, the information transmitted during the command
16 contains a number of bytes of changing data "p-dat" related to a
17 sequence i. The table TAB_APPLI therefore has the following
18 values:

Sequence number	Information related to the executable program and to the data of the application				Information related to the changing data of the sequences marked "i"			
	stora ge addre ss	number of bytes	signat ure	Loaded / Unload ed	Stora ge addre ss	number of bytes	signatur e	Loaded / Unload ed
P/1	ADR- Cod- P	p-cod	SGN- cod- P	Loaded	0	p-dat	0	0

Table 11: TAB_APPLI

The transactions are validated by an electronic circuit called a security module. This module can be located either in the card reading terminal 20 of Fig. 1 or, if maximum security is desired, in an approved banking center, which can be located very far from the terminal 20. A transaction of the electronic purse type takes place in several stages that require communications between the card, the terminal and the security module. The purchase can be made at a store equipped with a terminal with a module, but it can also be made at the home of the card holder, whose terminal is not equipped with a module.

The card is used to make a purchase by means of a transaction initialization order. The operating system of the card recognizes an order of the applicative type; it then queries its table TAB_APPLI. The query of the table tells it that the application corresponding to the order is in fact loaded and that no sequence has been allocated. The operating system then initializes a sequence by assigning it a number, "1" for example. It allocates this sequence a memory space of "n-dat" bytes,

starting at the address ADR-Dat-P/1. The load indicator corresponding to this sequence is set to "Loaded." The table TAB_APPLI then has the following values:

Sequence number	Information related to the executable program and to the data of the application				Information related to the changing data of the sequences marked "i"			
	storage address	number of bytes	signature	Loaded / Unloaded	Storage address	number of bytes	signature	Loaded / Unloaded
P/1	ADR-Cod-P	n-cod	SGN-cod-P	Loaded	ADR-Dat-P/1	n-dat	0	Loaded

Table TAB_APPLI 12

Next, the operating system of the card starts the applicative program by executing a jump to the address ADR-Cod-P; it specifies the address ADR-Dat-P/1 of the temporary data to be used, which allows the application to know the place where the data of the sequence is stored. This data includes, among other things, the amount of the transaction, the object of the transaction, the selling institution and the date of the transaction. On the other hand, data such as the balance of the electronic purse is not temporary sequence data, since its lifetime exceeds that of a sequence; being of the applicative type, this data is stored with the program of the application.

The purchase of a first product is in progress; the card sends the reader 20 a message in order to obtain a validation of the transaction from a payment center accessible through the

network. This communication can last for some time. In fact, the communications can be disturbed and the data sent can take a long time to be analyzed by the approved banking center. This causes an increase in the global duration of the transaction. During this time, the user decides to make a second purchase. The present invention makes it possible to avoid waiting for the end of the first transaction before beginning the second one.

To make this second purchase, the card is used a second time by a new transaction initialization order. Just as before, the operating system of the card verifies that the executable program of the electronic purse application is loaded into programmable memory. This verification is done by querying its table TAB_APPLI; the operating system thereby recognizes the presence of the program and of a sequence (1) that is in progress. For this reason, it assigns this second execution a new sequence number (2) and initializes the table TAB_APPLI by adding a new row to it. Then, it verifies whether there is enough space to allocate n-dat bytes in the programmable memory for the information of the non-executable data type. If there is enough space, a new address ADR-Dat-P/2 is determined and the second transaction can be started. The table TAB-APPLI has the following values.

Applicati oncode/ Sequence number	Information related to the executable program and to the data of the application				Information related to the changing data of the sequences marked "i"			
	stora ge addre ss	number of bytes	signat ure	Loaded / Unload ed	Stora ge addre ss	number of bytes	signatur e	Loaded / Unload ed

P/1	ADR- Cod-P	n- cod	SGN- cod-P	Loaded	ADR- Dat- P/1	n- dat	0	Loaded
P/2	ADR- Cod-P	n- cod	SGN- cod-P	Loaded	ADR- Dat- P/2	n- dat	0	Loaded

Table TAB_APPLI 13

The two transactions will then be performed in parallel in the card, without using the network. The reader must indicate, in the applicative commands sent to the card, the transactions to which they refer.

If there is not enough space, the operating system of the card decides to unload only the changing data corresponding to the first transaction (sequence number 1). It then calculates the signature of this data of the first sequence "SGN-dat-P/1," and writes it into the table TAB_APPLI. The new non-executable data can thus be in the same place as the unloaded data, i.e., at an address common to both sequences and marked ADR-Dat-P. Next, the card sends the reader the following command:

Order to unload to the network	Card C	Appli P - Data - sequence number 1	number n_dat	"n_dat" bytes of data
-----------------------------------	--------	---------------------------------------	-----------------	-----------------------------

This command has a structure identical to the one mentioned above, with the following difference: the third cell contains a

parameter specifying not only the code P of the application, but also the fact that the data is of the sequence type (through the term "Data"), and the number 1 of the sequence in question.

As a result of this command, the table TAB_APPLI has the following values:

Applicati oncode/ Sequence number	Information related to the executable program and to the data of the application				Information related to the changing data of the sequences marked "i"			
	stora ge addre ss	number of bytes	signat ure	Loaded / Unload ed	Stora ge addre ss	number of bytes	signatur e	Loaded / Unload ed
P/1	ADR- Cod- P	n-cod	SGN- cod- P	Loaded	ADR- Dat- P	n-dat	SGN-dat- P1	Unload ed
P/2	ADR- Cod- P	n-cod	SGN- cod- P	Loaded	ADR- Dat- P	n-dat	0	Loaded

Table TAB_APPLI 14

As a result of this operation, the second transaction having the sequence number 2 can continue. This new transaction also requires a validation from the payment center; a request is therefore sent to the security module. Let us assume that at this moment the card receives a validation message for the first transaction. The operating system of the card recognizes, by means of the sequence number, that this message relates to a

transaction other than the current one and, by reading the table TAB_APPLI, it recognizes the first transaction. In order to process it, it must then load the non-executable data of the first transaction.

Given that there is not enough memory space for both blocks of data, the operating system of the card must therefore unload the data of the second transaction. It therefore calculates the signature of this data "SGN-dat-P/2," and writes it into the table TAB_APPLI. Then, the card sends the reader the following command:

Order to unload to the network	card C	appli P - data - sequence number 2	number n_dat	"n_dat" bytes of data
-----------------------------------	--------	---------------------------------------	-----------------	-----------------------------

The table TAB_APPLI then has the following values:

Applicati oncode/ Sequence number	Information related to the executable program and to the data of the application				Information related to the changing data of the sequences marked "i"			
	stora ge addre ss	number of bytes	signat ure	Loaded / Unload ed	Stora ge addre ss	number of bytes	signatur e	Loaded / Unload ed
P/1	ADR- Cod-P	n- cod	SGN- cod-P	Loaded	ADR- Dat- P	n- dat	SGN-dat- P1	Unload ed

P/2	ADR- Cod-P	n- cod	SGN- cod-P	Loaded	ADR- Dat- P	n- dat	SGN-dat- P/2	Unload ed
-----	---------------	-----------	---------------	--------	-------------------	-----------	-----------------	--------------

Table TAB_APPLI 15

The operating system of the card then sends the reader the following command:

Command to reload from the network	Card C	Appli P - Data - sequence number 1	number n-dat
---------------------------------------	--------	---------------------------------------	-----------------

This command differs from the reload command already written in that the third cell contains a parameter specifying not only the code P of the application, but also the fact that the data is of the sequence type (through the term "Data"), and the number 1 of the sequence in question.

The reader receives the command and sends it the data bank specifically assigned to the card C. The data bank searches in the file of this card for the n-dat bytes of non-executable data related to the application P, sequence number 1. The data bank generates the following message, which is the response to the card's last command; this response is transmitted to the card via the reader:

Card C	Appli P - Data - sequence number 1	n-dat	n-dat bytes of data
--------	---------------------------------------	-------	------------------------

1
2 This command differs from the response to a reload command
3 already written in that the second cell contains a parameter
4 specifying not only the code P of the application, but also the
5 fact that the data is of the sequence type (through the term
6 "Data"), and the number 1 of the sequence in question.

7 The operating system of the card can perform a preliminary
8 operation in which it verifies whether the codes C, P, the
9 sequence number and the value n-dat received are in fact
10 identical to those of the command sent previously. If they match,
11 the n-dat bytes received are stored starting at the address ADR-
12 dat-P read in the table TAB_APPLI. Once the last byte has been
13 written, the operating system recalculates the signature of the
14 data by means of a cryptographic calculation using the value of
15 the key SWAP. The recalculated signature is then compared to the
16 value "SGN-dat-P/1" written in the table TAB_APPLI. If the two
17 signature values are not equal, the data received from the
18 network is considered not to be identical to that unloaded
19 previously. There is therefore a doubt as to the authenticity or
20 integrity of the data received. The card returns to the reader an
21 error message indicating the reception of erroneous data during
22 the last load operation, and the impossibility of continuing the
23 transaction.

24 If the two values are equal, the data received are ...
25 considered to be identical to those previously unloaded by the
26 card: the first transaction can therefore continue. The operating
27 system of the card then updates the table TAB_APPLI by setting
28 the indicator for the data of the application P/1 to "Loaded."
29

Applicati oncode/ Sequence number	Information related to the executable program and to the data of the application				Information related to the changing data of the sequences marked "i"			
	stora ge addre ss	number of bytes	signat ure	Loaded / Unload ed	Stora ge addre ss	number of bytes	signatur e	Loaded / Unload ed
P/1	ADR- Cod- P	n- cod	SGN- cod- P	Loaded	ADR- Dat- P	n- dat	SGN-dat- P1	Loaded
P/2	ADR- Cod- P	n-cod	SGN- cod- P	Loaded	ADR- Dat- P	n- dat	SGN-dat- P/2	Unload ed

Table TAB_APPLI 16

The updating of the table TAB_APPLI being finished, the operating system starts the application P, which will continue the first transaction.

The first transaction being finished, the execution of the program of the application ends with a return to the operating system that manages the virtual memory. The operating system recognizes the end of the sequence "1" and decides to release the memory space corresponding to the data of this sequence. To do this, it erases the information "storage address", "signature" and the load/unload indicator by setting them to the value zero.

The table TAB_APPLI then has the following values:

Applicati oncode/ Sequence number	Information related to the executable program and to the data of the application				Information related to the changing data of the sequences marked "i"			
	stora ge addre ss	number of bytes	signat ure	Loaded / Unload ed	Stora ge addre ss	number of bytes	signatur e	Loaded / Unload ed
P/1	ADR- Cod- P	n-cod	SGN- cod- P	Loaded	0	n- dat	0	0
P/2	ADR- Cod- P	n-cod	SGN- cod- P	Loaded	ADR- Dat- P	n- dat	SGN-dat- P/2	Unload ed

Table TAB_APPLI 17

When the card receives the validation of the second transaction, the operating system of the card recognizes, by means of the sequence number, that the message relates to another transaction that is not loaded. The first transaction being finished, the corresponding non-executable data are no longer useful. There is no therefore no need to unload them. It suffices to load the non-executable data corresponding to the second transaction. The operating system sends the reader the following command:

Command to reload from the network	Card C	Appli P - Data - sequence number 2	number n-dat
---------------------------------------	--------	---------------------------------------	-----------------

Just as for the loading of the sequence 1, the reader

receives the command and sends it to the data bank. The data bank searches in the file of this card for the n-dat bytes of non-executable data related to the application P, sequence number 2. The data bank generates the following message, which is transmitted to the card via the reader:

Card C	Appli P - Data - sequence number 2	number n-dat	n-dat bytes of data
--------	---------------------------------------	-----------------	------------------------

The operating system of the card can perform a preliminary operation in which it verifies the codes C, P, the sequence number, and the value n-dat received. If the verification is positive, the bytes are written. Then, the operating system calculates and verifies the signature of the data. If the two values are equal, the data received are considered to be identical to those previously unloaded by the card; the second transaction can therefore continue. The operating system updates the table TAB_APPLI by setting the load indicator of the application P/2 to "Loaded."

Applicati oncode/ Sequence number	Information related to the executable program and to the data of the application				Information related to the changing data of the sequences marked "i"			
	stora ge addre ss	number of bytes	signat ure	Loade d/ Unloa ded	Stora ge addre ss	number of bytes	signatur e	Loaded ... / Unload ed
P/1	ADR- Cod- P	n-cod	SGN- cod- P	Loade d	0	n-dat	0	0

P/2	ADR- Cod- P	n-cod	SGN- cod- P	Loade d	ADR- Dat- P	n-dat	SGN-dat- P/2	Loaded
-----	-------------------	-------	-------------------	------------	-------------------	-------	-----------------	--------

Table TAB_APPLI 18

The updating of the table TAB_APPLI being finished, the operating system starts the application P, which will continue the second transaction.

The second transaction being finished, the program of the application ends with a return instruction to the operating system managing the virtual memory. The operating system deduces from this that the sequence "2" is finished; the memory space can then be released. To do this, the locations in the table TAB_APPLI of: "storage address", "signature" and the load/unload indicator are set to zero. The table takes on the following values:

Applicati oncode/ Sequence number	Information related to the executable program and to the data of the application				Information related to the changing data of the sequences marked "i"			
	stora ge addre ss	number of bytes	signat ure	Loaded / Unload ed	Stora ge addre ss	number of bytes	signatur e	Loaded / Unload ed
P/1	ADR- Cod- P	n-cod	SGN- cod- P	Loaded	0	n-dat	0	0
P/2	ADR- Cod- P	n-cod	SGN- cod- P	Loaded	0	n-dat	0	0

Table TAB_APPLI 19

At this stage, the operating system of the card can completely erase one row of the table TAB_APPLI. The management of the rows of the table TAB_APPLI therefore takes place dynamically as a function of need.

Another, static method for managing the table is to decide once and for all the maximum number of executable sequences for an application; let "s" be this number. "s" is then transmitted during the initial load command for the application; the operating system reserves in the table TAB_APPLI the space corresponding to these "s" sequences. Let s have, for example, the value 2.

The command for loading the application K has the following values:

Loading Order	Card C	Appli K	number n		s=2
			n-cod	n-dat	

This command differs from the one described previously in that it includes a fifth cell defining the value of the parameter s. It will be noted that in this case the command specifies the number n-cod of bytes related to the application and sent by the command, and the number n-dat of bytes related to each future sequence and reserved for this use. In a variant, the number n-dat of bytes may not be transmitted at this stage, but provided to the operating system of the card later by the application that is loaded into the card.

As a result of this command, the operating system updates the table TAB_APPLI with the following values:

Applicati oncode/ Sequence number	Information related to the executable program and to the data of the application				Information related to the changing data of the sequences marked "i"			
	stora ge addre ss	number of bytes	signat ure	Loaded / Unload ed	Stora ge addre ss	number of bytes	signatur e	Loaded / Unload ed
K/1	ADR- Cod- K	n-cod	SGN- cod- K	Loaded	0	n-dat	0	0
K/2	ADR- Cod- K	n-cod	SGN- cod- K	Loaded	0	n-dat	0	0

Table TAB_APPLI 20

The application K can now be run: two sequences are possible.

The card can virtually contain several complete applications, each equipped with several sequences. For example, here is one particular configuration of the table TAB_APPLI:

Applicati oncode/ Sequence number	Information related to the executable program and to the data of the application				Information related to the changing data of the sequences marked "i"			
	stora ge addre ss	number of bytes	signat ure	Loaded / Unload ed	Stora ge addre ss	number of bytes	signatur e	Loaded / Unload ed
K/1	ADR- Cod- K	k-cod	SGN- cod- K	Unload ed	0	k-dat	0	0
K/2	ADR- Cod- K	k-cod	SGN- cod- K	Unload ed	ADR- Dat- K/2	k-dat	SGN-dat- K/2	Unload ed
K/1	ADR- Cod- K	k-cod	SGN- cod- K	Unload ed	ADR- Dat- K/3	k/dat	SGN-dat- K/3	Loaded
J/1	ADR- Cod- J	j-cod	SGN- cod- J	Loaded	ADR- Dat- J/1	j-dat	SGN-dat- J/1	Loaded
J/2	ADR- Cod- J	j-cod	SGN- cod- J	Loaded	ADR- Dat- J/2	j-dat	SGN-dat- J/2	Unload ed

Table TAB_APPLI 21

Corresponding to this example, the card virtually has two applications marked K and J. The executable program of the

1 application K is not in the loading area; three sequences of this
2 application, marked 1, 2 and 3, can be executed at the same time.
3 The first sequence is finished, the other two are currently being
4 executed. The sequence 2 is unloaded; it will therefore be
5 necessary to reload it in order to finish it. Also, to finish the
6 sequences 2 and 3, it will be necessary to reload the executable
7 program and the data of the application K.

8 The executable program of the application J is in the
9 loading area; this application can simultaneously execute two
10 sequences, marked 1 and 2, which are currently being executed.
11 The sequence 2 is unloaded; it will be necessary to reload it in
12 order to finish it.

13 This example illustrated the need to properly manage the
14 available memory space. The loading area should be occupied as
15 much as possible, thus avoiding the Unload and Reload commands as
16 much as possible.

17 Quite clearly, the improvement that consists of encrypting
18 the data, in addition to signing them, when unloading, and of
19 decrypting them when loading/reloading, can be applied to this
20 third improvement.

21 An improvement of the procedure for initially loading an
22 application into a card consists of entering into the card a
23 signature of the applicative information calculated from a key of
24 the application provider. This signature makes it possible to
25 ensure the integrity of the applicative information and to
26 authenticate the source of this applicative data.

27 The initial load according to the improvement consists of
28 presenting the card to the application provider. It is
29 recommended that this operation be done on the application

1 provider's premises. The application provider enters into the
2 card his provider key, the signature of the applicative
3 information, and the application code, K for example. The card
4 holder executes a request to initially load the application K.
5 This request, which has been described above, can be made at his
6 home. A method for performing the initial load of an application
7 in a protected way is described in the document FR-A-2.748.134.

8 According to a variant of embodiment of the invention, the
9 applications stored in a card are not unloaded into a remote data
10 bank through a network; it is the reader 20 of Fig. 2 that
11 receives and stores these applications; it therefore has for this
12 purpose a nonvolatile programmable memory in which the
13 applications are stored. The load and unload commands are
14 unchanged. This variant is advantageous when the card is always
15 inserted into the same reader, for example a reader located at
16 the card holder's home.

17 Another variant of embodiment of the invention uses the card
18 reader 40 and the chip card 41 of Fig. 4, wherein the elements in
19 common with Fig. 2 have the same references. The card 41 is
20 distinguished from that 21 of Fig. 2 in that it has an optical
21 stripe 42, for example a stripe to be written and read by laser
22 beam. The card reader 40 is distinguished from that 20 in that it
23 comprises an optical stripe reader 43 capable of reading and
24 writing information on the optical stripe 42, connected to the ...
25 microprocessor 2 and to the memories 3, 4.

26 According to the invention, the optical stripe 42 is used as
27 the data bank, in place of the remote ones 23 through 25 of Fig.
28 1. In practice, during the unloading of an application from the
29 card 41, the card transmits the unload command to the card reader

1 40. The stripe reader 43 receives the information of the
2 application and writes it onto the optical stripe 42. During a
3 reload command, the card reader activates the stripe reader 43 so
4 that it reads the information of the application on the optical
5 stripe 42; the card reader then transmits this information to the
6 microprocessor 9 of the card so that the latter stores them in
7 the loading area. The load and unload commands are nevertheless
8 unchanged.

9 In a variant, the optical stripe is replaced by another mass
10 storage medium, for example a magnetic stripe.

11 In the preceding exemplary embodiments, we have described an
12 unloading of applications from an information processing device
13 to outside this device; in the case of Fig. 2, the card 21
14 performed an unload to the reader 20 or the data banks 23-25 of
15 Fig. 1; in the case of Fig. 4, the information processing device
16 constituted by the microprocessor 9 and its memories 10, 14
17 performed an unload to the optical stripe 42. According to
18 another variant of embodiment of the invention, an information
19 processing device performs an unload between several memories of
20 this device. For example, this information processing device is
21 constituted by the card 21 of Fig. 2, and the microprocessor 9
22 unloads an application from its RAM 14 to its nonvolatile memory
23 10.

24 For example, several applications K, J are stored in the ...
25 nonvolatile memory 10. First, the application K is run. At this
26 time, working information It_k related to the application K is
27 processed in RAM, while a program of the application K remains in
28 nonvolatile memory 10. This working information specifically
29 comprises:

- 1 - temporary working variables, involved in calculations;
- 2 - context variables, which allow the card to subsequently
- 3 resume an interrupted running of an application;
- 4 - subprograms.

5 At a given moment, the card must run the other application J and,
6 for this reason, must load working information It_j into the RAM.
7 If the card determines that there is not enough available space
8 in the RAM to receive the working information It_j , it decides to
9 stop the running of the application K and to unload the working
10 information It_k of the application K into its nonvolatile memory
11 10. Then, it runs the application J by loading the associated
12 working information It_j into RAM. After the running of the
13 application J, the card resumes the running of the application K,
14 at the point where it was interrupted, reloading the working
15 information It_k into RAM.

16 In this last variant of the invention, the load and unload
17 commands are not used, since the information processing device in
18 question has not told an external device to perform the
19 operations for loading and unloading its memories. It still has a
20 table TAB_APPLI, but the latter is simplified compared to the
21 table 2 above; the parameter "signature of the information" is
22 eliminated. In effect, since the information does not leave the
23 information processing device, it does not risk being altered
24 during its unloading.

25 In the above, we have particularly described the decision by
26 the card to unload a set of information as a result of an order
27 received by the card to load another set of information. It is
28 noted, however, that the invention also covers the case in which
29 the order received by the card is for executing an operation

1 other than the loading of a set of information. For example, a
2 particular process requested with the card may require a memory
3 space larger than the space currently available in the memory of
4 the card; this could be, among other things, a cryptographic
5 calculation. In this case, the card will decide to unload a set
6 of information in order to be able to execute this operation.
7 Another example is the one in which the order received by the
8 card is an order to run an application K that has previously been
9 unloaded from the card. The card must therefore reload this
10 application in order to run it; if there is not enough memory
11 space for this reload, the card will decide to unload another
12 application J, then perform the reloading of the application K.

CLAIMS

1. Chip card comprising information processing means (9) and main information storage means (10, 14), characterized in that the processing means comprise:

- means for detecting, during the operation of the chip card, that the main storage means (10, 14) contain a quantity of information such that the execution of an operation is not possible;

- means for selecting, in the main storage means, a set of information to be unloaded (K), the unloading of which can release enough space in the main storage means to allow the execution of said operation;

- means for unloading the set of information to be unloaded (K) into secondary storage means (23 through 25; 42; 53), in the event that said secondary storage means do not contain said set of information to be unloaded.

2. Chip card according to claim 1, which comprises a loading table (TAB_APPLI) stored in the main storage means and including a storage indicator indicating, for at least one set of information, whether or not the latter is stored in the main storage means, so that when the processing means (9) must have access to said set of information, they consult said storage indicator; and

- in a first case in which the storage indicator indicates that the set of information is stored, the processing means access the latter; or

- in a second case in which the storage indicator indicates

12 that the set of information is not stored, the processing means
13 send the secondary storage means (23 through 25; 42; 53) a
 command for loading this set of information.

1 3. Chip card according to claim 2, wherein the storage
2 indicator comprises a "loaded" state indicating that the
3 corresponding set of information has been loaded into the chip
4 card from the secondary storage means (23 through 25; 42; 53) and
5 an "unloaded" state indicating that the set of information has
 been unloaded by the chip card into the secondary storage means.

1 4. Chip card according to claim 1, which comprises a loading
2 table (TAB_APPLI) stored in the main storage means (10, 14) and
3 including a modification indicator indicating, for at least one
4 set of information, a first version of which has been loaded into
5 the chip card from the secondary storage means (23 through 25;
 42; 53), whether or not this first version has been modified in
 the chip card, so that when this set of information must be
 unloaded into the secondary storage means, it is not actually
 unloaded unless this first version has been modified.

1 5. Chip card according to claim 1, which stores at least one
2 set of information in two parts, i.e., a subset of application
3 information (p-cod) containing a program and general data for ...
4 running an application, and a subset of sequence information (p-
5 dat) containing specific data defining a particular session of
6 the running of the application, and which comprises means for
7 detecting that several sets of information have the same subset
8 of application information (p-cod) and respective different

9 subsets of sequence information (p-dat), so that it stores said
10 subset of application information in the main storage means (10,
11 14) only once, and so that it associates with this subset each of
said subsets of sequence information.

1 6. Chip card according to claim 5, which comprises:

2 - means for detecting, during its operation, that the main
3 storage means (10, 14) contain a quantity of information such
4 that the additional storage of a subset of sequence information
5 (p-dat) to be stored, associated with a subset of application
6 information (p-cod) already stored, is not possible;

7 - means for selecting, in the main storage means, a subset
8 of sequence information to be unloaded, associated with the same
9 subset of application information, the unloading of which can
10 release enough space in the main storage means to allow the
11 storage of said subset of sequence information to be stored;

12 - means for unloading this subset into said secondary
13 storage means (23 through 25; 42; 53) in the event that said
14 secondary storage means do not contain said subset of sequence
15 information to be unloaded; and

16 - means for storing in the main storage means the subset of
sequence information to be stored.

1 7. Chip card according to claim 5, which comprises a loading
2 table (TAB_APPLI) stored in the main storage means and including,
3 for each subset of application information stored, a maximum
4 number (s) of associated sequences that can be stored in the main
storage means.

1 8. Chip card according to claim 1, which comprises means for
2 reloading into the main storage means (10, 14) a set of
3 information previously unloaded into the secondary storage means
4 (23 through 25; 42; 53).

1 9. Chip card according to claim 8, which comprises a loading
2 table (TAB_APPLI) stored in the main storage means (10, 14) and
3 including, for at least one set of information (K) processed by
4 the device, a first signature (SGN-K) of this set of information
5 calculated by the processing means (9) before the possible
6 unloading of the set of information, with a signature key (SWAP)
7 stored in the main storage means, the processing means being
8 designed to calculate a second signature of the reloaded set of
9 information, in order to compare this second signature with the
10 first one, so as to validate the reloading of the set of
11 information in the event that the two signatures are identical,
12 and to invalidate the reloading of the set of information in the
event that the two signatures are different.

1 10. Method for managing the memory in a chip card comprising
2 information processing means (9) and main information storage
3 means (10, 14), characterized in that it comprises the steps
4 consisting of:

5 - detecting, during the operation of the chip card, that the
6 main storage means (10, 14) contain a quantity of information
7 such that the execution of an operation is not possible;

8 - selecting, in the main storage means, a set of information
9 to be unloaded (K), the unloading of which can release enough
10 space in the main storage means to allow the execution of said

operation;

- unloading the set of information to be unloaded (K) into secondary storage means (23 through 25; 42; 53), in the event that said secondary storage means do not contain said set of information to be unloaded.

11. Method according to claim 10, which comprises the steps consisting of:

- detecting, during the operation of the chip card, that the main storage means (10, 14) contain a quantity of information such that an additional storage of a given previously unloaded set of information is possible;

- reloading into the main storage means said set of unloaded information.

12. Method according to claim 10, which comprises the steps consisting of:

- detecting, during the operation of the chip card, that the main storage means (10, 14) contain a quantity of information such that an additional storage of a given previously unloaded set of information (K) is not possible;

- selecting, in the main storage means, a set of information to be unloaded (J), the unloading of which can release enough space in the main storage means to allow the storage of said previously unloaded set of information;

- unloading the set of information to be unloaded (J) into the secondary storage means (23 through 25; 42; 53), in the event that said secondary storage means do not contain said set of information to be unloaded; and

15 - reloading into the main storage means said previously
unloaded set of information (K).

1 13. Method according to claim 10, wherein said secondary
2 storage means comprise a data bank (23-25) remote from the chip
card and linked to it by a data transmission network (26).

1 14. Method according to claim 10, in which said secondary
2 storage means belong to an information processing device (20)
cooperating with said chip card (21).

1 15. Method according to claim 10, wherein said secondary
storage means (42; 53) belong to the chip card.

1 16. Protocol for communication between a chip card and a
2 chip card reader, the card comprising information processing
3 means (9) and main information storage means (10, 14),
4 characterized in that it comprises the steps consisting in that:

5 - the reader transmits to the card an order to execute an
6 operation;

7 - the card searches to see whether it has enough space in
8 the main storage means to execute this operation;

9 - if so, the card executes said operation, then transmits an
10 execution report to the reader;

11 - if not, the card selects in the main storage means a set
12 of information to be unloaded (K), the unloading of which can
13 release enough space in the main storage means to allow the
14 execution of said operation, then the card unloads the set of
15 information to be unloaded (K) into secondary storage means by

16 transmitting an unloading order to the reader, in the event that
17 said secondary storage means (23 through 25; 42; 53) do not
18 contain said set of information to be unloaded, then it executes
19 said operation, then finally, transmits an execution report to
the reader.

1 17. Protocol according to claim 16, wherein said operation
2 is a loading of a set of information to be stored (J), the steps
3 consisting in that:

4 - the reader transmits to the card an order to load said set
5 of information to be stored (J);

6 - the card searches to see whether it has enough space in
7 the main storage means to execute this loading order;

8 - if so, the card executes the loading order, then transfers
9 an execution report to the reader;

10 - if not, the card:

11 - transmits to the reader an order to suspend the
12 loading;

13 - selects in the main storage means a set of information
14 to be unloaded (K), the unloading of which can release
15 enough space in the main storage means to allow the
16 execution of the unloading order;

17 - unloads the set of information to be unloaded (K) into
18 the secondary storage means by transmitting an
19 unloading order to the reader, in the event that said
20 secondary storage means (23 through 25; 42; 53) do not
21 contain said set of information to be unloaded;

22 - transmits to the reader an order to resume the loading;

23 - executes said loading, then transmits an execution

report to the reader.

- 1 18. Process according to claim 16, wherein said order to
- 2 execute an operation consists, for the reader, of activating an
- electric power supply of the card.