# Chip card and Process for the Use of the Chip Card

The invention relates to a chip card, a terminal for use with a chip card and a process for using the chip card as well as a chip card system.

Micro-processor chip cards having a payment function, e.g. electronic purse, ec-cash, credit function etc., are already in use and, depending on their nature, preconditions are laid down by organisations, e.g. ZKA (Zentraler Kreditausschuß), VISA or EMV (Europay Mastercard VISA), so that they can be used as a means of payment ("cash substitute). By way of example the following may be mentioned here:

- ZKA, Zentraler Kreditausschuß, "interaction specification for the ec-card with chip", 27.10.1995;
- Europay International, "Integrated Circuit Card, Specification of Payment Systems, EMV'96", V3.0, 30-Jun-1996;
- ISO/IEC 7816-4, "Information technology - Identification cards - Integrated circuit(s) cards with contacts, Part 4: Interindustry commands for interchange", 01-09-1995;
- prEN 1546-1, "Identification card systems - Inter-sector electronic purse, Part 1: Definitions, concepts and structures", 15.03.1995;
- prEN 1546-2, "Identification card systems - Inter-sector electronic purse, Part 2: Security architecture", 03.07.1995;
- prEn 1546-3, "Identification card systems - Inter-sector electronic purse, Part 3: Data elements and interchanges", 09.12.1994.

An up-to-date survey of chip cards may be found in:

- Stefan Schütt, Bert Kohlgraf: "Chipkarten, Technische Merkmale, Normung, Einsatzgebiete", ("Chip cards, Technical Characteristics, Standards, Fields of Employment"), R. Oldenbourg Verlag, Munich/Vienna, 1996, ISBN 3-486-23738-1.

Conventional chip cards are, as a rule, usable only for a particular purpose, for example as electronic money purse or as an electronic identification means. However, the applications applied to these chip cards, as a rule, are static, i.e. they are applied during the manufacture of the chip card and are preserved unchanged throughout the life cycle of the card.

Conventional chip cards are limited both with regard to their variability as well as with regard to their functionality. In particular, conventional chip cards are fixed according to their manufacturing process with regard to their functionality and can no longer be changed.

It is accordingly an object of the present invention to provide a chip card, the functionality of which is variable.

A further object of the invention resides in providing a chip card for which the number and nature of the application performable by the chip card or the applications and transactions may still be variable, even after the manufacturing process. It should be possible to load onto these chip cards additional applications, it should be possible to delete applications from the chip card and the individual applications are to be defined independently from one another in data and security-technological respects and to proceed independently from one another.

The chip card should, for example, comply with the data and security-technological conditions according to ISO 7816; the individual applications of the card should, however, be independent in particular from the card platform itself.

It is a further object of the invention to provide a chip card with which the user himself may determine or combine or change the number and nature of the applications available in his card.

It is furthermore an object of the present invention to provide a chip card by means of which both intraservices (i.e. closed-loop applications within the sense that no bookings and service transfers involving external partners have to take place) as

well as interservices (i.e. applications with additional external relations involving external partners) are possible and can be performed.

According to one aspect of the invention an apparatus is provided by means of which one or more card applications can be loaded onto the card by each of which the performance of transactions between the cardholder and one or more service providers is made possible.

By such loading the chip card is so configured that it is provided with a new functionality, i.e. can perform an application which was previously not open to it. The loaded data cause applications being defined and realised in conjunction with the basic functionalities of the card such as the operating system which previously were not provided on the card. As a result the functionality of the card is extended by such loading of an application by this particular application.

According to an embodiment of the invention a data structure ( DF_VAS) is provided on the chip card according to the invention which in turn is subdivided into a partial structure and a definition data set, the data structure being distinctly identified by means of a coding identifying it and is thus independent of the card platform. So-called applications can now be loaded into the partial structure, i.e. functionalities or applications of the chip card. As a result it becomes possible by loading a particular application of the chip card, to perform transactions between the cardholder and a service provider specific for this application. A definition data set within the data structure contains information concerning the nature and/or the structure of the applications loaded into the partial structure. At least the definition data set, preferably, however, the entire data structure are secured by means of at least one system key against modifications and are modifiable only by using this key. Instead of a system key other security mechanisms as well or security means can be conceived by means of which securing against modifications may be provided, for example by a personal identity number (PIN) or other means serving such security.

By the above structure it is possible to load into the partial structure a variety of applications and also to delete these once again from the partial structure whereby

the card is rendered variable with regard to its functionalities or the applications which may be performed therewith. Loading and deleting of applications proceeds by writing application specific data and keys into the provided partial structure. By the provision of the system key and the data structure coding, the data structure permitting the multi-functionality is rendered independent from the card platform as such and also with respect to its security architecture self-supportingly from and independently of the card platform.

Dependening on the applications loaded into the partial structure, it is then possible to perform by means of the card transactions between the cardholder and service providers which are dependent on the loaded applications,.

Preferably, the chip card further comprises a transfer storage region into which the data to be exchanged during the performance of transactions may be written or from which they can be read. For reasons that for the reading of data from the transfer storage region terminal-specific keys are provided, the individual accesses are rendered independent from one another from a security point of view.

The individual partial structures or applications provided for in the card are preferably mutually independent and each allotted to a specific service provider. They represent, so to say, the data for performing a specific application, proprietary or specific to this service provider. Depending on the application type and depending on the service provider they, therefore, contain different information, for example data representing a specific value (bonus points, account balance etc.), information data concerning the application, information data concerning the service provider etc. Preferably, however, they in particular contain keys which are specific for the application, by means of which access to the data of the partial structure is rendered possible in a manner which security-technically is independent from other partial structures. Loading or generating the partial structure or application itself on the other hand, is safeguarded by at least one superimposed system key.

Preferably, prior to the performance of a transaction, a mutual authentication between the chip card and a terminal takes place, an application, respectively partial structure specific authentication key being provided for that purpose.

In order to perform transactions, data are then written into a partial structure reserved for the particular VAS-application or are read therefrom or are performed by way of the transfer storage region. In the first instance, application specific keys are used. In the last mentioned case, an application specific and preferably also terminal specific key is used which is, for example, generated by means of a key generating key provided on the card and from application specific data. The data thus written into the transfer storage may then be withdrawn by the service provider by way of the terminal, which preferably takes place by marking as invalidated the data stored in the transfer storage. If this involves a service provider who is not specific for the application being written, this involves the performance of a so-called interservice, i.e. monetary data are exchanged for the benefit of or the cost of the cardholder between different service providers.

Furthermore, for additional security, a PIN or a password is provided for verification of the right to perform a transaction procedure.

The variable structure of the card permits the accommodation on the card at different times of a variety of applications in a variety of numbers.

The veracity of data withdrawn from the transfer storage is preferably secured by the use of a signature key or by using a digital signature or at least a key. However, in this context it is particularly advantageous if the data withdrawn remain in the transfer storage and are merely labeled as having been withdrawn by way of the card. In this manner it is still possible to obtain information concerning the transaction performed even after the transaction has taken place. In this manner and by means of the securing of the withdrawal it becomes provable that a transaction has only been performed once.

It is furthermore particularly advantageous if the data written into the transfer storage are encoded with an expiry date after which they lose their value. Thus, it becomes possible to perform applications which, for example, involve the making available of a ticket valid only for a specific period.

By means of a transaction counter which is preferably provided, transaction dates, respectively the value data with respect to their associated transaction can be positively determined and identified.

In an advantageous embodiment, the chip card according to the invention therefore comprises a hierarchical database concept which at its individual planes is secured by means of different keys against modifications, which at the level of the applications regarding the application stored in the database is variable, each individual application being secured in relation to the remainder by its own specific key and being independent of the remainder, and the overall structure being secured by at least one system key and by coding identifying the structure and being independent of the card platform itself. The concept of the transfer storage permits the exchange of data both between cardholder and service provider as well as between different service providers as such. Also, reading from and writing into the transfer storage is secured by keys, these being likewise card and application specific, additionally also, terminal specific. An authentication performed prior to each transaction as well as optionally a PIN or a password additionally increase the security of the chip card according to the invention.

In patent claims 21 to 30 a terminal for use with the chip card according to the invention is defined. This terminal serves for loading or deleting applications, for performing transactions, for viewing of data as well as for performing further functions in conjunction with the respective applications and transactions. A process for performing transactions between the cardholder and service provider is defined in claims 31 to 33, the loading of data onto a chip card according to the invention is defined in claim 34 and 35, and claim 36 defines an overall system comprising a chip card and terminal.

In the following the present invention is further described by way of a preferred example, with reference to the accompanying drawings. There is shown in:

Fig. 1 diagrammatically, the chip card according to the invention,

Fig. 2 A schematic representation of the overall system of the components of the invention,

Fig. 3 the data flow in the overall system,

Fig. 4 the possible application classes and operations through a transaction model in schematic illustration,

Fig. 5 a schematic illustration of the security architecture of the chip card according to the invention,

Fig. 6 the data structure of a general implementation class of the VAS-container,

Fig. 7 a variety of implementation classes of the VAS-container according to a working example of the present invention,

Fig. 8 the database structure of the VAS-container according to a working example of the present invention,

Fig. 9 schematically, the database structure of the implementation class DF_PT,

Fig. 10 schematically, the database structure of the implementation class DF_AD.

Before the invention is further described, a number of terms used will be defined in what follows:

*VAS*: Value Added Services

*VAS-card*: The VAS-card is a chip card by means of which it is possible to participate in the Value Added Services. The VAS-card besides other applications such as e.g. payment applications (i.e. electronic money purses) contains the *VAS-container*.

*VAS-container*: The VAS-container comprises data structures, access conditions, keys and (supplementary) commands for administering VAS-applications and for the availability of the functionality of the VAS-applications.

*VAS-application*: VAS-applications contain VAS data. Access to the VAS data is controlled by way of the VAS-application. A *VAS-provider* operates in the VAS-container one or more VAS-applications. The utilisation of the VAS-application is defined by the application, reading and processing of VAS data. A VAS-application may take the form either of an *intra-* or an *interservice*.

*VAS-provider*: The VAS-provider is responsible for his VAS-application which he develops according to basic conditions of the *system operator* and according to his own discretion and thereafter makes available for use to the *cardholder* by way of the system operator and the *terminals*. The VAS-applications are to be loaded into the VAS-container of the VAS-card prior to it being possible to participate therein.

*Intraservice*: Intraservice is a kind of VAS-application utilised under the exclusive management of the respective VAS-provider. Intraservice applications are closed-loop applications, meaning, that no accounting or performance transfers with external partners take place. A VAS-application may be either of an intra- or an *interservice* nature.

*Interservices*: Inter service applications are intra service applications which by way of additional external connections interact with external partners. A VAS-application may be either intra or inter service by nature.

*SO, system operator*: The system operator offers to the VAS-providers and the *cardholders* the VAS system for use by them.

*Issuer*: The issuer or card issuer brings the VAS-cards, including VAS-containers, into circulation.

*CH, cardholder*: The cardholder or card proprietor is the person who owns and employs the card (in this case: VAS-card) in order to participate in the Value Added Services. This person is not necessarily the actual owner of the card.

*Service terminal*: The service terminal is set up by the system operator for VAS-applications. At the service terminal the cardholder may administer the VAS-applications on his VAS-card (loading, viewing, deleting and transfer of VAS-applications).

*Dealer terminal*: The dealer terminal possesses payment functions and in addition offers VAS functionality. This is where the cardholder applies his VAS-card in order, on the one hand, to make payment and, on the other hand, to participate in the benefits of VAS.

*AID*: (Application Identifier), name of applications no longer than 16 bytes for the unambiguous differentiation of applications and the application selection from outside without knowledge of the database structure of a card. The AID consists of a Registered Application Provider ID (RID), 5 bytes long, and optionally a Proprietary Application Identifier Registration (PIX), not longer than 11 bytes.

*DF*: Directory File according to ISO 7816

*EF*: Elementary File according to ISO 7816

*Valid VAS-container*: VAS-container which has the ability to authenticate itself in relation to the outside world.

*KID*: (Key Identifier) number of a key within an elementary file containing keys

*R&R*: Rules and Regulations

*p*: Maximum number of objects of the implementation class DF_PT

*a*: Maximum number of objects of the implementation class DF_AD

$nr_{DIR}$: Maximum total number of objects: $nr_{DIR} = p + a$

The number of records in EF_DIR is $nr_{DIR}$

$nr_{EF_TRANSFER}$: Number of records of the EF_TRANSFER

Fig. 1 shows schematically the structure of the chip card according to the invention. In addition to the static, immutable data applied during the manufacturing process such as the master file MF and the (optionally provided) money purse function DF_purse, there is further provided on the card in accordance with the invention an index or a data structure or database structure DF_VAS. This serves for accommodating additional functionalities, so-called Value Added Services, VAS. On the basis of these additional functionalities which represent applications which even at a later instance, that is to say after the card has been manufactured, may be loaded onto the card, the card in respect of its functionalities and the transactions which may be performed therewith, is flexible and variable. The so-called VAS-container (DF_VAS) provided on the chip card in accordance with the invention, permits the variability and flexibility of the chip card in respect of its functions and besides, releases the applications accommodated on the chip card security-technologically from the card platform, rendering these independent of the card platform and permitting these even to be transferred to another card.

The new additional functionalities according to the invention (Value Added Services, VAS) are realised by way of microprocessor chip cards. The conversion of these additional functionalities is brought about by the VAS-container. The VAS-container

on the microprocessor chip card constitutes the platform for accommodating the VAS-applications. The VAS-applications are the respective realisations of specific additional functionalities.

In the case of the electronic money purse payment by means of the card (payment function) and the utilisation of a VAS-application (additional functionality) is performed by way of separate mechanisms; in the hands of the card user or cardholder these may, however, appear as a single procedure.

The microprocessor chip card is extended by the VAS-container which can accommodate a variety of independent applications. It makes available functions for the application, deletion and transfer of such applications; these may only be used by the authorised system operator. The VAS-container is independent of other system components on the microprocessor chip from a data and security-technological point of view. The VAS-container is totally self-defined and functional on its own. An independent security architecture is defined for it, so that VAS-applications utilise independent security functions. The security architecture utilises card specific keys which are not manufacturer specific and which are independent of the identification features of the card platform.

The VAS-container also uses mechanisms for deriving terminal specific keys. By means of these the VAS-container itself may actively check the authenticity of terminals, respectively of the data generated by them.

In the VAS-container the VAS-applications are filed which by way of mechanisms of the VAS-container render data available, thereby bringing about the control of the associated interface points. The VAS-container permits and controls also the secure interchange of data for interservices between partners. The VAS-container actively takes care of the control, i.e. the authenticity and the uniqueness of the transferred data values.

An advantage of the VAS-container as compared with other approaches to multi-application cards is that this concept is independent of a specific card platform. It

offers a security architecture which is independent of platform specific security mechanisms (such as keys, identification data, PIN, signature procedures).

A further advantage of the concept of a VAS-container is that the number of different applications on the card is not rigidly predetermined by limitations and preconditions at the time of card manufacture or card issue; the current loading of the card with applications may be freely selected by the card user and is restricted only by the storage capacity available on the specific card. The number of VAS-applications loaded onto a card at any given time depends on the actual use of the card. The card user individually assembles the VAS-applications on his card; this also involves modifying the combination at a subsequent instance. The VAS-container permits a multifunctional card in which the card functionality during the life cycle of the card may be assembled and used in a variable manner with regard to number and nature of applications. It is thus also possible to load onto and use with a single card applications for which in the past individual special cards were needed. VAS-applications may also be transferred to other cards. Accordingly, VAS-applications may survive beyond the life cycle of a card; they accompany the card user during the life cycle of such applications.

The microprocessor chip card with additional services is a suitable medium for the distribution and marketing of services for which access must be had to protected data. This microprocessor chip card may be used as a means of payment, a counting unit and for value storage. It may be provided with additional services and be used for the control thereof in accordance with client wishes by the client himself and after the card has been issued, in a flexible manner. It also actively controls the authenticity of participating terminals and secures the uniqueness and authenticity of transferred data.

Fig. 2 shows a system diagram. It illustrates the system components. A system operator makes the system available. VAS-applications may be loaded, deleted and transferred at the service terminals (ST); further operations are: selection, viewing, interpretation etc. of VAS-applications.

The providers of VAS-applications, the so-called VAS-providers design their own VAS-applications according to the basic conditions of the system operator and, as far as possible, according to their own wishes. The corresponding terminal programmes may subsequently be checked for authenticity by closure with a digital signature.

Fig. 3 illustrates the data flow in the system.

The VAS-applications are made available for use at the terminals of the card users by the system operator. VAS-applications must be loaded into the VAS-container of the chip card according to the invention, prior to participation therein being possible.

At the dealer terminal (HT) the VAS-applications present on the card are utilised by the application or deleting of VAS-data.

In order to provide the microprocessor chip card with VAS-functionality, the VAS-container is applied onto the card in addition to existing applications (e.g. payment applications such as the electronic money purse).

The VAS-container uses functions which permit entering, deleting and transferring of VAS-applications. These administrative functions are used exclusively by the system operator and are secured in the card against foreign use. The VAS-container includes a transfer data storage by means of which exchange of data between VAS-applications is brought about.

Two commands, TRANSFER and TAKE are employed for controlling the transfer storage. The command TRANSFER generates an entry in the transfer storage with data specific for the respective application. Besides the utility data these furthermore include entries such as date, expiry date and identification data needed for the control of the processing. By means of the command TAKE objects are withdrawn from the transfer storage and marked as having been withdrawn. Depending on the specific application, the objects are then marked either as remaining valid or as

being invalidated. Transferred data are checked by the VAS-container in respect of authenticity and uniqueness.

VAS-applications utilise VAS-data for making available and controlling the applications. Access to the VAS-data is controlled by the VAS-application utilising mechanisms which are made available to all applications in the VAS-container. A VAS-provider operates in the VAS-container one or more VAS-applications. The use of the VAS-application is defined by the entering, reading and processing of VAS-data.

The VAS-container supports interservices. Interservices require access to common data, transfer of service demands and the invoicing of services between different partners.

In the following the services and applications made possible by the chip card according to the invention are to be listed with reference to examples.

In each case the description relates to the realisation and function according to the present state of the art. In future the function is to be mimicked by one or more VAS-applications.

Firstly intraservices are to be presented.

Example A: Customer club

A department store operates a customer club. The customer becomes a club member and in conformity with this status receives specific club services which are not available to non-members. Nowadays a club member identifies himself in the club environment by a club membership document. The club membership document is prepared when joining, is non-transferable and, as a rule, has a limited term. No specific transactions are performed by way of the club membership card, i.e. there is no linkage with the customer turnover. In this manner the club membership card is

separated from bonus programmes in which a connection exists between status and turnover.

Analogue: Wholesaler membership card, senator card, book-club card

Object: The club membership is to be evidenced by an application in the VAS-container.

Example B: Bonus system

A customer is credited for each transaction by a bonus claim. The bonus claim is cumulated and may at the time determined by the customer be exchanged for a monetary benefit. The bonus claim is valid for a defined period and may be administered anonymously or person-related. The bonus claim arises from turnover or user frequency.

Analogue: Points status of Miles & More

Object: The points account is to be administered by an application in the VAS-container.

Example C: Discount

The customer receives a scheduled turnover discount. The discount is allowed for each individual transaction. The card administers no turnover history. Each transaction stands alone.

Object: The discount entitlement is to be authenticated by a VAS-application.

Example D: Identity document function

A person is enabled by features in the card to prove to third parties his entitlement to predetermined services. The association of the person with the identity document

must be authenticated for each transaction (photograph, PIN, biometrics). The genuineness of the identity document is used as a security feature.

Analogue: Internet access, access to homebanking, telephone card

Object: The authorisation is to be proven by a VAS-application.

Example E: Value units

Value units are purchased and consumed by single or multiple uses. By each transaction the value is reduced by one or more units. The user entitlement is transferable and may be subject to limitations as to usage.

Analogue: Single travel ticket, multiple travel ticket, subscriber concert ticket, squash 10-points ticket, cinema ticket, telephone units

Object: The transaction is to be rationalised by a VAS-application.

Example F: User invoicing

The use of a service is registered in terms of time, frequency or quantity and is invoiced according to a tariff. The extent to which the service will be called upon is not known in advance.

Analogue: Meals voucher, short-term parking ticket

Object: By a VAS-application invoicing according to tariff is to be permitted. The invoicing data applicable to each specific instance, are to be stored in the VAS-container.

Example G: Data record (mobile database)

This application permits the transfer of data of the cardholder to the VAS-provider. Transactions are thereby automated which at present must still proceed manually. No monetary transfer can be derived from these data.

Analogue: Completion of numbers pool vouchers, shopping lists, cash receipts, telephone registers

Object: A VAS-provider is to derive the data from the card so as to be able to directly provide the required service (e.g. provide a telephone connection, compile desired purchases, electronic completion and recording of numbers pool ticket). The data may be stored in the card either short-term or long-term.

These examples of intraservices are now to be followed by some examples for interservices.

An interservice arises whenever a plurality of VAS-providers participate in a service. This is invariably connected with data leaving the environment of one VAS-provider. To date this is performed by paper records.

Example A: Travel cost refund

A department store refunds to a customer the travel ticket of a public transport enterprise for travelling to the department store. The customer must produce to the department store the single travel ticket. The department store notes on the ticket that it has been refunded. It may be that the department store in turn receives from the transport enterprise part of the refund to the customer.

Object: The refund procedure is to be performed electronically.

Example B: Travel voucher

A department store refunds to the customer when making a purchase the cost for a homeward trip by issuing a voucher. The customer at the ticket office receives a ticket from the public transport enterprise or pays a lower price for the ticket. The transport enterprise invoices the voucher to the department store.

Object: Mimicking of the mechanisms by VAS-applications involving electronic accounting between the trade and the public transport enterprise.

Example C: Customer parking

A department store refunds to the customer part of the parking fees when using a particular parking garage. The parking garage is operated by an independent enterprise and receives from the department store a monetary payment for each customer credit.

Object: Mimicking the mechanisms by VAS-applications involving electronic accounting between the trade and the parking garage.

Example D: Multilateral bonus programme

A group of trading enterprises and service providers agrees on a joint bonus programme.

Object: Each partner may credit or debit bonus points against a common account on the card. Accounting for the services between the partners proceeds by the underlying system.

Example E: Recognition of bonus points between service providers.

Each service provider operates its own bonus programme but has entered into agreements with others to recognise collected points. Known examples are agreements between car rentals and airways for the collection of "miles".

Object: Supporting the transfer of bonus points via the card. Each VAS-provider is initially to collect his points for himself without interference by somebody else. Certain mechanisms are to be made available for the transfer.

Example F: Night taxi

By purchasing a travel ticket of the public transport corporation, the right to travel in a collective taxi (e.g. after 22h00) is simultaneously acquired. For accounting purposes, the taxi enterprise must prove that a travel ticket had been presented. The use of the taxi is noted on the ticket in order to avoid abuse.

Object: The VAS-application is to permit the utilisation, control and accounting of this service.

In the following the structuring of the chip card according to the invention will be further described.

More particularly, the microprocessor chip card with VAS-functionality contains the VAS-container.

The VAS-container constitutes an application in its own right and exists either alone or even in parallel to other applications on the basic card platform. The VAS-container is completely self-defined and functional on its own. It can be operated without the payment function usually present on the same card. In particular, an independent security architecture is defined for the VAS-container so that VAS-applications can use independent security functions.

Part of the Value Added Services are transactions by the client performed at terminals of the VAS-providers. The VAS-provider has an interest in monitoring these transactions, be it for the purpose of system control or for the collection of statistic and other data. In order to optimise and unify the data structures in the card, the use of application specific identifications is not recommended and the use of an

unambiguous VAS-container ID applicable to the whole system is advised. This number can be used by the VAS-provider for practising the abovementioned functions and it releases him from the burden of administering his own numbering systems.

The security architecture of the VAS-container uses this system wide unambiguous ID in order to derive card specific keys. The use of the card specific number would be possible in principle but is preferably not used because, should the VAS-container be implemented on different platforms, these may in certain instances use clashing numbering systems.

The VAS-container ID is issued by the system operator and entered by the card issuer when generating the VAS-container. It is destroyed on deleting the VAS-container and thereby removed from the system. A VAS-container is deemed as having been deleted if it contains no VAS-applications and if the VAS-container ID has been removed.

During a total transfer of the VAS-container from an old into a new card, the VAS-container ID is transferred jointly with the VAS-applications. This transfer corresponds to a removal of the VAS-container from the old into the new card. After this procedure the old card no longer contains a VAS-container. The VAS-container present in the new card is overwritten during this operation and thereby deleted. Since the VAS-container ID is transferred from the old to the new card, no transformation of the reference numbers is necessary in the basic systems of the VAS-providers.

Individual VAS-applications can be transferred between two different VAS-containers only under the control of the prevailing VAS-provider. During this function the association between VAS-container ID and VAS-application is changed which may have to be recorded in the basic system of the VAS-provider.

The VAS-container contains no personalised features. VAS-applications may contain personalised data, the extent thereof should, however, be kept to a

minimum for data protection reasons and for improved memory utilisation. VAS-applications should, if necessary, store person-related data in the basic system and create the association with the card by way of the VAS-container ID.

The support of interservices constitutes an important feature of the VAS-container. Interservices require access to common data, transfer of services claims and the accounting of services between different partners. The VAS-container must make these possible and, in spite of this, warrant the security of the applications in the intraservice.

So-called intraservice VAS-applications are applications which are operated under the exclusive control of the VAS-provider. The VAS-provider defines the security of its application independently from any external entity. No external party can modify VAS-data without proliferation of the key.

For the efficient performance of interservices the partners must be able to access common data. The *joint* access to the data is realised by way of multiple-step security mechanisms.

The first step of the joint access proceeds exclusively by way of public *transfer fields*. VAS-providers may interchange data by way of these fields without mutual knowledge of the applications and keys being necessary. The terminals only require suitable keys for the entering of data into the transfer fields but not for reading. Anybody may read without limitation. The data interchange may proceed in both directions between VAS-provider and partner if each has his key available for data entering. The transfer data may be generated from an intraservice VAS-application of the VAS-provider or, vice versa, the VAS-provider may enter data from the transfer field into his intraservice VAS-application.

The second stage is the cancellation of value units represented by VAS-data. A VAS-provider introduces value units into the VAS-data by means of a special key for entering data. The value units may be consumed by interservice partners who have available a key for unit cancellation obtained from the overall responsible VAS-

provider. In this stage partners having mutually different rights interact with non-public VAS-data.

The third step is the direct data entry access to the VAS-data by all participating interservice partners. This method requires an appropriate degree of trust between the parties, because VAS-data can be modified without limitation.

The transfer field serves as a connecting link between VAS-applications. Data in the transfer field are generated by the VAS-applications in the VAS-container. Data may also be entered directly into the transfer field if the person doing so operates no VAS-application of his own in the VAS-container.

The transfer field is in principle available to all applications, however, entries may only be made by whoever is entitled thereto (by a key). Only receivers who are entitled thereto in terms of the regulations, i.e. the Rules and Regulations R&R may remove data from the transfer field. The receiver checks for the presence of transfer data addressed to him and withdraws these for processing in his own system.

In order to prevent the manipulation of transfer data in a public interchange, the generator introduces an authenticity feature and the VAS-container introduces a sequence number. The uniqueness of a transfer report is secured by these elements and the origin of the data is demonstrated.

In case of need, the receiver of transfer data is furnished by the generator with the means for performing an authenticity check. If this is not desired, the acceptance of the data may take place in good faith; in that event, the authenticity feature may possibly only be checked when accounting takes place in the basic system of the generating VAS-provider.

The withdrawal of data from the transfer field is possible only once without loss of an authenticity feature. On withdrawal, the data are marked and remain (as a copy) in the transfer field. This permits proof of the transfer procedure even after the withdrawal of the data for a certain period.

Data in the transfer field carry an expiry date. Expired data may be overwritten when required by arbitrary applications. The data in the transfer field may be marked on withdrawal thereby to be freed for immediate overwriting. Should the transfer storage nevertheless be completely filled before the expiry of a set of transfer data, the cardholder must delete no longer needed entries at the service terminal.

For the modelling of VAS-applications 3 *operations* are defined. These operations act onto the VAS-data. Loading and deleting of applications are functions of the VAS-container and are not considered in the following paragraphs.

*Purchasing:* In general, the purchase of an entitlement to services and the filing of proof thereof in the VAS-data of a VAS-application.

*Cancelling:* Generally the redemption of the entitlement without or with complete or with partial consumption of VAS-data of an application. The procedure generates an electronic receipt which is filed in the transfer field. This receipt bears an appropriate expiry date by which it may be deleted from the transfer storage.

*Withdrawing:* Generally the retrieval of the electronic receipt from a transfer field for further processing (e.g. basic system). A copy of the receipt remains and serves as proof of the "cancellation".

"Acquiring" of VAS-data may only proceed through the respective VAS-provider. "Cancellation" of VAS-data may take place only by way of the VAS-provider who generated these by "purchasing" or by an interservice partner authorised thereto by the VAS-provider. "Withdrawing" may be performed also by any other VAS-provider. An interservice without equal rights of access to the VAS-data causes triggering the status transition "withdrawing" by the partner. The electronic receipt thereby recovered may then be accounted for by way of the system operator and the basic system of the VAS-provider. "Purchasing" and "cancelling" may take place in a single step.

VAS-applications with common features may be subdivided into classes. These classes constitute the basis for data structures in the VAS-container. The VAS-provider during the implementation of his application selects an application class.

In this context the following *application classes* are defined:

- Points database
- Ticket
- Identification document
- Voucher
- Database

*Points database* denotes an application class in which an account of points values is administered. Onto and from the points account values may be credited and withdrawn. The crediting of values proceeds through the VAS-provider in that the database has entered thereinto the new accounts balance. The debiting of values proceeds by way of the "cancellation" operation in which context an electronic receipt is filed in the transfer storage as evidence. Access to the two functions is brought about by two different access keys.

*Ticket* denotes an application class in which a value field exists which may be cancelled and thereby consumed either once or several times. Crediting of values in the application class Ticket is performed once only.

*Identity document* denotes an application class in which the VAS-data receive evidence of an entitlement. The evidence is generally not cancelled by use; however, after predetermined criteria, e.g. after an elapse of time, it becomes invalidated. Depending on the definition of the application the authenticity of the identity document (e.g. neighbourhood parking authorisation) or the act of presenting the document is documented (e.g. collective taxi trip using a monthly ticket: Generation of an electronic receipt by the card. The receipt is submitted by the taxi enterprise to the public transport corporation and is accounted for pro rata).

Optionally the usage of the identity document may be documented in the transfer storage of the card by electronic receipts.

*Voucher* denotes an application class by means of which the service entitlement (as a rule for short duration) is placed in an interim store of the card. These electronic vouchers are stored exclusively in the transfer storage of the VAS-container. The application utilising the voucher is, as a rule, different from the generating application. The withdrawal of the voucher from the transfer store is possible once only and is documented by the card. An authenticity feature generated by the VAS-container may by used during accounting in the basic system.

*Data memory* denotes an application class in which a VAS-provider stores data in a VAS-container to provide an additional service for the client (e.g. last menu in a fast-food restaurant, last number pool numbers, service preferences, proposal lists). These data are for information only and represent no claim to services against the VAS-provider. Access to the data is controlled by the VAS-provider.

Each application class is characterised by a typical user cycle. The following table illustrates the frequency at which the three above defined operations are applied to the VAS-data of the application class (*Please note*: "Purchasing" does not mean "loading application" and "withdrawing" does not mean "deleting application").

Table 1 - User Cycle

| | Points database | Ticket | Identity Document | Voucher | Data memory |
|---|---|---|---|---|---|
| Purchasing | Multiple | Single | Single | Single | Multiple*** |
| Cancelling | Multiple | Multiple | Multiple | Single | Never |
| Withdrawing | Multiple | Multiple | Multiple | Single | Never |

*** In the application class "data memory" no entitlement is acquired, the application merely enters data into the application.

Fig. 4 illustrates the above listed application classes and operations by way of a transaction model.

In the following the security architecture of the chip card according to the invention will be further dealt with. In order to ensure security the following keys are defined:

Table 2 - List of Keys

| Name | Place | Holder | Description |
|---|---|---|---|
| $K_{SO}$ | VAS-container | System operator | Key for administering the VAS-container |
| $K_{AUT}$ | VAS-container | System operator | Key for authentication of the VAS-container |
| $K_{SIG\ VASC}$ | VAS-container | System operator | Key for encoding the transaction data |
| $KGK_{DEC}$ | VAS-container | System operator | Key for the derivation of the $KGK_{DED.\ PIX}$ |
| $K_{LVASP}$ | VAS-application | VAS-provider | Key for writing access to VAS-data |
| $K_{RVASP}$ | VAS-application | VAS-provider | Key for reader access to VAS-data |
| $KGK_{DEC.PIX}$ | VAS-provider | VAS-provider | Key for the derivation of $K_{DEC}$ |
| $K_{DEC}$ | Dealer terminal | VAS-provider | Key for authentication of the trader terminal |

The security architecture is based on the life cycle of the VAS-container or VAS-applications and is stratified according to the responsibility of the participating instances. An explanatory schematic illustration is apparent from Fig. 5.

In what follows, some elucidations of the structure of the VAS-containers as well as of the applications applied thereon are provided.

The VAS-container as well as VAS-specific supplementary commands are either applied onto the card by the issuer jointly with other non-VAS-applications or at the latest are integrated at a service terminal by an authorised VAS-provider into an existing card platform at a service terminal.

For the second possibility the following mechanism may find application: The system operator agrees with the issuer on a temporary key $K_{SO}$*. The issuer opens the card with his key which is only known to him, establishes the files of the VAS-container and in particular writes the $K_{SO}$* into the DF_VAS. The system operator may then at

a later instant (e.g. the card enters into contact with a service terminal for the first time) replace the key $K_{SO}$* by his own key $K_{SO}$ which is then only known to himself. The system operator may now himself enter further data such as, for example, the $KGK_{DEC}$ or may himself in the case of a platform with dynamic memory administration generate or delete files for VAS-applications. It is thereby ensured that the issuer after the first use of a VAS-card no longer has access to the VAS-container and that the system operator only has access to the VAS-container. Because of the absence of any common data structures and data interchange with other applications, the security architecture of the VAS-container is thus independent of other applications present on the card platform.

In a special embodiment of the invention use is, however, to be made of the *first possibility*: The issuer is required on instructions of the system operator to apply the VAS-container including all keys onto the VAS-cards.

The VAS-container consists of a predetermined data structure, predetermined access conditions (Acs) and some global data. The global data contain inter alia the key $K_{SO}$ for loading or deleting of VAS-applications. It is ensured by way of this key, known only to the system operator that only permitted VAS-applications can be loaded. For this purpose the card requires an external authentication by the system operator via $K_{SO}$.

Whenever a cardholder wishes to load a VAS-application at a service terminal, the applicable VAS-provider instructs the system operator accordingly to do this on his behalf. When loading the VAS-application into the VAS-container the VAS-provider transfers the keys $K_{LVASP}$ and $K_{RVASP}$ to the system operator who then enters these into the application. The key $K_{LVASP}$ permits the VAS-provider to protect data of the application against written access and in additional internal data against reading access. For this purpose the card requires an external authentication of the VAS-provider based on $K_{LVASP}$; i.e. the card checks actively the authenticity of the terminal. After successful performance of this function, a terminal is permitted written access to a VAS-application and reading access to internal VAS-data of the application. An internal authentication, i.e. checking the VAS-application (and

thereby the card) as to authenticity by way of the dealer terminal, may take place optionally. When the VAS-application is used by the cardholder, these internal VAS-data may be written into the application or be modified at optional terminals which have access to the key $K_{LVASP}$. The function "purchasing" is supported therefor by an UPDATE RECORD command which must be preceded by an external authentication by means of the key $K_{LVASP}$.

Reading access to all non-internal data of the VAS-application is only permitted if a previous external authentication by $K_{RVASP}$ or by $K_{SO}$ or by the correct entry of a PIN or a password by the system operator has taken place. The PIN/password protected access is provided in order to permit the cardholder to inspect data at the terminal or at the wallet. The cardholder may elect at a service terminal to activate or deactivate a PIN/password protection for reading the value or status data.

The global data of the VAS-container are associated with a key $K_{SIG_VASC}$ for signing data extracted from the transfer field. By way of the signature the intactness of these transaction data can be checked whenever they have to be transferred protected against manipulation between interservice partners for mutual accounting. In addition to a signature optionally introduced by way of the interservice partners, a signature based on $K_{SIG_VASC}$ and a transaction counter administered by the VAS-container are added for sets of data which the card issues by way of the operation "withdrawing". Because on the one hand reading of the transfer field is permitted by any one but, on the other hand, the signature of the card via $K_{SIG_VASC}$ can be generated only when calling for the operation "withdrawing" (and this can happen only once) any unpermitted double accounting for vouchers is recognised. Each interservice partner is able to have the authenticity and uniqueness of a withdrawn voucher acknowledged by the system operator. In addition, the authenticity of the VAS-container can be verified by checking the signature by the system operator.

In the event that a card platform employs asymmetrical key procedures, it is possible also to employ as $K_{SIG_VASC}$ a private (secret) key within the card for signing purposes or to derive such a key from a private Key Generating Key. The VAS-providers may

in such case employ public (non secret) keys for themselves checking the signature without having to consult the system operator.

Part of the data of the VAS-container is formed by a global Key Generating Key $KGK_{DEC}$, which has the ability to generate the access key $K_{DEC}$ for all terminals of all VAS-providers for the authority check for the operation "cancelling" (the subjects key derivation and checking will be dealt with again further below). The cancellation of monetary value data are not subject to the same security measures as the loading or purchasing of an entitlement. For this purpose it is sufficient to use a global key instead of a key specific to the application. This global key is converted initially by derivation into an application key and thereafter by renewed derivation into a terminal key. The VAS-provider is only informed of the application specific derivation of the global key for generating the own terminal key. This is described briefly in what follows:

We denote as **mac(k, d)** the calculation of a message authentication code for a message d and a DES-key k by means of DES. As long as the message is no longer than 8 bytes this corresponds to the codification as such (presupposing ICV=O). We denote as **macp(k, d)** the calculation of mac(k, d) with subsequent adaptation of the parity bits. The result k' = macp(k, d) is once again a valid DES key.

The calculation of the application specific terminal key then proceeds as follows:

1. Each card stores a key

   $$KGK_{DEC},$$

   which is equal for all cards and which is kept secret by the system operator. The system operator provides that this key is personalised into the card.

   From this key all other VAS-application and terminal keys can be derived.

2. A VAS-provider wishes to introduce a VAS-application A with $AID_A=RID_{VAS}.PIX_A$. The system operator now calculates from the key $KGK_{DEC}$ and the PIX the application specific key

$$KGK_{DEC,PIX}=\mathbf{macp}(KGK_{DEC}, PIX_A)$$

and hands this key over to the VAS-provider.

3. This VAS-provider, by means of their terminal ID derives from $KGK_{DEC, PIX}$ for all terminals which the TRANSFER command is supposed to apply to the VAS-application A, their specific keys:

$$K_{DEC}=\mathbf{macp}(KGK_{DEC,PIX}, \text{Terminal ID}) =$$
$$\mathbf{macp}(\mathbf{macp}(KGK_{DEC}, PIX_A), \text{Terminal ID})$$

The VAS-provider stores these keys in his terminals. In the event that the VAS-provider does not himself operate the terminals, he generates the keys and distributes these to the operators of the terminals.

4. The VAS-card only performs the TRANSFER command if the command comprises a cryptogram C generated by the terminal which is formed by way of specific data *data* of the message in the transfer storage. The card itself recognises $KGK_{DEC}$ and obtains from a terminal besides the user data *data* and the cryptogram C the terminal ID as well as the PIX (or alternatively the card itself knows the PIX, for which see also the description of the command TRANSFER). The card is now able to calculate the cryptogram C' by way of the user data:

$$\mathbf{mac}(\mathbf{macp}(\mathbf{macp}(KGK_{DEC},PIX), \text{Terminal ID}),data) =$$
$$= \mathbf{mac}(\mathbf{macp}(KGK_{DEC,PIX}, \text{Terminal ID}),data) =$$
$$= \mathbf{mac}(K_{DEC},data) =$$
$$= C'$$

The card now compares the cryptogram C' calculated by itself with the cryptogram C calculated by the terminal. If they differ, there takes place a termination of the transaction and a fault report. In the case of congruence the VAS-card will execute the TRANSFER command.

The various security measures are in line with the different constructions of service terminals or dealer terminals (for loading or purchasing respectively) and simple dealer terminals (for cancelling). In order to perform the "cancellation" operation a terminal must authenticate itself to the card by having knowledge of a $K_{DEC}$. If that key $K_{DEC}$ is compromised, the aggressor can merely perform the function of this single terminal. However, this procedure is recorded in the card. The documentation contains an unambiguous sequence number, the cancellation and the terminal ID. The VAS-applications utilise the security services of the VAS-container in order to regularise access to VAS-data. The execution of VAS specific functions is limited to authorised parties by the definition of access rights.

In general, it is necessary that for each VAS-application publicly accessible data regions (e.g. for reading of values by means of a wallet) and private data regions of the responsible VAS-provider are available which the latter may protect against access by third parties (e.g. internal administrative data).

Having regard to the fact that according to ISO 7816-4 cards do not provide differentiated access protection for individual Records of Elementary Files (EF), it is necessary to use a plurality of files, each containing a record for displaying different sights on a VAS-application.

Thus, for the application classes points store, ticket, identification document and data memory differentiated access protection may be brought about by subdividing the VAS-data into four Elementary Files as illustrated in Fig. 6.

The four Elementary Files contain the following data or are protected in the following manner:

## Table 3 - Summary of Files

| Field | Contents | Typical Usage | Access protection |
|---|---|---|---|
| EF_KEY | Contains the key $K_{LVASP}$, $K_{RVASP}$ which is known only to the VAS-provider (remark: for each VAS-application and optionally for each card the VAS-provider provides a single pair of keys) | A VAS-provider must authenticate himself via $K_{LVASP}$, prior to being permitted to write VAS-data into EF_INFO, EF_INTERNAL and EF_VALUE, respectively before he is permitted to read data from EF_INTERNAL.<br><br>A terminal must authenticate itself by means of $K_{RVASP}$ before being permitted to read VAS-data from EF_INFO, EF_VALUE | |
| EF_INFO | Non-internal information by way of the VAS-application | • Ticket information<br>• Bonus programme information<br>• Identity document information<br>• Parking ticket information | The contents of these data units can only be written by the VAS-provider (external authentication by knowledge of $K_{LVASP}$). Reading should only be possible at terminals having access to $K_{RVASP}$ or $K_{SO}$, or if the cardholder enters a correct PIN. (The PIN protection may be deactivated by the cardholder) |
| EF_INTERNAL | Internal data of the VAS-application | Internal counters, statements of account, tax information, additional keys for:<br>• Bonus programmes<br>• Discount scales<br>• Hidden identification features<br>• Codes | The contents of these data units can only be written and read by the VAS-provider. Access protection is based on external authentication by knowledge of $K_{LVASP}$ |
| EF_VALUE | Value units of VAS-application | This data unit contains accountable values of a VAS-provider application.<br><br>• Statement of account<br>• Residual value public transport ticket<br>• Credit<br>• User counter | Only the VAS-provider can write these data explicitly (external authentication by knowledge of $K_{LVASP}$). Implicitly the value can be reduced by the command "cancel" by a partner who has been entitled to perform the function by the VAS-provider (by signature of the operation cancelling with $K_{DEC}$). Reading as for EF_INFO. |

This structure of Elementary Files (EF) supports the same differentiated rights of access for all application classes.

Bearing in mind that application classes are now combined in a single implementation class which lays down the number and size of the Elementary Files, waste of storage capacity due to different space requirements for applications may be minimised. The application classes points storage and ticket require the same resources EF_KEY, EF_INFO, EF_INTERNAL, EF_VALUE and are therefore combined in an implementation class "DF_PT". For the application classes identity document and data memory the Elementary Files EF_KEY and EF_INFO suffice and are therefore combined in the implementation class "DF_AD". Considered in terms of numbers: There are p objects of the implementation class DF_PT and a objects of the implementation class DF_AD in the VAS-applications of the application classes point storage and ticket, respectively identification document and data storage can be loaded.

Specifically for the application class voucher, which should provide for public reading access for all VAS-participants for purposes of joint data interchange, the implementation class transfer field is used. Writing access is exclusively possible indirectly by the application of the TRANSFER command which presupposes a signature with a correct $K_{DEC}$. This implementation class consists of exactly one entry in Elementary File EF_TRANSFER belonging to the global data of the VAS-container. The objects of this class are records in this file. We refer to this implementation class also as "EF_TRANSFER".

The implementation classes illustrated in Fig. 7 exist within the VAS-container. These implementation classes form the storage model of the VAS-container.

The storage model may be made available in two manners. The nature depends on the card platform:

- *Fixed partitioning of the storage region for the VAS-container:*

For the implementation classes DF_PT and DF_AD a maximum number p or a, as the case may be, of objects is fixed by the issuer and is loaded by the issuer onto the card using CREATE FILE commands. The objects are denoted as DF_PT and DF_AD respectively. VAS-applications may be loaded later on into free objects, i.e. not occupied by other VAS-applications. For loading or deleting VAS-applications UPDATE RECORD commands only are then necessary.

Thus the issuer fixes the number of possible objects of the two implementation classes according to his own requirements; these may, however, differ from the actual VAS-user pattern of the cardholder. The subdivision is maintained over the entire life cycle of the card. A VAS-application is loaded into an object belonging to a suitable implementation class. Thus, for example, a VAS-application representing an identity document, may also be accommodated in an object of the implementation class DF_PT, if no (more) objects of the implementation class DF_AD are available. The assignment of a VAS-application to an object takes place by entering a tripel (PIX of the VAS-application, FID of the loaded object, clear text name of the application) into the global database EF_DIR of the VAS-container. The removal of the application corresponds to the removal of this tripel from EF_DIR and the destructive reading (by UPDATE RECORD commands) of the memory loaded with the application.

This modification is used with card platforms which do not support dynamic generating or deleting of DF/EF structures.

- *Dynamic setting up of objects of the implementation class:*

For each VAS-application to be loaded the files are set up by CREATE FILE commands as required for the underlying implementation class. When deleting the VAS-application, the DF/EF occupied by it, is removed entirely from the card and the storage space is freed. The maximum number of objects of implementation classes is not here explicitly determined by the issuer and depends only on the storage capacity of the card available.

This modification presupposes a dynamic database administration by the card operating system.

In the next following working example we will start off from the first modification, because the second one makes higher demands on the available card platform. However, if a dynamic memory administration is available and it has been assured that more storage capacity can be saved by the dynamic setting up of objects than corresponds to the administration costs, the existing concept may be adopted and offers more flexibility to the cardholder.

In the following the data structures and commands are presented by means of which the VAS-container and the functionality of the VAS-client card can be practised as compared with other system components. Moreover, VAS-operations are laid down for typical business situations which describe the respective interaction between the VAS-container and terminal.

The following preconditions apply to the implementation:

- Each card, regardless of the platform, makes available to the dealer terminal the same data and command interface. Accordingly, the required commands are clearly described in their codification.

- Service terminals are able to recognise the card platform. The functionality can therefore be attained by specific commands of the platform. Accordingly, these transactions are written partly informally. The manner in which this function is provided, is left to the card manufacturer.

In Fig. 7 a variety of implementation classes of the VAS-container is schematically illustrated in this context. Fig. 8 shows schematically the data structure of the VAS-container, Fig. 9 shows schematically the data structure of the implementation class DF_PT, Fig. 10 shows the data structure of the implementation class DF_AD.

Access to the files of the VAS-container is restricted explicitly by the following access conditions (AC):

Table 4 - Access conditions

| Database | Admin | Reader access | Writing access |
|---|---|---|---|
| DF_VAS | $K_{SO}$ | NEV | NEV |
| EF_ID | $K_{SO}$ | ALW | $K_{SO}$ |
| EF_DIR | $K_{SO}$ | ALW | $K_{SO}$ |
| global KEYs | $K_{SO}$ | NEV | $K_{SO}$ |
| PIN | $K_{SO}$ | NEV | $K_{SO}$ |
| EF_VERSION | $K_{SO}$ | ALW | $K_{SO}$ |
| | | | |
| EF_SEQ | $K_{SO}$ | ALW | $K_{SO}$ |
| EF_TRANSFER | $K_{SO}$ | ALW | $K_{SO}$ |
| | | | |
| DF_X (X=$PT_1$,...,$PT_D$,$AD_1$,... $AD_a$) | $K_{SO}$ | NEV | NEV |
| EF_KEY | $K_{SO}$ | NEV | $K_{SO}$ |
| EF_INFO | $K_{SO}$ | PIN or $K_{RVASP}$ or $K_{SO}$ | $K_{LVASP}$ |
| EF_INTERNAL | $K_{SO}$ | $K_{LVASP}$ | $K_{LVASP}$ |
| EF_VALUE | $K_{SO}$ | PIN or $K_{RVASP}$ or $K_{SO}$ | $K_{LVASP}$ |

In this context the AC have the following meaning:

- ALW (Always) = Access of the command to the database is always permitted.
- NEV (Never) = Access of the command to the database is never permitted.
- $K_{SO}$ = Prior to access external authentication of the system operator by way of the key $K_{SO}$ must take place.
- $K_{LVASP}$ = Prior to access external authentication of the VAS-provider by way of the key $K_{LVASP}$ must take place.
- $K_{RVASP}$ = Prior to access external authentication of a terminal authorised by the VAS-provider must take place by way of the key $K_{RVASP}$.

- PIN = Prior to access the correct PIN must be entered by the cardholder and be transmitted in clear to the card by the command VERIFY.

- PIN or $K_{RVASP}$ or $K_{SO}$ = Prior to access either the correct PIN must be entered by the cardholder or an external authentication by the VAS-provider using $K_{RVASP}$ or by the system operator using $K_{SO}$ must take place.

In this context it should be noted that the Or-linking of access rights as denoted above, in the card operating system is, as a rule, not provided. Where applied, this might involve special implementation costs (alternative: special READ command with implicitly fixed security attribute).

Data fields within the records of Elementary Files are differentiated according to the following formats: ASCII, Binary, BCD, Date, Format string.

Data elements of the type "format string" contain VAS-data in packaged presentation which can be displayed to the cardholder at the terminal.

For optimal data storage utilisation clear text and binary data are mixed and rendered displayable by way of formatting macros.

All Elementary Files (EF) of the VAS-container are defined according to ISO 7816-4 as *linear* formatted EF databases with records of *constant* length (*linear fixed record files)*.

The Elementary File EF_ID of DF_VAS consists of a record which contains the VAS-container ID.

The Elementary File EF_DIR of the DF_VAS consists of $nr_{DIR}$ records. For each VAS-application loaded into the VAS-container its proprietary identifier extension (PIX) and its physical storage site (FID of the DF_X into which the application has been loaded) is fixed in a record of the EF_DIR. The PIX identifies, e.g. as a code

number, the application and the service provider to which the application has been assigned.

The entries in EF_DIR are dynamically set up at the service terminal of the system operator. Records without entry are set up with an empty TLV object '61'.

When loading a VAS-application a free storage region for the appropriate implementation class is sought, certain VAS-data are there entered and finally a new record is generated in EF_DIR.

When deleting an application, an empty TLV object must accordingly be written into EF_DIR.

The Elementary File EF_VERSION of DF_VAS consists of a record which contains a version number of the VAS-container.

The version number may be utilised by the terminal in order to differentiate between different VAS-container modifications and/or different software versions.

The Elementary File EF_SEQ of DF_VAS consists of a record which contains the number of the next transfer field entry.

The sequence number is read out by the supplementary command TRANSFER. This command then generates in the transfer field EF_TRANSFER a record into which *inter alia* the sequence number from EF_SEQ is transferred. Jointly with the command TAKE which assures that each record from the transfer field is withdrawn once only and which records this withdrawal by signature via the sequence number, the once only withdrawal of (original) vouchers or receipts can be assured.

In the following, the transfer storage is to be dealt with in more detail.

The transfer storage is set up within the VAS-container and comprises $nr_{EF_TRANSFER}$ records. Entries in the records of these files contain transfer data generated by the

TRANSFER command. The data interchanged between VAS-applications as well as the storage of VAS-data of the class Voucher is performed by way of the transfer storage.

The transfer storage can be read without restrictions, however, the writing access is possible only by way of the VAS-specific command TRANSFER and TAKE.

Loading the data fields by the TRANSFER command proceeds by a 'last-recently-used' algorithm in the card. The oldest record in the database can be determined by searching for the lowest value in the first 2 bytes of the record.

Data fields may be marked by the command "TAKE" in the transfer storage as withdrawn and/or removed. In each case the deleting of data in the transfer storage takes place by overwriting with new data.

Each record in the transfer storage contains, for example, an expiry date, the terminal-ID, the PIX, the sequence number and optionally further data.

Each Elementary File EF_INFO within lists DF_X (where X = $PT_1$, ..., $Pt_p$, $AD_1$, ..., $Ad_a$) comprises a record containing the expiry date as well as general VAS-data of a VAS-application. For example, the nature of a ticket (single or multiple ticket) or the boarding locality may be entered here. EF_INFO must, however, contain at least one clear text name of the application which can be read for the operation *viewing VAS-applications*. Sensible data must first be protected by the VAS-provider by suitable external key algorithms.

This EF is readable if an external authentication with $K_{SO}$ or $K_{RVASP}$ takes place or where the cardholder, in the case of an activated PIN-protection enters a correct PIN.

Each Elementary File EF_INTERNAL within records DF_X (where X = $PT_1$, ..., $PT_p$, $AD_1$, ..., $AD_a$) consists of a record which may contain internal VAS-data of the VAS-

provider concerning its loaded VAS-application. Neither the cardholder nor any other VAS-provider can read these internal data.

Each Elementary File EF_VALUE within records DF_X (where X = $PT_1$, ..., $PT_p$, $AD_1$, ..., $AD_a$) comprises a record which may contain an integer value field of the VAS-data. This EF can be read if an external authentication with $K_{SO}$ or $K_{RVASP}$ takes place or the cardholder in the event of an activated PIN-protection enters a correct PIN or a correct password.

The following key fields are used by the VAS-container or by the VAS-application. In the following we will start from a pure DES-codification. I.e. all keys of the VAS-container are DES-keys and are coded in 8 bytes (including parity bits).

Within the VAS-container and within the VAS-applications the following keys referred to by their *KID* (Key Identifier) may be referenced:

Table 5 - Keys VAS-container

| KEY | KID |
|---|---|
| $K_{SO}$ | '00' |
| $K_{AUT}$ | '01' |
| $K_{SIG\ VASC}$ | '02' |
| $KGK_{DEC}$ | '03' |

Each key is coupled to a faulty operation counter. This registers failed authentication attempts with this key and blocks a further use once a limit entered for this counter has been attained.

Within the VAS-application the following keys can be referenced by their KID.

Table 6 - Keys VAS-application

| KEY | KID |
|---|---|
| $K_{LVASP}$ | '04' |
| $K_{RVASP}$ | '05' |

Each key is connected to a faulty operation counter. This registers failed authentication attempts with this key and blocks any further use once a limit entered for this counter has been reached.

The VAS-container contains a personal identification number or password (PIN). This is used for identification of the cardholder by way of the VERIFY command. The PIN is connected to a faulty operation counter which registers each false input and which on attainment of a limit blocks the PIN-comparison. This blockage can be turned back by the system operator using suitable administration commands. The faulty operation counter is turned back once the correct PIN has been entered.

The following parameters exist for the VAS-container which may be selected by the card issuer in accordance with space available on a card platform and his individual wishes:

- p Maximum number of objects of the implementation class DF_PT
  = maximum number of VAS-applications of the application classes Points Storage and Ticket which can be loaded into a VAS-container simultaneously;

- a Maximum number of objects of implementation class DF_AD
  = maximum number of VAS-applications of the application classes Identity Document and Data Storage which can be loaded simultaneously into a VAS-container;

- $nr_{DIR}$ Maximum overall number of objects: $nr_{DIR} = p + a$
  The number of records in EF_DIR is $nr_{DIR}$

- $nr_{EF_TRANSFER}$ number of records of the EF_TRANSFER

The storage requirements for the data of the described Elementary Files is as follows:

| | | *Bytes* |
|---|---|---|
| Global data of the VAS-container: | EF_ID | 4 |
| | EF_DIR | 9*(p+a) |
| | EF_VERSION | 1 |
| | EF_SEC | 2 |
| | Global Keys, Pin | 64+2 |
| Transfer field: | EF_TRANSFER | $48 * nr_{EF_TRANSFER}$ |
| Proprietary Data of the VAS-provider: | EF_KEY | (p+a) * 32 |
| | EF_INFOR | (p+a) * 62 |
| | EF_INTERNAL | p * 10 |
| | EF_VALUE | p * 3 |

If we select, e.g. for the parameters p, a and $nr_{EF_TRANSFER}$ the following values, then the following minimum storage requirements for the VAS-container (without storage requirements for supplementary commands) arise:

| Parameters | Storage requirements |
|---|---|
| p = 8, a = 3, $nr_{EF\text{-}TRANSFER}$ = 15 | 2030 bytes |
| p = 10, a = 5, $nr_{EF_TRANSFER}$ = 20 | 2758 bytes |

The maximum storage requirement is probably higher than the minimum storage requirement by about 10%.

In the following the commands of the chip card according to the invention will be dealt with in more detail.

The READ RECORD command is used for reading out data from lineary Elementary Files. The card in its reply supplies the contents of the records. The EF is referenced by the Short File Identifier (SFI).

The status code '9000' signals a successful performance of the command; any different code is considered an error.

The UPDATE RECORD command is used in order to enter data into the records of a linear EF. The command message contains the reference to the EF, the record and the data.

The response of the card contains the status code. The status code '9000' signals the successful conclusion of the command. Other status codes indicate an error.

The GET CHALLENGE command requires from the card a random number. The random number is used in conjunction with the dynamic authentication in EXTERNAL AUTHENTICATE.

The response message of the card contains a random number, 8 bytes long, and the status code. The status code '9000' indicates the successful performance of the command. Any different status codes indicate an error.

The command EXTERNAL AUTHENTICATE permits the authentication of a terminal against the card. The EXTERNAL AUTHENTICATE is used within the context of VAS-applications for authentication of the system operator and the VAS-provider. EXTERNAL AUTHENTICATE transfers a cryptogram into the card which prior to this has to be generated by the terminal by encodification of a random number. The card compares the cryptogram with a reference value which it has itself calculated using the same procedure. If both values are equal, the card notes internally that the access condition authentication for this key has taken place. If the comparison should turn out negatively, the card will issue the status "non authorised" and decrements an internal faulty operation counter. Once this counter reaches the value 0, any further performance of the EXTERNAL AUTHENTICATE is blocked with the status "authentication blocked".

The response message of the card contains the status code. The successful performance of the command and the authentication of the terminal in relation to the card is indicated by the status code '9000'. Any different status codes indicate an error.

The command INTERNAL AUTHENTICATE is used in order to check the authenticity of the VAS-container by a terminal. The card for this purpose calculates a cryptogram by way of reference data derived from the terminal. The terminal in its turn forms the cryptogram and compares it with the value of the card. In the event of equality the terminal can assume the authenticity of the VAS-container.

The response of the card contains the cryptogram and the status code for the execution of the command. The successful execution of the command is indicated by the status code '9000'. Any different status codes indicate an error.

The VERIFY command is used for verifying the cardholder PIN. The command transfers the PIN data uncodified to the card where a comparison with a stored reference value is performed. If the entered and the stored value are identical, the access conditions "PIN" are considered complied with.

The response message of the card contains the status code. The status code '9000' indicates a successful execution of the command. Other status codes indicate an error.

The TRANSFER command generates an entry in the transfer storage. Three operating modes are defined for the command:

1. Generation of an entry in the transfer field by reduction of values in the EF_VALUE field of the application of the class Ticket or Point Storage.
2. Generation of an entry in the transfer field by creation of a receipt in application of the class Identity Document.
3. Generation of an entry in the transfer field by generating a voucher in applications of the class Voucher.

The operating mode is automatically selected by the card: If the command is executed within a selected application-DF, the presence of an EF_VALUE is first checked for. If the EF_VALUE is present, the card executes the command in mode

1, alternatively in mode 2. If within the VAS-container no application-DF has been selected, mode 3 is used.

The terminal, when calling for the TRANSFER command, furnishes the card with the following data:

- Current date
- Expiry date for the entry in the transfer field
- Identification for the terminal which generates this entry
- User data for the transfer field
- PIX of the VAS-application (mode 3 only)
- Number of units to be subtracted (mode 1 only)
- MAC concerning the abovementioned data, the sequence number and the VAS-container number.

The card, once the TRANSFER command has been called up, performs the following sequence:

1. Searching for a free entry in the transfer storage. (The following list provides in reverse order the priority in which existing entries are to be overwritten: "Entry marked as removed", "entry marked as withdrawn" and "expiry date reached", "expiry date reached").
2. Attaching the PIX to the data of the terminal in modes 1 and 2.
3. Attaching the sequence number to the data from step 2.
4. Attaching the VAS-container ID to the data from step 3.
5. Deriving the $KGK_{DEC,PIX}$ by encoding the PIX under the $KGK_{DEC}$.
6. Deriving the $K_{DEC}$ by encoding the terminal ID under $KGK_{DEC, PIX}$.
7. Generating the MAC by way of the data of step 4.
8. Comparison of the MAC from step 7 with the MAC of the terminal. If the values are different, the card interrupts the function and reduces the faulty operating counter for the $KGK_{DEC}$.
9. For mode 1: Testing the value field EF_VALUE in the register into which the application had been loaded. If insufficient units are present in this field, the

card interrupts the function at this point. Otherwise the value field in the application is reduced by this amount.

10. Assembly of the command message.
11. Incrementation of the contents of EF_SEQ by 1.

The command message contains, for example, fields including an expiry date, the terminal ID, the transaction data, a field for the operating mode (contains, e.g. in mode 3, the PIX), as well as a cryptogram.

The cryptogram is calculated using the key $K_{DEC}$, the data formed by way of the MAC, for example, embracing the transaction data and the terminal-ID.

The response message of the TRANSFER command includes, when successful, a data field 8 bytes long and the status code '9000' which is 2 bytes long. Reply messages with a status code differing from '9000' are interpreted as error codes. The data field of the reply message contains *in the error free situation*, (i.e. in particular the cryptogram of the command message was correct) the cryptogram encoded with $K_{DEC}$ of the command message. In this manner (instead of an internal authentication using $K_{aut}$) an authentication is performed implicitly by the VAS-container.

The command TAKE serves for the withdrawal of objects from the implementation class EF_TRANSFER. Technically, the execution of the command TAKE means the reading out of a record to be denoted from the storage EF_TRANSFER, the record being retained in the storage for so long until the storage space is required for a new entry, the set of data being marked as *withdrawn*. Considered technically, anybody can use the command TAKE in order to withdraw a set of data, however, according to the R&R only a VAS-provider for whom the set of data was intended, should do this.

For the removal procedure the following may be assumed. The VAS-provider who wishes to remove a voucher or a receipt, searches the transfer storage for a suitable

set of data (e.g. by the command SEEK or by explicit reading of each record). The set of data will in any event be read and its contents may be checked.

A display of the set of data in the response is therefore not necessary. It may furthermore be assumed that the number of the record is known.

The command TAKE provides for the following modes:

- Removal with simultaneous annotation that the data have been invalidated
- Removal without the aforesaid annotation. The data remain valid.

The command message contains fields with terminal-ID, PIX of the application and a random number generated by the terminal.

The PIX in the command denotes the application which removes the data. It may differ from the PIX of the set of data which is being removed. It serves exclusively for the derivation of $K_{DEC}$ of the dealer terminal which effects the removal.

The response message of the card contains a cryptogram $C_1$ with $K_{SIG\text{-}VASC}$ concerning the data of the records of the transfer field reported in the command message and the VAS-container ID, a cryptogram $C_2$ with $K_{DEC}$ of the terminal effecting the removal via $C_1$ and the random number taken from the command message. The response message, in addition, contains the status code. The key $K_{DEC}$ is derived as described further above. Authenticity is implicitly proved by the VAS-container by virtue of the cryptogram via $K_{DEC}$. Proof of authenticity and uniqueness is calculated by means of the cryptogram C (since C is formed by way of the sequence number, removal bit of the transfer field record and the VAS-container ID) which can be verified by the system operator.

The status code '9000' indicates the successful execution of the command. Different status codes are interpreted as errors.

It is to be assumed that the SO requests *one* $AID_{VAS}$ according to IDO/IEC 7816-5. More specifically: He requests an $RID_{VAS}$ , 5 bytes long for the VAS-system.

The $AID_{VAS}$ of the list DF_VAS is to read: $AID_{VAS} = RID_{VAS}.PIX_{DF_VAS}$.

For each VAS-application A a $Pix_A$, 3 bytes long, is issued in accordance with R&R in order to identify the former unambiguously within the VAS-container via $AID_A = RID_{VAS}.PIX_A$. After the selection of DF_VAS a list in which the VAS-application A is contained, may be selected using SELECT FILE < $PIX_A$ >.

The UPDATE KEY (KID, K) denotes a command which depends on the card platform by which a key is replaced by the new value K using the Key Identifier ID.

Before a VAS-application out of one of the implementation classes DF_PT or DF_AD (corresponding to the application classes Point Storage, Ticket, Identification Document or Database) can be utilised by the cardholder at the dealer terminal, it must be loaded into the VAS-container at a service terminal of the appropriate VAS-provider. In principle it is also possible that a card issuer instructed by a VAS-provider and an SO loads one or more VAS-applications into the VAS-container already when the VAS-container is installed. Such loading process constitutes a special case of what is here described.

Procedure of loading of a VAS-application:

1. The cardholder inserts a VAS-card into a service terminal.
2. The service terminal checks whether a VAS-container is present:

   - SELECT FILE < $AID_{VAS}$ > (error report if VAS-container not selectable)
   - READ RECORD < SFI of EF_ID, 0 > (display of VAS-container number).

   Optionally, the validity of the VAS-container is checked. For this the service terminal requires an internal authentication of the VAS-container:

   - INTERNAL AUTHENTICATE < random number, KID of $K_{AUT}$ >

The service terminal checks the response and in the event of an error (with error display) terminates the procedure.

3. The service terminal offers to the cardholder a selection of several options. One thereof reads *loading VAS-application*. This is selected by the cardholder. All VAS-applications which can be loaded at the service terminal are now displayed to the cardholder and a selection is awaited. For this purpose the operation *viewing VAS-applications* is activated. The cardholder selects a VAS-application A of the implementation class DF_PT or DF_AD.

4. The service terminal inspects the VAS-container by way of the operation *selection of a VAS-application*, as to whether the selected VAS-application having $PIX_A$ has already been loaded into the card. In the affirmative, an error signal is displayed. If not, a check is made whether an object of the implementation class suitable for A is still available in the VAS-container. This proceeds by searching for a free record in EF_DIR (e.g. using the command SEEK). If nothing is available, an error signal is displayed. If a record is free, this contains an $FID_{DF_X}$ of a DF_X into which no VAS-application has been loaded.

5. The next following free object of the implementation class suitable for A is loaded with the VAS-application A. For this purpose the service terminal first requests off-line from the VAS-provider (e.g. through a VAS-provider SAM) two keys $K_{LVASP}$ and $K_{RVASP}$ and assigns these to the new VAS-application:

   - SELECT FILE < $FID_{DF_X}$ >
   - GET CHALLENGE
   - EXTERNAL AUTHENTICATE < $K_{SO}$ (random number), KID of $K_{SO}$ >
   - *UPDATE KEY < KID of $K_{LVASP}$, $K_{LVASP}$ >*
   - *UPDATE KEY < KID of $K_{RVASP}$, $K_{RVASP}$ >*
   - GET CHALLENGE
   - EXTERNAL AUTHENTICATE < $K_{LVASP}$ (random number), KID of $K_{LVASP}$ >

- UPDATE RECORD < SFI of EF_INFO of DF_X, data >
- UPDATE RECORD < SFI of EF_INTERNAL of DF_X, data >
- Optional (initialling): UPDATE RECORD < SFI of EF_VALUE of DF_X, data >

6. After the successful entry in the EFs the service terminal performs the operation *enter VAS-application* < $PIX_A$, $FID_{DF_X}$ >, which connects the DF_X to the $PIX_A$ and thereby permits SELECT FILE by means of the $PIX_A$ (after the prior selection via SELECT FILE $AID_{VAS}$).

The mechanism made available by way of the transfer storage may, for example, be utilised by VAS-providers who wish to perform intra- or interservices without proprietary DF-structures. A cardholder, in particular prior to using such VAS-application, need not then first load this at a service terminal. On the contrary, he may issue to himself at the dealer terminal directly a voucher or a receipt and have this redeemed at a different terminal (by the operation *removing*) or simply present the voucher or receipt (by reading). *Loading* of a VAS-application of the implementation class EF_TRANSFER is therefore realised implicitly by the *entering* of VAS-data or is caused by such operation. In this context, reference is made to the description of the purchase of the implementation class EF_TRANSFER further below.

In the following the course of the operation of entering a VAS-application is described.

If a VAS-application has been loaded into the VAS-container, a terminal will not know the physical locality at which DF_X wherein X = $PT_1$, ..., $PT_p$, $AD_1$, ..., $Ad_a$ the VAS-application has been loaded or whether it has been loaded at all. From the aspect of the card manufacturer two possible implementation modes are possible which are to be checked separately; in this context, consistency with the ZKA-standard must be particularly observed.

The first situation: Access to EF_DIR is possible

The following preconditions must be met:

- An EF_DIR (e.g. under DF_VAS), in which AIDs are associated with the FIDs of the DF_X in which VAS-applications are currently present, exists in standard manner in the card platform.

- This EF_DIR is readable by anybody (AC of READ RECORD: ALW).

- If a VAS-application A with $PIX_A$ is loaded by the system operator into a free (unused) DF_X, i.e. an entry is made into the existing Elementary Files DF_X (no CREATE FILE!), it should then be possible to extend the database EF_DIR by the entry $PIX_A$ by means of an UPDATE RECORD which refers to this DF_X by way of $FID_X$. The AC of UPDATE RECORD should therefore enforce an external authentication by means of the key $K_{SO}$.

- If the command SELECT FILE < $PIX_A$ > after the prior selection of DF_VAS is transmitted to the card, the DF_X into which a VAS-application A has been loaded, must be directly selectable.

- If an VAS-application A, loaded into DF_X and its subsidiary Elementary Files, is to be deleted at a service terminal at the request of the cardholder, the corresponding databases are not deleted by way of DELETE FILE, but the entered data are simple written over with dummy values. Thereafter it should be possible to delete from EF_DIR the entry $PIX_A$ (more specifically the pair $PIX_A$ and the reference to DF_X) (e.g. by UPDATE RECORD with prior external authentication via the key $K_{SO}$)

- Since the number of DF_X is fixed, the number of records of EF_DIR is known.

If this approach is realisable, the following consequence results:

- A direct selection of a VAS-application using SELECT FILE $PIX_A$ is possible following a selection of DF_VAS.

The second situation: Access to EF_DIR is not possible

In the event that with the particular card platform no EF_DIR is available or no reading or writing access to this EF_DIR - as described in the previous paragraph - is possible, provision is to be made under DF_VAS for a database EF_VASDIR for a connection to be made of $PIX_A$ loaded VAS-application to their physical storage locality in a DF_X, brought about by the system operator by explicit UPDATE RECORD commands (after prior external authentication via $K_{SO}$). Reading and deleting of records from EF_VASDIR should be possible as previously described.

The performance of the operation entry VAS-application proceeds as follows:

A VAS-application A including $PIX_A$ was previously loaded into a free DF_X with $FID_{DF_X}$. The number of the record including $FID_{DF_X}$ was noted by the service terminal.

1. SELECT FILE < $AID_{VAS}$ > (error display if VAS-container not selectable)
2. GET CHALLENGE
3. EXTERNAL AUTHENTICATE < $K_{SO}$ (random number), KID of $K_{SO}$ >
4. UPDATE RECORD < SFI of EF_DIR of DF_VAS, number record with $FID_{DF_X}$, $PIX_A$, $FID_{DF_X}$>

VAS-applications of the implementation classes DF_PT or DF_AD can only be deleted at the request of the cardholder at the service terminal (being under the control of the system operator), VAS-applications of the implementation class EF_TRANSFER can be deleted anywhere by anybody. When deleting it is necessary to differentiate between VAS-applications of the implementation classes DF_PT and DF_AD, respectively the implementation class EF_TRANSFER.

Deleting of such VAS-application for the implementation class DF_PT or D_AD proceeds as follows:

1. The cardholder inserts a VAS-card into a service terminal.
2. The service terminal checks whether a VAS-container is present:
   - SELECT FILE < $AID_{VAS}$ > (error display if VAS-container not selectable)
   - READ RECORD < EF_ID > (display of VAS-container ID)
3. The service terminal provides the cardholder with a selection of several options. One thereof reads *delete VAS-application*. This is selected by the cardholder. All VAS-applications of all implementation classes loaded into the VAS-container are now displayed to the cardholder and his selection is awaited. For this purpose, the operation *viewing VAS-applications* is activated. The cardholder selects a VAS-application A of the implementation class DF_PT or DF_AD by means of $AID_A$. The object into which the VAS-application has been loaded is assumed to be denoted DF_A.
4. After the selection of DF_A, the service terminal authenticates itself:
   - SELECT FILE < $PIX_A$ >
   - GET CHALLENGE
   - EXTERNAL AUTHENTICATE < $K_{SO}$ (random number), KID of $K_{SO}$ >
5. The content of the files of DF_A is now deleted ($K_{LVASP}$ is required for EF_KEY, EF_INTERNAL and EF_VALUE, for which reason the former is first deleted by the system operator):
   - *UPDATE KEY < KID of $K_{LVASP}$, "00...00">*
   - *UPDATE KEY < KID of $K_{RVASP}$, "00...00">*
   - GET CHALLENGE
   - EXTERNAL AUTHENTICATE < $K_{LVASP}$ (random number), KID of $K_{LVASP}$ >
   - UPDATE RECORD < SFI of EF_INFO of DF_A, "00...00" >
   - UPDATE RECORD < SFI of EF_INTERNAL of DF_A, "00...00" >
   - UPDATE RECORD < SFI of EF_VALUE of DF_A, "00...00" >
6. After successful entries into the EFs the service terminal subsequently causes the entry of the VAS-application to be deleted from EF_DIR:

- SELECT FILE < $AID_{VAS}$ > (error display if VAS-container not selectable)
- UPDATE RECORD < SFI of EF_DIR of DF_VAS, number record with $FID_{DF_A}$, "00...00", $FID_{DF_A}$ >

The interlinkage of $PIX_A$ to DF_A is undone in that SELECT FILE with $PIX_A$ is no longer possible.

Next follows the implementation class EF_TRANSFER:

A VAS-application of the implementation class EF_TRANSFER, in accordance with R&R, can only be deleted at the explicit request of a cardholder at a dealer terminal or service terminal (even though this might be technically feasible by either). Deleting an object from EF_TRANSFER becomes necessary in particular if the transfer storage has no more space for storing a new object (e.g. a voucher or a receipt). Deleting proceeds always indirectly by way of the supplementary command TAKE. This command merely marks an object as having been removed. Thereafter it is freed to be overridden by a subsequent supplementary command TRANSFER.

Deletion of this VAS-application proceeds as follows:

1. The cardholder inserts a VAS-card into a terminal capable of displaying the content of EF_TRANSFER (special case of viewing operation VAS-applications).

2. The terminal checks whether a VAS-container is present:
   - SELECT FILE < $AID_{VAS}$ > (error display if VAS-container not selectable)
   - READ RECORD < SFI of EF_ID, 0 > (display of VAS-container ID)

3. The terminal makes available to the cardholder a selection of several options. One thereof reads *delete VAS-application*. This is selected by the cardholder. At least the VAS-applications of the implementation class EF_TRANSFER

are now displayed to the cardholder and a selection is expected. This is brought about by a special case of the operation *viewing VAS-applications*. The cardholder selects an object from the implementation class including a voucher or a receipt. This object has the record number A. The cardholder activates the selection.

4. The terminal marks the record with record number A as removed in that it transmits A, a random number calculated by it, its PIX and terminal ID, using the command TAKE:
   - TAKE < A, random number, PIX, terminal ID >

The record marked as having been removed is now available again for receiving a new voucher or a receipt.

The selection of a VAS-application proceeds as follows:

In the event that a VAS-application A of the implementation class DF_PT or DF_AD has been loaded into the VAS-container by the operation *loading VAS-application*, this can be selected by a terminal in two stages. Firstly, the VAS-container is selected and thereafter the VAS-application including its $PIX_A$:

1. SELECT FILE < $AID_{VAS}$ > (error display if VAS-container not selectable)
   A service terminal is able to check the authenticity of the VAS-container. For that purpose a service terminal demands an internal authentication of the VAS-container:

   - READ RECORD < SFI of EF_ID, 0 > (display of the VAS-container ID)
   - INTERNAL AUTHENTICATE < random number, KID of $K_{AUT}$ >

   The service terminal checks the response and interrupts the operation in the event of an error (with error display).

2. SELECT FILE < $PIX_A$ > (error display if VAS-application A has not been loaded into the VAS-container)

A dealer terminal (which has no $KGK_{AUT}$ available) may indirectly determine the authenticity of the VAS-container by an authenticity check of the VAS-application A. This is so because a VAS-application can only have been loaded at a service terminal which during the loading procedure has checked the authenticity of the VAS-container. The authenticity check of the VAS-application A can be taken back to the test at the dealer terminal whether the VAS-container contains the key $K_{LVASP}$ or $K_{RVASP}$ for the VAS-application A. This may be checked by the dealer terminal, e.g. as follows:

- INTERNAL AUTHENTICATE < random number, KID of $K_{RVASP}$ or $K_{LVASP}$ >

In order to test whether a VAS-application A is loaded into the VAS-container, it may be selected at random; based on the error display of the response message of SELECT FILE, the conclusion may be made that it is not present.

A selection of a VAS-application of the implementation class EF_TRANSFER results implicitly by the operation *viewing VAS-applications* and the storage of the data in the terminal. Since the terminal knows the record number of each indicated object from EF_TRANSFER, it is possible to select any desired object for further processing with reference to the record number.

The performance of viewing a VAS-application proceeds as follows:

The operation *viewing of VAS-applications* lists at a service terminal all VAS-applications of the implementation classes DF_PT, DF_AD and EF_TRANSFER. Only VAS-applications of the implementation class EF_TRANSFER can be displayed at a dealer terminal because of the absence of access protection therefor, and optionally also applications of the implementation class DF_PT or DF_AD for which the dealer terminal has reader rights (possession of $K_{RVASP}$).

Such a VAS-application proceeds as follows:

1. The cardholder inserts a VAS-card (chip card with valid VAS-container) into the terminal.

2. The service terminal tests whether a VAS-container is present:

   - SELECT FILE < $AID_{VAS}$ > (error display of VAS-container not selectable)
   - READ RECORD < SFI of EF_ID, 0 > (display of VAS-container ID)

Optionally the validity of the VAS-container is checked. For this purpose the service terminal demands an internal authentication of the VAS-container:

- INTERNAL AUTHENTICATE < random number, KID of $K_{AUT}$ >
  The service terminal tests the response and in the event of an error (with error display) interrupts the procedure.

3. The service terminal presents the cardholder with a selection of several options. One of these reads *viewing VAS-applications*. This is selected by the cardholder.

4. The service terminal authenticates itself:

   - GET CHALLENGE
   - EXTERNAL AUTHENTICATE < $K_{SO}$ (random number), KID of $K_{SO}$ >

5. For VAS-applications of the implementation classes DF_PT and DF_AD the individual VAS-applications are selected, controlled successively by the contents of EF_DIR and, after external authentication with the key $K_{SO}$ the contents of the individual EF_INFO (or part thereof according to R&R) as well as EF_VALUE are displayed:

- For i = 0, ..., $nr_{DIR}$ - 1
  - READ RECORD < SFI of EF_DIR, i >
    In the response message $PIX_A$ is displayed which is "00..00" if the corresponding DF is not loaded with a VAS-application. As an alternative to READ RECORD from EF_DIR it may also be possible to employ a SEEK command.
  - If $PIX_A$ does not equal "00..00"
    - SELECT FILE < $PIX_A$ >
    - READ RECORD < SFI of EF_INFO, 0 >
    - Display (optionally only in part) of the response message
    - READ RECORD < SFI of EF_VALUE, 0 >
    - SELECT FILE < $AID_{VAS}$ >

6. For VAS-applications f the implementation class EF_TRANSFER the contents (or a portion thereof according to R&R) of the EF_TRANSFER of each record is displayed:

   - SELECT FILE < $AID_{VAS}$ >
   - If i = 0, ..., $nr_{EF_TRANSFER}$ - 1
     - READ RECORD < SFI of EF_TRANSFER, i >
       (the response message displays the contents of the i-th records of EF_TRANSFER)
     - If the contents are not empty, the contents is displayed in interpreted form (e.g. taken out/expired).

Viewing of the applications of the implementation class DF_PT or DF_AD for which the dealer terminal has viewing rights (possession of $K_{RVASP}$), proceeds as described - subject to two changes: For external authentication the key $K_{RVASP}$ is used instead of $K_{SO}$ which is available only to the system operator. If a dealer terminal does not have available $KGK_{AUT}$ and nevertheless wishes to check the authenticity of the VAS-applications, the dealer terminal instead of internal authentication may demand

the authentication (of at least one) VAS-application. This proceeds as described by the knowledge of the card of the corresponding $K_{RVASP}$ or $K_{LVASP}$ key.

For VAS-applications of the implementation class EF_TRANSFER the contents (or a portion thereof according to R&R) of EF_TRANSFER of each record are listed successively as previously described.

The operation *interpretation VAS-applications* proceed analogous thereto. For that purpose the terminal offers the cardholder an option *interpretation VAS-applications* for selection. In addition to the data which become displayed due to the operation *viewing*, it is possible to also display data from EF_INFO and EF_VALUE which require an interpretation by the VAS-provider (e.g. externally encoded data) and data from EF_INTERNAL (e.g. miles from previous years with Miles & More). This, however, is possible only for the VAS-applications for which a VAS-provider (at his own option) makes available at the terminal suitable keys. The key $K_{LVASP}$ (external authentication) is required for reading the EF_INTERNAL of a VAS-application. For externally encoded data within the Elementary Files EF_INFO, EF_INTERN and EF_VALUE as well as the transfer storage EF_TRANSFER the appropriate keys of the VAS-provider are required. The terminal programme can readily distinguish with the aid of the PIX of a selected VAS-application, for which applications keys are available for the interpretation of the data.

The operation *transfer of VAS-applications* denotes the transfer of all or selected VAS-operations of a source card onto a target card at a service terminal. This is subject to the precondition that in the VAS-container of the target card, no VAS-applications are loaded and that after the transfer all VAS-applications are deleted from the source card. Both may be attained by successive applications of the operation *delete a VAS-application*. Furthermore the authenticity of the VAS-card must be checked and the target card must provide adequate storage capacity. The operation transfer itself is based essentially on an extension of the operation *viewing of VAS-applications* for which additionally the data from EF_INTERNAL and the keys $K_{LVASP}$ and $K_{RVASP}$ are read and on a repeated application of the operation *loading a VAS-application*.

Jointly with the VAS-data the VAS-container ID as well is transferred to the target card in order for the VAS-applications of the target card to perform exactly as they did on the source card. The reasons therefor is that keys derived from a VAS-provider are supported by the VAS-container ID, the keys, however, being copied without change. Moreover, a VAS-provider does not wish to change his accounting in the basic system because he normally identifies VAS-cards by way of the VAS-container ID. It is essential to ensure that during the transfer of the VAS-container ID this number which is unique for the system, is deleted from the source card.

In order to be able to specifically read the Elementary File EF_INTERNAL and the keys $K_{LVASP}$ and $K_{RVASP}$, the service terminal, after an external authentication by means of the key $K_{SO}$ (as in the operation *deleting a VAS-application)* first overwrites the key $K_{LVASP}$ and then after a renewed authentication with $K_{LVASP}$ overwrites the data from EF_INTERNAL.

In addition to the VAS-applications of the implementation classes DF_PT and DF_AD, the file EF_TRANSFER must be overwritten. For this purpose the operation *remove* is used successively to remove the objects of this implementation class which have not yet been marked as removed or expired, checks their signature by way of $K_{SIG_VASC}$ and performs the transfer into the EF_TRANSFER of the target card. On the target card, on the other hand, the objects are denoted as not removed, so that they will remain valid.

Finally, the global data of the VAS-container must be transferred. In particular, the sequence number of the target card must be adjusted to the value of the source card.

The procedure of entering VAS-data is described in what follows:

There are three possible cases of entering VAS-data at a dealer terminal which depend on the nature of the VAS-application.

Firstly, the purchase in the event of implementation class DF_PT or DF_AD

A VAS-provider is to enter data into a VAS-application A of the implementation class DF_PT or DF_AD.

The entering of the VAS-data proceeds as follows:

1. The cardholder inserts a VAS-card into a dealer terminal.

2. The dealer terminal checks whether a VAS-container is present:

   - SELECT FILE < $AID_{VAS}$ > (error display if VAS-container not selectable)
   - READ RECORD < SFI of EF_ID, 0 > (display of VAS-container ID)

3. The authenticity of the VAS-container is checked.

   If the dealer terminal itself is in possession of the master key $KGK_{AUT}$, it can demand an internal authentication of the VAS-container:

   - INTERNAL AUTHENTICATE < random number, KID of $K_{AUT}$ >

   A dealer terminal (which does not have available the $KGK_{AUT}$) may indirectly check the authenticity of the VAS-container by the authenticity check of the VAS-application A. The reasons is that a VAS-application can only have been loaded at a service terminal which during loading had checked the authenticity of the VAS-container. The authenticity check of the VAS-application A can be referred back to the test of the dealer terminal whether the VAS-container contains the key $K_{LVASP}$ or the key $K_{RVASP}$ for the VAS-application A. This may be checked by the dealer terminal, e.g. by:

   - SELECT FILE < $PIX_A$ > (error display if VAS-application A not loaded into the VAS-container)
   - INTERNAL AUTHENTICATE < random number, KID of $K_{RVASP}$ or $K_{LVASP}$ >

The dealer terminal checks the response and in the event of error (with error display) breaks off the procedure.

4. The dealer terminal selects the VAS-application A, authenticates itself and describes the Elementary Files which are necessary for performing the transaction:

   - SELECT FILE < $PIX_A$ > (obviated if already performed in step 3)
   - GET CHALLENGE
   - EXTERNAL AUTHENTICATE < $K_{LVASP}$ (random number), KID of $K_{LVASP}$ >
   - optional: UPDATE RECORD < SFI of EF_INFO of DF_X, data >
   - optional: UPDATE RECORD < SFI of EF_INTERNAL of DF_X, data >
   - optional: UPDATE RECORD < SFI of EF_VALUE of DF_X, data >

Now follows the purchase in the case of implementation class EF_TRANSFER

Specifically for VAS-applications of the implementation class EF_TRANSFER (application class Voucher) a VAS-provider need not occupy a proprietary file structure DF_X for his application. The result is that a cardholder need not, prior to using a VAS-application voucher, go to a service terminal in order to load a VAS-application. He may rather, directly at the dealer terminal, issue a voucher or a receipt and have these reimbursed at a different terminal (by the operation *removing*) or simply show the voucher or receipt (by reading). The entering of VAS-data accordingly results in an implicit loading of VAS-applications of the implementation class EF_TRANSFER. For this application class the implementation class EF_TRANSFER exists which is composed of individual objects of the type record. An entry in EF_TRANSFER is possible exclusively by the command *TRANSFER*. The dealer terminal must for that purpose be in possession of a valid cancellation key $K_{DEC}$ and must have available a $PIX_A$ of the VAS-application A.

The entering of VAS-data proceeds as follows:

1. The cardholder inserts a VAS-card into a dealer terminal.
2. The dealer terminal checks whether a VAS-container is present:

   - SELECT FILE < $AID_{VAS}$ > (error display if VAS-container not selectable)
   - READ RECORD < SFI of EF-ID, 0 > (display of VAS-container number)

3. The dealer terminal reads the sequence number which additionally enters the MAC from step 4:
   - READ RECORD < SFI of EF_SEQ, 0 > (display of sequence number)

4. The command TRANSFER causes a record to be entered in EF_TRANSFER:

   - TRANSFER < transaction date, expiry date, generator code, data, $PIX_A$, MAC with $K_{DEC}$ >

5. The dealer terminal can check by inspection of the response message of the TRANSFER command whether the VAS-container is genuine (i.e. in possession of the joint secret $K_{DEC}$). In this context reference is also made to the response message following the command TRANSFER, described further above.

Finally, the purchase of VAS-data by value cancellation

A VAS-provider may by a value cancellation step using the command TRANSFER in the transfer field EF_TRANSFER generate an entitlement (e.g. voucher or receipt) which can be further used by a different VAS-provider. Data are cancelled from the EF_VALUE or EF_INFO of the VAS-application A of the VAS-provider effecting the cancellation. The cardholder acquires the right in the form of an object in EF_TRANSFER.

The entry of VAS-data proceeds as follows:

1. The cardholder inserts a VAS-card into a dealer terminal.

2. The dealer terminal checks whether a VAS-container is present:

   - SELECT FILE < $AID_{VAS}$ > (error display if VAS-container not selectable)
   - READ RECORD < SFI of EF_ID, 0 > (display of VAS-container ID)

3. The dealer terminal reads the sequence number which, in addition, enters the MAC in the course of step 4:

   - READ RECORD < SFI of EF_SEQ, 0 > (display of sequence number)

4. The dealer terminal selects the VAS-application A:

   - SELECT FILE < $PIX_A$ > (error display, if VAS-application A has not been loaded into the VAS-container)

5. The dealer terminal uses the command TRANSFER for cancelling the data from the EF_VALUE or the EF_INFO as the case may be.

   - TRANSFER < data, MAC with $K_{DEC}$ >
     The composition of the command message of TRANSFER has already been described. The entitlement to perform the cancellation is obtained by the terminal if it can form a correct signature concerning the data by means of the $K_{DEC}$. This signature is checked by the VAS-container. If successful, a record is set up in EF_TRANSFER by the VAS-container and the sequence number is incremented.

6. The dealer terminal can check by checking the response message of the TRANSFER command whether the VAS-container is authentic (i.e. in possession of the common secret $K_{DEC}$).

We shall now deal with the procedure for cancelling VAS-data. There are two kinds of cancellation procedures:

On the one hand, VAS-data for VAS-applications of the implementation class DF_PT or DF_AD may be cancelled by the appropriate VAS-provider by the operation cancellation, i.e. purchase of VAS-data by cancellation. In that manner, a value is consumed, whereby a potential entitlement to a different value is created.

On the other hand, VAS-data may be withdrawn from the EF_TRANSFER once only by the supplementary command TAKE. In that instance, the entitlement is consumed, the data remain for possible further use (e.g. refunded ticket is still required for the return trip) in the transfer field, denoted as having been removed until overridden by other objects, when required.

Cancellation of VAS-data by the command TAKE proceeds as follows:

1. The cardholder inserts a VAS-card into a dealer terminal.

2. The dealer terminal checks whether a VAS-container is available:
   - SELECT FILE < $AID_{VAS}$ > (error display if VAS-container not selectable)
   - READ RECORD < SFI of EF_ID, 0 > (display of VAS-container ID)

3. The dealer terminal first displays from EF_TRANSFER the objects which are available using the special case of operation *viewing VAS-applications* in order to decide whether a required object is available. Alternatively, the command SEEK may be employed for seeking a sample. The terminal, if successful, knows the number i of the record searched for.

4. The terminal marks the record to show record number i in that it transmits i, a random number calculated by it, its PIX and terminal ID with the command TAKE:

- TAKE < i, random number, PIX, terminal ID >

The dealer terminal uses the command TAKE for reading the data from the EF_TRANSFER, simultaneously identifying the data as having been taken out. The execution of the command, in addition, brings about the generation of two different cryptograms $C_1$ and $C_2$.

The cryptogram $C_1$ is calculated by the VAS-container using the key $K_{SIG_VASC}$. In this manner the producer of the removed object signed with $C_1$ can obtain evidence of uniqueness and authenticity from the system operator. The uniqueness and authenticity of the transaction arises from the cryptogram of the VAS-provider from which the object initially originated (and which was checked by the card, see also TRANSFER), and from the maintenance of a sequence counter over the transactions as well as the cryptogram $C_1$ by the card during the withdrawal.

The cryptogram $C_2$ is calculated by the VAS-container using the key $KGK_{DEC}$, which is derived by the VAS-container from $KGK_{DEC}$, PIX and terminal ID. By having knowledge of the joint secret $K_{DEC}$, the VAS-container can directly verify to the terminal the authenticity of the VAS-container. Due to the fact that during the TRANSFER command a check is likewise made that only genuine vouchers or receipts are stored in an authentic VAS-container, the authenticity of the withdrawn object can be concluded. Since $C_2$ is formed indirectly via the sequence number, the withdrawal bit and the VAS-container ID, the VAS-container can even verify to the terminal the uniqueness of the object withdrawn.

Anybody has the right to withdraw using the command TAKE.

Moreover, at the service terminal the deactivation or activation respectively of a password or a PIN-protection for the reading of VAS-applications of the implementation classes DF_PT and DF_AD is possible. In addition, the PIN of the VAS-container can be altered by the cardholder having knowledge of the PIN and may be restored by the system operator by means of external authentication by $K_{SO}$.

Non-numerical symbols or symbol sequences of optional length can also be used as PIN or password.

REPLACED SHEETS SG
☐ Description : paged ... to paged ...
☑ Claim(s) : paged 69 to paged 79
☐ Drawing(s) : paged ... to paged ...
☐ Abstract : paged ... to paged ...

This Sheet Has Been Replaced

Patent Claims

1. Chip card for performing transactions in which monetary units or other value data representing non-monetary entitlements are transferred between the cardholder and at least one transaction partner (service provider) or are presented to the service provider for verification of entitlements, wherein the chip card includes a storage, in which data required for performing the transactions are stored and the chip card is characterised in that it comprises the following:

a means for loading one or more card applications (VAS-applications) onto the card which each permits the performance of transactions between the cardholder and one or more service providers.

2. Chip card according to claim 1, in which the means for loading comprise:

a data structure (DF_VAS, VAS-container) stored therein which includes:

a partial structure (DF_PT, DF_AD, VAS-application) into which the data required (VAS-data) for making possible the performance of a transaction between the cardholder and a service provider can be loaded;

a definition data set which contains information concerning the nature and/or structure of the data stored in the partial structure (VAS-application); and wherein the definition data set further comprises:

a denotation (container-ID) which identifies the data structure (VAS-container) and/or the chip card; and

at least one system key ($K_{SO}$) which secures the integrity of the definition data set and/or the data structure (VAS-container) against modifications.

3. Chip card according to claim 1 or 2, characterised in that it further comprises:

a transfer storage region (EF_TRANSFER) for accommodating data which in the performance of the transaction are to be interchanged or presented and which represent the monetary units or non-monetary entitlements and



* C L M S *

This Sheet Has Been Replaced

a means for writing data into the transfer storage by reaction to a write command (Transfer).

4. Chip card according to any one of claims 1 to 3, characterised in that it comprises at least one of the following features:

a means for loading data necessary for the performance of transactions into the partial structure with the aid of at least one system key;

a means for the writing of data into the definition data set for adapting the data of the definition data set to the data loaded into the partial structure;

a means for generating a further partial structure into which data required for performing a transaction can be loaded in the database of the card; and/or

a means for dynamic memory administration on the card.

5. Chip card according to any one of the preceding claims, characterised in that

the partial structures (VAS-applications) are represented by mutually independent partial structures each being assigned to a particular service provider,

the definition data set is protected against modification by the at least one system key ($K_{SO}$) and can only be modified by means of this key, and that

loading of the partial structures (VAS-applications) can proceed only with the use of the system key contained in the definition data set.

6. Chip card according to any one of the preceding claims, characterised in that the definition data set comprises at least one of the following features:

at least one authentication key ($K_{AUT}$) for authenticating the entitlement of the chip card in relation to a terminal and/or for authentication of the entitlement of a terminal in relation to the chip card;

at least one signature key ($K_{SIG_VASC}$) for signing data removed from the transfer storage;

* C L M S *

This Sheet Has Been Replaced

a key generating key ($KGK_{DEC}$) for generating terminal and application-specific keys for the verification of the authorisation for a writing procedure into the transfer storage and/or a invalidation of value data;

a PIN for the verification of the authorisation of a transaction procedure by the cardholder,

that the system key ($K_{SO}$), the authentication key ($K_{AUT}$), the signature key ($K_{SIG_VASC}$) and the key generating key ($KGK_{DEC}$) are keys individual to the cards or specific to the data structure (DF_VAS),

and that the partial structure (VAS-application) comprises at least one of the following features:

at least one value storage (EF_VALUE) for the storage of value data;

at least one internal storage (EF_INTERNAL) for the storage of internal data relating to the partial structure;

at least one information storage (EF_INFO) for the accommodation of non-internal data relating to the partial structure;

a key storage (EF_KEY) for accommodating at least one key ($K_{LVASP}$, $K_{RVASP}$), which provides security for the writing and/or information retrieval procedure into or from the value storage and/or the internal storage and/or the information storage and that the chip card further includes:

a means for writing and/or retrieving data into or from the value storage, the internal storage or the information storage and that the means for loading includes:

a means for writing the key into the key storage, protected by the at least one system key.

7. Chip card according to any one of the preceding claims, characterised in that the partial structure comprises at least one of the following integers:

a key ($K_{LVASP}$) for checking the authorisation to write data in the partial structure:

a key ($K_{RVASP}$) for checking the authorisation to retrieve data from the partial structure.

* C L M S *

This Sheet Has Been Replaced

8. Chip card according to any one of the preceding claims, characterised in that keys stored in the key storage are specific for the respective partial structure and that the

at least one partial structure (VAS-application) protects the performance of transactions by means of at least one of the specific keys ($K_{LVASP}$, $K_{RVASP}$) which is specific to the respective partial structure (VAS-application) and is independent of the keys for other partial structures (VAS-applications).

9. Chip card according to any one of the preceding claims, characterised in that the card comprises a plurality of partial structures (VAS-applications) which each serve for the performance of transactions between a particular service provider and the cardholder and that

the performance of transactions includes the writing and/or retrieval of data into, respectively from the transfer field and/or the writing and/or retrieval of data into, respectively out of the value storage.

10. Chip card according to any one of the preceding claims, characterised in that it further includes:

a means for performing transactions both between individual partial structures (VAS-applications) as well as between a partial structure and a service provider.

11. Chip card according to any one of the preceding claims, characterised in that

the system key ($K_{SO}$) is known only to the system operator (SO) and is a key individual to a card and/or specific to a data structure (VAS-container), and that

the further keys contained in the definition data set are keys individual to the card and/or specific to the data structure (VAS-container).

12. Chip card according to any one of the preceding claims, characterised in that the definition data set includes one or more of the following integers:

an identification number (EF_ID) specifying the data structure

a directory of partial structures (EF_DIR) contained in the data structure, the directory containing partial structure specific identification numbers of partial



* C L M S *

This Sheet Has Been Replaced

structures (VAS-applications) loaded into the data structure (VAS-container), as well as information containing that part of the data structure (VAS-container) in which the partial structures (VAS-applications) are physically stored;

a version number of the data structure (EF_VERSION).

13. Chip chard according to any one of the preceding claims, characterised in that it includes at least one of the following features:

a means for performing a withdrawal procedure (Take) by means of which data can be removed and/or value-cancelled from the transfer storage;

a means for generating one or more authenticity features of the data during withdrawal or cancellation of data from the transfer storage.

14. Chip card according to any one of the preceding claims, characterised in that the chip card for generating the authenticity features comprises the following:

a signature key ($K_{SIG_VASC}$) for generating a digital signature having the data withdrawn;

a means for generating a transaction number characterising the transaction which is also used in the generation of the digital signature.

15. Chip card according to any one of the preceding claims, characterised in that the signature key ($K_{SIG_VASC}$) is a private key, derived from a private key generating key and that for checking the signature by the service provider, public keys are employed.

16. Chip card according to any one of the preceding claims, characterised in that it at least comprises one of the following integers:

a means for generating terminal and partial structure-specific keys ($K_{DEC}$) by means of the key generating key ($KGK_{DEC}$);

a means for verifying the authorisation and/or the protection of a transaction by using at least one of the following integers:

the terminal and partial structure-specific key ($K_{DEC}$),

the at least one authentication key ($K_{AUT}$),

the at least one system key ($K_{SO}$),



* C L M S *

the at least one partial structure specific key ($K_{LVASP}$, $K_{RVASP}$),

the signature key ($K_{SIG_VASC}$),

the PIN,

the identification of a partial structure,

the identification of a terminal.

This Sheet Has Been Replaced



17. Chip card according to any one of the preceding claims, characterised in that the chip card includes at least one of the following integers:

a means for authentication of the authority and/or the terminal by means of the authentication key prior to starting a data retrieval or writing procedure,

a means for performing data retrieval or writing procedures which are protected by a digital signature and/or key formation.

18. Chip card according to any one of the preceding claims, characterised in that the chip card further includes at least one of the following integers:

a means for activating and deactivating the PIN-protection,

a means for changing the PIN.

19. Chip card according to any one of the preceding claims, characterised in that the data structure according to any one of the preceding claims is independent of the card platform and the chip card further includes:

a means for transferring the data structure or of parts of the data structure to another card.

20. Chip card characterised in that it comprises the following:

a storage region for the writing or retrieval of data into or out of the storage region for the transfer of monetary units and/or of value data representing other non-monetary entitlements between identical or different service providers.

21. Terminal for the use of a chip card according to any one of claims 1 to 20, characterised in that the terminal comprises:



This Sheet Has Been Replaced

a means for identifying the data structure (VAS-container) of the chip card as well as for identifying the characterisation (container-ID) identifying the data structure;

and that it further comprises at least one of the following features:

a means for retrieval of data from at least one of the partial structures and/or the definition data set and/or the transfer storage of the card;

a means for writing of data into the transfer storage of the card;

a means for loading the data required for the performance of transactions into at least one of the partial structures (VAS-applications) of the card.

22. Terminal according to claim 21, characterised in that it comprises at least one of the following integers:

a means for performing transactions between a service provider and the cardholder wherein the performance of a transaction includes at least one of the following steps:

the writing of data into the value storage,

the writing of data into the transfer storage,

the removal and/or cancellation of data from the transfer storage,

the retrieval of data from the partial structure,

the retrieval of data from the transfer storage.

23. Terminal according to either of claims 21 or 22, characterised in that it further comprises:

a means for verifying the authorisation for and/or the protection of a transaction with the aid of at least one of the following integers:

a terminal and partial structure-specific key ($K_{DEC}$),

the at least one authentication key ($K_{AUT}$) which is individual to the card or specific to the data structure (DF_VAS),

at least one system key ($K_{SO}$) which is individual to the card or specific to the data structure (DF_VAS),

at least one partial structure specific key ($K_{LVASP}$, $K_{RVASP}$),

at least one signature key ($K_{SIG_VASC}$) which is individual to the card or specific to the data structure (DF_VAS),



* C L M S *

This Sheet Has Been Replaced

a PIN which is individual to the card or specific to the data structure (DF_VAS),

an application-specific characterisation of a partial structure,

a terminal-specific characterisation of a terminal.

24. Terminal according to any one of claims 21 to 23, characterised in that the means for writing data into the transfer storage comprises:

a means for encoding of data with the use of a terminal and partial structure specific key ($K_{DEC}$) for verification of the write authorisation.

25. Terminal according to any one of claims 21 to 24, characterised in that it further comprises:

a means for performing a process of withdrawing data from the transfer storage by means of which data are removed from the transfer storage and/or are value cancelled.

26. Terminal according to any one of claims 21 to 25, characterised in that it further comprises at least one of the following integers:

a means for characterising data of the transfer storage as having been removed,

a means for characterising data of the transfer storage as having expired.

27. Terminal according to any one of claims 21 to 26, characterised in that the means for performing transactions comprises at least one of the following integers:

a means for changing value data in the partial structure;

a means for performing transactions between different service providers (interservices) at the expense of, respectively for the benefit of the cardholder.

28. Terminal according to any one of claims 21 to 27, characterised in that it further comprises:

* C L M S *

This Sheet Has Been Replaced

a means for the authentication of the authority of the terminal in relation to the card and/or the card in relation to the terminal with the aid of at least one authentication key;

a means for protecting a transaction by the cardholder by means of a PIN;

a means for activating and deactivating the PIN-protection.

29. Terminal according to any one of claims 21 to 28, characterised in that it further comprises at least one of the following integers:

a means for transferring a terminal specific characterisation to the card;

a means for transferring a characterisation specifying the partial structure to the card;

a means for authentication of the authorisation using a terminal and partial structure specific key as well as the terminal and partial structure specific characterisation,

a means for performing data retrieval or writing procedures which are protected by a digital signature and/or encodification.

30. Terminal according to any one of claims 21 to 29, characterised in that it further comprises at least one of the following integers:

a means for selecting a partial structure (VAS-application);

a means for viewing a partial structure at the terminal;

a means for viewing the data of a partial structure at the terminal;

a means for loading a partial structure (VAS-application) onto the card;

a means for loading a characterisation onto a loaded partial structure (VAS-application) onto the card,

a means for deleting a partial structure from the card,

a means for substituting a partial structure by a different partial structure;

a means for transferring a partial structure onto another card;

a means for interpreting a partial structure with regard to its function and its associate service provider as well as for data retrieval and viewing of information stored therein.

* C L M S *

This Sheet Has Been Replaced

31. Process for performing transactions between a cardholder and at least one service provider involving the use of a chip card as well as a terminal wherein the process comprises one of the following steps:

Provision of a data structure stored on the chip card into which data can be loaded for enabling the performance of the transaction between the cardholder and a service provider (VAS-data) as well as writing into or reading of data from the data structure (VAS-application);

provision of a transfer storage region (EF_TRANSFER) for the storage of monetary units or data representing non-monetary entitlements to be exchanged or presented during the performance of the transaction, as well as writing of data into the transfer storage or retrieval of data from the transfer storage.

32. Process according to claim 31, characterised in that the process comprises at least one of the following steps:

Use of a chip card according to any one of claims 1 to 20 ;

use of a terminal according to any one of claims 21 to 30;

writing or retrieval of data into or from the value storage or internal storage or information storage of at least one of the partial structures (VAS-applications).

33. Process according to claim 31 or 32, characterised in that it furthermore includes at least one of the following steps:

authentication of the authority of the terminal and/or the chip card with the use of at least one key;

protection of the transaction by the use of a digital signature and/or encodification by the use of at least one key.

34. Process for loading of data onto a chip card using a terminal, characterised in that the process comprises at least one of the following steps:

Loading of data into a partial structure (VAS-application) of the card;

writing of data into the definition data set of the card.

35. Process for loading of data according to claim 34, which includes at least one of the following steps:



* C L M S *

use of a chip card according to any one of claims 1 to 20,

use of a terminal according to any one of claims 21 to 30.

36. System for performing transactions, characterised by a chip card according to any one of claims 1 to 20 and a terminal according to any one of claims 21 to 30.

This Sheet Has Been Replaced





