



159159

WO 2004/072911

PCT/DE2004/000083

1

**METHOD FOR VERIFYING THE VALIDITY OF DIGITAL FRANKING
NOTES AND DEVICE FOR CARRYING OUT SAID METHOD**

Description:

5

It is a known procedure to provide mailpieces with digital postage indicia.

10 In order to make it easier for the senders of the mailpieces to produce postage indicia, it is possible, for example, with the franking system used by the Deutsche Post AG, to produce postage indicia in a customer system and to output them on a printer via any desired interface.

15

A method of this generic type is disclosed in German Preliminary Published Application DE 100 20 402 A1.

20 In order to avoid fraudulent use of this method, the digital postage indicia contain cryptographic information, for example, about the identity of the customer system controlling the production of the postage indicium.

25 The invention is based on the objective of creating a method with which the authenticity of the postage indicia can be verified quickly and reliably. In particular, the method should be suitable for use in verification procedures on a large scale, especially in
30 mail centers or freight centers.

According to the invention, this objective is achieved in that the method for verifying the authenticity is



G00002

carried out in such a way that a data key is generated and transmitted from a central database (ZinS central system) to local payment assurance systems (ZinS local). This is done by using a generated franking key
5 and a postage indicium applied onto a mailpiece, whereby cryptographic information contained in the postage indicium is decrypted and used to verify the authenticity of the postage indicium.

10 In order to increase the security of the method, it is advantageous for the local payment assurance systems to import the data key and to transmit a result of the import to the central database (ZinS central system).

15 In an especially advantageous embodiment of the method, the result of the import is transmitted as a data record.

Preferably, the data record contains a key. The key can
20 have various attributes. In particular, it can be either a symmetrical key or an asymmetrical key. Depending on the intended use, the key serves to decrypt and/or encrypt information. By virtue of its nature, the key can likewise transport information. For
25 example, the key contains information about the franking key, its key generation and/or its end date.

In order to ensure a uniform key exchange, it is advantageous for the central database (ZinS central
30 system) to check the result of the import and to transmit it to a value transfer center (Postage Point).

This embodiment of the method makes it possible to carry out further process steps in the value transfer center as a function of the result of the import.

- 5 The result can be checked in various ways. However, it is especially advantageous for the result to be checked by decryption of the key.

A preferred embodiment of the method is characterized
10 in that, once the data key has been successfully imported into essentially all local payment assurance systems (ZinS local), the data key is released as a new franking key at the value transfer center (Postage Point) and is subsequently used for producing new postage
15 indicia.

This embodiment of the method makes it possible to carry out a key exchange in the value transfer center as a function of previously performed key exchanges in
20 the payment assurance systems. This measure ensures that postage indicia will only be produced using the new key once the new key is already present in the local payment assurance systems. This ensures that the local payment assurance systems can verify the validity
25 of each of the postage indicia produced.

This especially preferred embodiment of the method with a control of the key exchange in the value transfer center as a function of the key exchanges in the local
30 payment assurance systems is especially advantageous in combination with at least some of the other process steps according to the invention.

Particularly, through a combination of several features, it is ensured that the key exchange can be performed quickly and reliably and that the execution of each key exchange is verified.

5

When the key exchange is performed, it is especially advantageous for the new data key to be transmitted from the value transfer center (Postage Point) to the central payment assurance system.

10

Here, it is especially advantageous for the value transfer center to encrypt the data key with a transport key (KT).

15 Here, it is practical for the transport key to be encrypted with a master transport key (KTM).

Preferably, the data key is generated in the area of the value transfer center. This has the advantage that
20 fraudulent changes to the data key during its transport to the value transfer center are prevented.

In order to further increase the data security, it is advantageous for the data key to be provided with key
25 identification information.

Advantageously, the key identification information is likewise transmitted in encrypted form.

30 In order to ensure that a valid data key is used for each of the encryption and/or decryption steps employed, it is advantageous for the data key to be

provided with a generation counter and/or to be transmitted together with the generation counter.

In order to avoid the use of invalid keys, it is also
5 advantageous for the franking key to be provided with an end date for the preceding data key and/or to be transmitted together with the end date.

The described data key can be used for one or more
10 partial verifications as well as for producing postage indicia and/or for decrypting information contained in the postage indicia.

It is especially advantageous for one of the partial
15 verifications to comprise the decryption of cryptographic information contained in the postage indicium.

By integrating the decryption of the cryptographic
20 information into the verification process, it is possible to ascertain the authenticity of the postage indicia directly so that the verification can be carried out in real time.

25 Moreover, it is advantageous for one of the partial verifications to comprise a comparison between the production date of the postage indicium and the current date. The integration of the production date of the postage indicium - especially in encrypted form -
30 increases the data security, since multiple use of a postage indicium is prevented through the comparison between the production date of the postage indicium and the current date.

In order to further increase the verification speed, it is advantageous for the reading unit and the verification unit to exchange information by means of a synchronous protocol.

5

In another likewise advantageous embodiment of the invention, the reading unit and the verification unit communicate with each other via an asynchronous protocol.

10

Here, it is especially advantageous for the reading unit to send a data telegram to the verification unit.

15

Preferably, the data telegram contains the content of the postage indicium.

Additional advantages, special features and practical refinements of the invention can be gleaned from the subordinate claims and from the presentation below of preferred embodiments making reference to the drawings.

20

The drawings show the following:

Figure 1 an especially preferred embodiment of a key distribution according to the invention;

25

Figure 2 application cases of a key distribution system according to the invention;

Figure 3 a schematic diagram of an interface between a central payment assurance system (ZinS central system) and a value transfer center (Postage Point);

30

Figure 4 process steps for integrating a data key into the central payment assurance system (ZinS central);

5

Figure 5 a schematic diagram of a key distribution from the central payment assurance system (ZinS central system) to local payment assurance systems including the appertaining crypto-systems;

10

Figure 6 a connection of the DLL interface;

Figure 7 a schematic diagram of process steps for encapsulating components and for handling error messages.

15

Below, the invention will be described with reference to a PC franking system. Here, the process steps that serve for payment assurance are independent of the system used for producing the postage indicia.

20

The described decentralized verification at individual verification stations, especially in mail centers, is especially preferred, but by the same token, centralized verification is also possible.

25

In especially preferred embodiments of the invention, the individual verification stations are local payment assurance systems that are preferably connected to a central payment assurance system via a data connection.

30

In the especially preferred embodiments described, the central payment assurance system is also connected to a value transfer center.

- 5 Especially preferred embodiments of the value transfer center are referred to below as a Postage Point. Advantageous embodiments of the central payment assurance system are referred to below as ZinS central.
- 10 The technical interface to be implemented between the value transfer center and the payment assurance system consists of an exchange of cryptographic keys.

The key that is to be exchanged between the value
15 transfer center and the payment assurance system ensures that the produced postage indicia are forgery-proof. This is achieved especially in that part of the content of the 2D barcode that forms the postage indicium is encrypted using said key. Since, for
20 performance-related reasons, the selected key is a symmetrical key, it has to be transmitted from the value transfer center to the payment assurance system, where it is distributed to the individual mail centers.

- 25 The reliable storage of the keys is ensured by using special crypto-cards. The invention encompasses several application cases in which these keys are managed using this special hardware. The entire life cycle of these keys, from their generation, their export and their
30 distribution all the way to their import into the decentralized systems is used in order to optimize the process parameters.

An especially preferred sequence for the key distribution is shown in Figure 1.

Figure 1 shows an especially preferred key distribution
5 between the value transfer center and the central payment assurance system.

In a first process step 1 for an exchange of a franking key used in the production of postage indicia, first of
10 all, a new franking key is generated.

The term franking key here is by no means to be understood in a restrictive sense since the described keys can be used in different ways for producing
15 postage indicia.

For example, it is possible to use the franking key directly for producing postage indicia.

20 However, it is likewise possible and especially advantageous for systems with an especially high security against manipulation to produce postage indicia with a multi-encrypted data content, whereby the data content is preferably formed as a result of
25 several operations, whereby at one or more places, the result of an operation is incorporated into the postage indicium with the franking key.

For example, a crypto-string of the type disclosed in
30 German Preliminary Published Application DE 100 20 402 A1, is incorporated into the franking key.

In order to further improve the protection against a fraudulent production of postage indicia, an event-specific exchange of the franking key takes place.

- 5 In the presented case, the franking key is newly generated at regular intervals, for example, after the expiration of a predefinable time interval.

10 However, it is likewise possible to carry out a new generation of the franking key as a function of other parameters, for example, on the basis of the loaded postage amounts and/or of the franked mailpieces.

15 As a matter of principle, a new generation of the new franking key can be carried out anywhere. However, in order to increase the data security, it is advantageous to minimize the transmission effort for the new franking key and to generate the key in the value transfer center, or in the area of the value transfer
20 center.

In order to ensure a high level of protection of the method against fraud, it is advantageous for information contained in the postage indicia to be
25 decrypted on the basis of a franking key in the area of local payment assurance systems, for example, in mail centers or freight centers.

In order to ensure this, the franking keys are
30 transmitted from the value transfer center to the central payment assurance system. Preferably, the transmission is automated.

The exchange is preferably effectuated via a server of the payment assurance system (ZinS central server). The value transfer center (Postage Point) does not have to be configured. Since the server does not have to know
5 the value transfer center (ZinS local systems) and its appertaining crypto-systems, exclusively the ZinS central server is of importance for said server.

After the new generation of a preferably symmetrical
10 franking key, the latter is securely transmitted to the central payment assurance system - process step 2 in Figure 1 - and distributed from there to the crypto-systems located in the local payment assurance systems - process step 3 in Figure 1. The local payment
15 assurance systems return the result of the import operation - process step 4 in Figure 1 - and thus confirm the successful key import. The result is compiled by the central system, validated and transmitted to the value transfer center (Postage
20 Point) - process step 5 in Figure 1.

Once the key has been successfully imported into all of the crypto-systems of the local payment assurance systems, it is released at the value transfer center
25 (Postage Point) and subsequently used for the generation of new postage amounts - process step 6 in Figure 1.

The preferably symmetrical keys are an integral part of
30 the forgery security of postage indicia produced using the franking key, whereby said postage indicia are, for example, in the form of two-dimensional barcodes. Thus, the exchange of these keys has to be secured by strong

cryptography. In order to ensure this, adherence to the following points is especially important:

- 5 • **Confidentiality** of the content: it must not be possible for third parties to read out the transmitted keys during the transmission. Moreover, it must also be ensured that the keys are securely stored and cannot be read out by third parties. The latter functionality is
10 ensured by the WebSentry crypto-card.
- **Integrity** of the content: it must not be possible for third parties to forge parts of the
15 key.
- **Authentication**: both communication partners must be certain that the identity of the sender/recipient is correct. From the viewpoint of the recipient, this means that the key was
20 generated by the Postage Point. From the viewpoint of the sender, it must be ensured that only valid ZinS local crypto-systems are allowed to receive the key.
- 25 In the symmetrical method presented, both communication partners have the same key KT . The value transfer center (Postage Point) uses the key KT to encrypt the data key KD that is to be transmitted and then transfers said data key KD to the server of the payment
30 assurance system (ZinS central system).

From the central payment assurance system, this key is then further distributed to the ZinS local crypto-

systems of the local payment assurance systems. These systems likewise have access to the key KT and can thus decrypt the key KD once again. Since the key KT is used to securely protect the key KD during the transport, it
5 is also referred to below as the transport key.

Since all local payment assurance systems receive the same data, it is not necessary to define a separate transport key KT between each local crypto-server and
10 the value transfer center (Postage Point). For security reasons, however, this key KT should be renewed like the data key KD at regular intervals. However, since not as much text is encrypted with this key as with the key KD, a longer exchange interval can be selected. An
15 exchange interval of one or two years is especially advantageous here.

An essential component of this approach is the key exchange of the transport keys KT which, for security
20 reasons, should not take place via the same channel as the exchange of the data keys KD. This exchange is not carried out manually. Since this procedure has to be carried out at regular intervals for several of the local payment assurance systems, another method has to
25 be provided here, by means of which the exchange of the transport keys can likewise be automated.

For this purpose, the ANSI standard X9.17 (Key Management for Financial Services on the basis of
30 symmetrical methods) defines another key level that is referred to as the master transport key (referred to below as KTM). This key has to be installed on the crypto-card under special security precautions and

consequently only has to be exchanged at extended intervals. Here, the same KTM is installed on all systems. This key then encrypts the transport keys KT that are thereby distributed and imported in an automated manner via the same channel. The distribution takes place in the same way as the distribution of the data keys. The initialization of the individual systems or of their crypto-cards is described in greater detail in the sections below.

10

In order to ensure the integrity of the message as well as the authentication of the sender (= Postage Point), a matrix code (MAC) is formed for the individual key messages. In order to generate the MAC, the signature key KS is needed which, like the KTM, is imported during the initialization of the crypto-card. The signature of the data key messages is subsequently carried out with this KS. The confidentiality of the information during the transmission of the messages in the Intranet of the Deutsche Post is thus secured through the use of this strong cryptographic method. An especially preferred method for the encryption and decryption of messages is Triple DES (key length 168 bits). Hash values are preferably computed by means of the SHA-1 algorithm.

20
25

The various periods of validity that exist at the Postage Point and on the crypto-systems of the payment assurance systems have to be taken into account for the distribution and storage of the data keys. A maximum of two keys KD have to be available at the Postage Point at any given point in time, namely, one key that is

30

currently valid and another key with which newly generated postage amounts are to be encrypted (KD_{new}).

In the 'exchange' mode of operation of the existing key
5 with the key (KD_{new}), the new key is transmitted to the
crypto-systems of the payment assurance systems. Once
this key has successfully been imported by all of the
crypto-systems of the local payment assurance systems,
a release message of the ZinS central system is
10 generated and, as of this point in time, KD_{new} is used
for the encryption of new postage amounts in the
Postage Point. As of this point in time, the old KD
should be deleted from the Postage Point and should not
be used any more for the generation of new postage
15 amounts.

The situation is different with the crypto-systems of
the local payment assurance systems: since a downloaded
postage amount can be used for a predefinable period of
20 time of, for example, three months, in order to produce
postage indicia, there are several keys KD that have to
be available for verifying the postage indicia.

Moreover, when it comes to the distribution of the
25 keys, it should also be taken into consideration what
happens to keys that could not be imported into some of
the crypto-systems and could therefore not be activated
at the Postage Point. It is possible that these keys
were imported into other crypto-systems and, as a
30 matter of principle, should be taken into consideration
there during future verifications.

In order to reach a uniform condition here, which allows a clear versioning of the keys as well as the simplest possible system maintenance, the following execution form of the method of key distribution is especially advantageous:

- 10 a) Data keys are transmitted in encrypted form with an unambiguous key ID (key phase indicator), an ambiguous generation counter and an end date for the valid preceding data key.
The generation counter serves to distinguish erroneous multiple loading attempts from desired updates of the symmetrical data keys (ensuring transaction security, see g)).
- 15 b) Each transmission of a data key from the value transfer center (Postage Point) should be acknowledged by the central payment assurance system by means of a confirmation message.
- 20 c) If the acknowledgement is positive, the data key has been successfully installed in all local payment assurance systems and can be issued to the customers using PC franking.
25 In this case, the generation counter is increased by one for the transmission of a succeeding data key.
- 30 d) If the acknowledgement is negative, the data key has not been successfully installed in all local payment assurance systems and should not be issued by the value transfer center to customer systems meant for the production of postage

indicia, since otherwise there is a risk of a mass diverting of valid mailpieces.

5 In this case, the generation counter is not increased by one for the transmission of a succeeding data key, that is to say, it remains at the value of the preceding transmission.

10 e) If there is no acknowledgement, in the interim, the value transfer center (Postage Point) cannot start any further key transmission to the central payment assurance system (ZinS central) (such attempts would, of course, be ignored by the payment assurance system, but should also be blocked in the value transfer center).

15 f) If an acknowledgement by the payment assurance system is absent for a prolonged period of time (time-out exceeded), then the value transfer center (Postage Point) can trigger an alarm via
20 its direct or indirect user interface.

25 g) As soon as a crypto-card receives a data key, it deletes all of the data keys that have the same generation counter value as the most recent transmission. This ensures that, in case of error-related multiple transmission, keys that could previously only be successfully loaded onto some of all of the crypto-cards are deleted. This allows a transaction-secure key
30 transmission.

h) On the crypto-cards in the local payment assurance systems, a routine is invoked

repeatedly, preferably regularly, especially daily, that deletes the data keys that are no longer needed. A data key is deleted (in an advantageous embodiment in addition to Point g))
5 when the end date stored for the succeeding data key in the CKA_END_DATE attribute is smaller than the current system date.

10 i) When it comes to the end date of the preceding key, a grace period (as short as possible) should be planned since postage indicia produced with an expired postage amount are still recognized as being valid for two more days
15 after the end of their validity when they are checked in the area of the local payment assurance system. Moreover, a time-lag between the production and the release of the new data key also has to be taken into account.

20 Figure 2 shows an overview of areas of application of the presented key management and its use in the area of payment assurance. Moreover, preferred areas of application are presented.

25 The application cases are described in greater detail below.

Subsequently, details about the described key management method will be presented.

30

The described applications are presented by way of an example with the use of crypto-cards.

First of all, the way in which the crypto-cards are initialized in the area of the value transfer center is presented:

- 5 • installation and configuration of the card, uploading the card firmware, insofar as the manufacturer has not already done so. The functionality of the firmware was expanded by embedded code (proprietary software routines)
- 10 and, for security reasons, the PKCS#11-specific (Public-Key Cryptography Standards) key routines (generation, deletion, etc.) should be blocked for the user. Thus, a higher key security is ensured on the card.
- 15 • definition of clusters (with several crypto-cards)
- 20 • Generation and storage of a local master key LMK. The LMK should be distributed over at least three components of which two components are especially advantageous for recreating or newly initializing crypto-server cards. Preferably, each of the components is written onto Smart-
- 25 Cards with PIN protection, whereby the security administrators receive a SmartCard. In addition, backup SmartCards should also be made. Subsequently, the LMK serves as the above-described master transport key KTM.
- 30 • user administration: deletion of the default security administrator and definition of the

security administrators including the requisite authentication rules (SmartCard-based)

- 5 • generation of an initial signature key KS, encryption (wrapping) with the KTM and storage as a file. The later copying of this file onto a diskette should be possible (access to the file/diskette should only be possible for the security administrators).

10

- generation of a first transport key KT, creation of the appertaining key message and storage of the message in a file as well as later copying of this file onto a diskette (access should only
15 be possible for the security administrators).

Generating the transport master keys

20 New transport master keys (KTM) are preferably generated in the manner described below. The local master key (LMK) of an appertaining confirmation card serves as the KTM. For security reasons, the LMK should be divided into at least three components, of which at least two components are needed for re-importation.

25

The individual components are stored on SmartCards, whereby each security administrator receives a SmartCard and secures it with an individual PIN. Keeping the PIN secret and storing the SmartCards in a safe
30 place is to prevent unauthorized persons from being able to access them.

In order to implement the transport master keys, there are preferably at least two LMK components - corresponding to two different authorization cards - so that an automated implementation of a dual control principle is carried out.

The KTM has to be a Triple DES key. The ID attribute of the key consists of a type identification and an unambiguous number.

10

Type identification: KT

Unambiguous number: 01 to 99.

Length: fix 4 bytes is filed as CK_CHAR[].

15 As a matter of principle, various security mechanisms are suitable for ensuring that only authorized persons are able to carry out an exchange of the keys. The embodiment described below using signature keys, however, is particularly advantageous since it accounts
20 for especially high security against manipulation.

The signature key secures the integrity of the key messages. It can also be used before the import of the key to ascertain whether the sender of the key messages
25 is the Postage Point. The generation of the signature key should only be possible by a security administrator who is registered via a SmartCard. This should be a TDES key that has the security attributes 'sensitive' set to TRUE and 'extractable' set to FALSE in order to
30 prevent the key plain text from being discovered outside of the card. The ID attribute of the key consists of a type identification and an unambiguous number.

Type identification: KS

Unambiguous number: 01 to 99.

Length: fix 4 bytes is filed as CK_CHAR[].

5

In order for the key to be exported, it has to be wrapped with the KTM and it is then stored as a file on a diskette. The diskette has to be kept in a secure place and serves for the initialization of the crypto-server cards. The integrity of the key file is ensured by a processing routine preferably stored in the cards that is used for the wrapping.

Preferred attributes of the transport key KS are compiled in the following table.

15

Attributes of the transport key KS	
KeyAttribute	Value
CKA_ID	KS + consecutive unambiguous number (KS01 to KS99) for unambiguous designation of the key.
CKA_LABEL	Is not filled, corresponds to the default value of the hardware at the time of the generation.
CKA_START_DATE	Date as of which the key is supposed to be valid in the PKCS#11 format (is specified by the security administrator)
CKA_END_DATE	Date after which the preceding key is to be deleted.
CKA_SENSITIVE	TRUE
CKA_EXTRACTABLE	FALSE
CKA_ENCRYPT	TRUE

CKA_DECRYPT	TRUE
-------------	------

The random number in the LABEL attribute serves to confirm the successful import into the crypto-server cards. A hash value (for example, by means of SHA-1) is formed for this random number and it is used for confirming the successful import as well as for releasing the transport key.

The CKA_ID and CKA_LABEL attributes are to be filled as CK_CHAR. All further attributes are defined in the type via the pkcs11.h file.

The signature key is encrypted with the KTM (=LMK, CKM_KEY_WRAP_WEBSENTRY mechanism) and is uploaded onto the hardware on site like the LMK.

The generation of the transport key will be described below.

In this application case, a transport key including the appertaining key message is generated. Preferably, the key generation module is configured in such a way that the generation of the transport key and/or of the appertaining key message can only be initiated by a security administrator. The exchange interval should be one or two years.

The transport key, in turn, is a TDES key with the following attribute settings:

Attributes of the transport key KT	
KeyAttribute	Value

CKA_ID	KT + consecutive unambiguous number (KT01 to KT99) for unambiguous designation of the key.
CKA_LABEL	Random value having the length of 64 bytes that was generated using the PKCS#11 C_GenerateRandom method.
CKA_START_DATE	Date as of which the key is supposed to be valid in the PKCS#11 format (is specified by the security administrator)
CKA_END_DATE	Date after which the preceding key is to be deleted.
CKA_SENSITIVE	TRUE
CKA_EXTRACTABLE	FALSE
CKA_ENCRYPT	TRUE
CKA_DECRYPT	TRUE

The random number in the LABEL attribute serves to confirm the successful import into the crypto-server cards. A hash value (for example, by means of SHA-1) is
5 formed for this random number and it is used for confirming the successful import as well as for releasing the transport key.

The CKA_ID and CKA_LABEL attributes are to be filled as
10 CK_CHAR []. All further attributes are defined in the type via the pkcs11.h file.

The transport key is encrypted with the KTM (=LMK, CKM_KEY_WRAP_ WEBSENTRY mechanism) and a message having
15 the following structure is formed for the transmission from the value transfer center to the central payment assurance system:

Item	Length	Byte no.	Designation	Description
1	2	f1 - f2	MsgType	Identification of the key message, constant 'KT'
2	1	f3	Version	Version of the structure of the message, 1 byte starting with the value '01'
3	4	f4 - f8	KeyID	ID of the transport key in plain text
4	4	f9 - f12	SigKeyID	ID of the key used for the signature
5	n-13	f13 - f _n	TranspKeyEncrypt	Result of the encryption (wrap) of the transport key
6	24	f _{n+1} - f _{n+24}	MAC	MAC for the key message; is formed in that an SHA-1 hash value is formed for the Fields 1 to 5 of the message, and this hash value is encrypted with the signature key (CKM_DES3_CBC_PAD mechanism, the IV is filled up with zeros; ID see field 5).

Subsequent to the further distribution, this transport key message is transferred to the ZinS central server.

- 5 The interface is realized as a Session Bean, this service is looked up by means of a naming service (Java Naming and Directory Interface - JNDI). The method for

the transport should expect the above-mentioned data block.

Moreover, the message is stored at the Postage Point as
5 a file so that the security administrators can later store it onto one or more diskettes that are to be kept in a safe place. The diskettes are then used for the initialization of new crypto-server cards.

10 A preferred embodiment for releasing the transport key will be explained below.

The value transfer center is configured in such a way that it can release the transport key KT. In order to
15 release the transport key KT, there is an interface via which the central value transfer center can release this transport key once it has successfully been distributed and successfully imported into all local payment assurance systems (ZinS crypto-systems). Only
20 through the release is the appertaining transport key used subsequently for the encryption of data keys (KD).

The interface is realized as a Session Bean. This service is looked up by means of a naming service.

25

The data structure for the releasing procedure preferably has the following parameters:

Item	Length	Byte no.	Designation	Description
1	4	f1 - f4	KeyID	ID of the transport key in plain text
2	201	f4 - f24	RandomHash	SHA-1 hash value

				of the random number that was transferred as the LABEL attribute of the transport key
3	1	f25	State	Result of the encryption: true = processing necessary, release can take place false = processing not successful
4	128	f26 - f153	Message	Detailed message about the error causes, or detailed success message

The authentication of the key allocation system of the payment assurance system (ZinS KeyManagement system) vis-à-vis the PP KeyManagement system is carried out via Parameter 2. This is a hash value that is formed using the SHA-1 method from the LABEL attribute of the transport key. The LABEL attribute is filled with a random value at the time of the generation of the transport key. Since the transport key and all of its attributes are encrypted during the transmission, only the ZinS crypto-server can compute the correct hash value.

If the transmitted hash value is identical to the hash value of the KT computed for the LABEL attribute, whereby said hash value is located on the Postage Point WebSentry module, and if the transmitted processing status = true, then the transport key is activated.

This means that subsequently, the data key messages are encrypted with the transport key.

A variant of this application case exists if the
5 processing is erroneous (transmitted status = false).
In this case, the status for this key generation and
key distribution is protocolled and the appertaining
transport key is deleted accordingly.

10 A preferred example of a generation of new data keys is
shown below.

In this application case, a data key including the
appertaining key message is generated. Preferably, the
15 key allocation system is configured in such a way that
this action can only be initiated by a security
administrator. This should be done every three months.
If a data key KD is in circulation for which no
feedback (release or error) has yet been received from
20 the central payment assurance system (ZinS central
system), then the generation of new KDs is blocked
until the feedback has been received.

The data key (KD) serves for the encryption of certain
25 matrix code contents and also ensures that no valid
postage indicia can be produced for which no payment
has been made to the postal service provider. This data
key serves to verify the digital postage indicia at the
ZinS crypto-servers. The data keys are likewise TDES
30 keys that are generated using the PKCS#11 function
C_GenerateKey and that have the following attributes:

<i>Attributes of the data key KD</i>

KeyAttribute	Value
CKA_ID	Consecutive unambiguous number (e.g. 01) for unambiguous designation of the key (without a prefix); corresponds to the key phase indicator that is encoded in the postage indicium and in the PostageID.
CKA_LABEL	1st Byte: Generation counter for simplified deletion of keys in the ZinS cryptosystems that could not be activated Bytes 2-65: random value having the length of 64 bytes that was generated using the PKCS#11 C_GenerateKey method.
CKA_START_DATE	Date as of which the key is supposed to be valid in the PKCS#11 format (is filled with the current date)
CKA_END_DATE	Does not indicate the end of the validity of the key itself but rather the end of the validity of the predecessor. The date after which the preceding key is to be deleted.
CKA_SENSITIVE	TRUE
CKA_EXTRACTABLE	FALSE
CKA_ENCRYPT	TRUE
CKA_DECRYPT	TRUE

The values for the CKA_ID attribute and the generation counter are prescribed by the system. The CKS_ID is always counted upwards by one, whereas the generation counter is only increased when the key release has been successful. The CKA_ID and CKA_LABEL attributes are to be filled as CK_CHAR []. All further attributes are defined in the type via the pkcs11.h file.

The random number in the LABEL attribute serves to confirm the successful import into the crypto-server cards. A hash value is formed (by means of SHA-1) for this random number and it is used for confirming the successful import as well as for releasing the transport key.

The generation of a message for the exchange of the data key is somewhat more complex and consists of the following sequence:

1. The data key is encrypted with the KTM (=LMK, CKM_KEY_WRAP_ WEBSENTRY mechanism). This has the advantage that all of the attributes of the key (among others, CKA_EXTRACTABLE=false) are concomitantly encrypted and are set correctly again at the time of the decryption. With this encrypted byte sequence, an interim message with the following structure is formed:

Structure of an interim message for data key KD				
Item	Length	Byte no.	Designation	Description
1	n	f1 - n	DataKeyEncrypt	Result of the encryption (wrap) of the data key KD with the special WebSentry mechanism (key KTM)

2. The interim message, in turn, is encrypted with the active transport key KT making use of the CKM_DES3_CBC_PAD mechanism (the initialization vector IV is filled with zeros).

3. The message to be distributed is formed and it has the following structure:

Structure of the distribution message for the data key KD				
Item	Length	Byte no.	Designation	Description
1	2	f1 - f2	MsgType	Identification of the key message, constant 'KT'
2	1	f3	Version	Version of the structure of the message, 1 byte starting with the value '01'
3	1	f4	KeyID	ID of the data key in plain text
4	2	f5 - f7	KeyGeneration	Generation counter in plain text, e.g. '01'
5	4	f8 - f12	SigKeyID	ID of the key used for the signature
6	4	f12 - f16	KTID	ID of the transport key used for the encryption of the interim message
7	n	f16 - f _n s	HelperMsgEncrypt	Result of the encryption of the interim message
8	24	f _{n1+1} - f _{n1+24}	MAC	MAC for the interim message; is formed in that an SHA-1 hash value is generated for the Fields 1-7 of the message and this hash value, provided with the signature key

				(CKM_DES3_CBC_PAD mechanism, the IV is filled with zeros; ID see field 5), is encrypted.
--	--	--	--	--

4. Subsequently, the data key message is transferred for further distribution to the ZinS central server. Moreover, it is stored by the security administrators onto one or more diskettes that have to be kept in a secure place so that they can be used for the initialization of new ZinS crypto-server cards.
- 5
- 10 The advantage of the double encryption of the message content is that less cipher text has to be transmitted to the KTM via the Intranet and consequently, cryptanalysis of this key is rendered much more difficult.
- 15 For purposes of releasing the data key KD, an interface and a protocol mechanism are provided by means of which the release of the data key is protocolled. The central payment assurance system is preferably configured in such a way that the data key is only released after the successful distribution and successful import of a data key into the crypto-systems of the local payment assurance systems. Only through this release is the appertaining data key subsequently used for the encryption of crypto-strings that are to be incorporated into the postage indicia and then the appertaining key identification information KeyID is encoded in the identification information (PostageID) of postage amounts.
- 20
- 25

The interface is realized via CWMS (computerized work management systems) between the central payment assurance system (ZinS central) and the local payment assurance system (ZinS local). The value transfer center (Postage Point) PP receives the information via the appropriate bean. The data structure for the releasing procedure has the following contents:

Structure of the method for releasing data keys KD				
Item	Length	Byte no.	Designation	Description
1	1	f1	KeyID	ID of the data key in plain text
2	20	f2 - f22	RandomHash	SHA-1 hash value of the random number that was transmitted as the LABEL attribute of the data key
3	1	f23	State	Result of the encryption: true = processing successful, release can take place false = processing not successful
4	128	f23 - f151	Message	Detailed message about error causes, or detailed success message

10 The authentication of the key allocation system of the payment assurance system vis-à-vis the key allocation system of the value transfer center PP is carried out via parameter 2. This is preferably a hash value that is formed using the SHA-1 method from the LABEL

attribute of the data key. The LABEL attribute is filled with the random value at the time of the generation of the data key. Since the data key and all of its attributes are encrypted during the transmission, the correct hash value can only be computed by the crypto-server of the payment assurance system.

If the transmitted hash value is identical to the hash value of the KD that has been computed for the LABEL attribute and is located on the Postage Point-WebSentry module, and if the transmitted processing status = true, then the data key is activated. This means that subsequently, the crypto-strings for the postage indicia/postage amounts are encrypted with this data key. When a data key is activated, the generation counter of the signature key is increased by one.

A variant of this application case exists if the processing is erroneous (transmitted status = false). In this case, the status for this key generation and key distribution is protocolled and the appertaining data key is correspondingly deleted on the card. In this case, the generation counter is not increased by one.

Preferably, the key allocation systems contain a status memory to store the executed key generations. As long as no feedback has been received from the central payment assurance system (ZinS central system) about the executed key distribution, the status is displayed as open. After the successful feedback and distribution, the key is designated as having been

activated. In case of error, the transmitted status message is displayed. In case of errors or the prolonged absence of a release message, an error investigation routine is invoked.

5

It is advantageous to perform an archiving of the keys. A preferred archiving of the keys in the area of the value transfer center will be described below. In order to secure the keys, the security administrator can start an archiving function that encrypts all of the keys (except for KTM) with the KTM (CKM_KEY_WRAP_WEBSENTRY mechanism) and returns them. These keys should be securely stored in the database or in a secure file system area.

15

After the successful archiving, all of the keys are deleted that have exceeded their initial validity date by more than six months and are no longer active.

20 Keys are restored - especially in the area of the value transfer center PP - in that the archived key data is decrypted again (unwrapped) and stored on the card. The mechanism used is once again CKM_KEY_WRAP_WEBSENTRY.

25 After a key has been decrypted, a key with the same key identification information KeyID that already exists on the card is deleted.

Through special security measures of the WebSentry card and through the distribution onto several SmartCards, the master transport key KTM is reliably secured against being compromised.

30

If nevertheless an exchange of the master transport key is to be made, then analogous to the application case involving an initialization of a "crypto-card" (PP), a new KTM as well as new signature keys, transport keys and data keys have to be generated.

The prior KTM remains on the card as a so-called "dormant LMK" and has to be appropriately deleted by the security administrator.

10

Preferred application cases of the key allocation system will be described below. The presentation also applies to applications in all areas of the payment assurance system. This is especially advantageous if individual components are implemented preferably in the area of the local payment assurance system or of the central payment assurance system. However, use in each of the other payment assurance system is likewise possible.

20

A first application case is the initialization of the crypto-card in the area of the payment assurance system.

25 In order to initialize the crypto-card, the following basic configurations are to be undertaken, whereby most of the functionality can be managed via a management tool (WebSentryManager):

- 30
- installation and configuration of the card, uploading the card firmware, insofar as the manufacturer has not already done so. The firmware contains the Embedded Code (proprietary

software routines) (the latter functionality is integrated into the WebSentryManager)

5 • definition of clusters (with several crypto-cards) (WebSentryManager)

 • import of the transport master key (KTM), see separate application case (functionality is provided by the WebSentryManager)

10 • user administration: deletion of the default security administrator and definition of the security administrators including the requisite authentication rules (SmartCard-based)

15 Generation of two "normal" users (one for key verification / one for key import) that authorize themselves via a predefined PIN. The functionality of the user administration is likewise provided by the WebSentryManager.

20 • import of the signature keys (KS); see separate application case

25 • import of the transport keys (KT); see separate application case

 • optionally also import of the data keys (KD); (likewise see their own application case) insofar as they have already been generated.

30

Here, the keys have to be imported in the above-defined sequence. The card can be initialized at a central place, whereby the complete crypto-system computer has

to be initialized and subsequently has to be sent. This is because the WebSentry card deletes the internal memory as soon as it is pulled out of the PCI slot.

5 Preferably, the transport master key can only be imported by at least two security administrators who have locally identified themselves vis-à-vis the crypto-system by means of a SmartCard reader and the associated SmartCards. Due to the importance of the key
10 KTM, this key can only be imported onto the card using the dual control principle. This means that at least two of the PIN-protected SmartCards that were generated in the application case "transport master key" are needed for the import.

15 Since the master transport key KTM can only be loaded onto the card using both SmartCards and since the key use is PIN-protected, it is ensured that this key can only be installed onto the card using the dual control
20 principle. This provides a very high level of protection against the key being breached.

This functionality is provided via the management tool WebSentryManager. This management tool makes it
25 possible to load a so-called local master key (LMK corresponds to the KTM described in this concept) via SmartCards and to store them in a secure storage area of the card.

30 It is especially advantageous to distribute the LMK onto three SmartCards, whereby all three parts are needed in order to import the LMK onto the crypto-

hardware. In this case, three security administrators are needed for importing the master transport key KTM.

The signature key ensures the integrity of the key
5 messages; it can also be used before the import of the
key to determine whether the sender of the key message
is the value transfer center (Postage Point). The
signature key can be generated in different ways,
whereby the examples presented here are especially
10 advantageous. The appertaining key message is stored by
the administrator onto a diskette and, through this
application case, is imported onto a crypto-card of the
payment assurance system that is to be initialized.

15 In order to import the key, the key message stored on
the diskette is decrypted with the master transport key
KTM (PKCS#11 function C_Unwrap, CKM_KEY_WRAP_WEBSENTRY
mechanism). The integrity of the key message is
automatically checked by this routine. If a key with
20 this KeyID already exists, it is subsequently deleted.

For the further transport of the transport key
messages, the server of the central payment assurance
system provides an interface via which the distribution
25 and the subsequent import into the crypto-systems of
the individual local payment assurance systems can be
initiated.

The interface is realized as an RMI service. This
30 service is looked up by means of a naming service
(JNDI). The distribution is carried out via the CWMS
(computerized work management system) used for the
distribution of the P/N list.

If a new distribution job is created, the key message is forwarded to all of the currently registered crypto-systems and a protocol entry is written for each case.

5 The management of the crypto-systems is carried out via an application case of the payment assurance system.

The receipt of new key messages is checked at the individual crypto-systems by an ImportController at

10 regular intervals (as a function of the distribution mechanism). When a new message is received, the application case "import transport key" is automatically launched. The return value of this action is checked. If a negative feedback is received, then

15 the import attempt is repeated up to three times.

If the import is still not possible after three attempts, then a protocol message about the failure is sent to the ZinS central system (once again as a

20 function of the transport mechanism). If the import was successful, a positive protocol message is issued.

The protocol messages are processed centrally via the application case "protocol key processing". The release

25 of the transport key is also triggered accordingly.

The import of the transport keys is either carried out by a security administrator who initializes the crypto-system on site or else this import is triggered

30 automatically by the ImportController of the key distribution when the ImportController receives a new transport key message. The key is preferably imported according to the following process steps:

1. A registration on the card is carried out with the ID and PIN belonging to the KeyImport user.
- 5 2. A hash value is formed for Fields 1 - 5 of the transport key message using the SHA-1 method.
3. The signature key is ascertained that owns the KeyID that was indicated in Field 4.
- 10 4. This key is used to encrypt the hash value formed under Point 2 (CKM_DES3_CBC_PAD mechanism, the initialization vector IV is filled with zeros) and compared to Field 6. If these two values match, the integrity is ensured and it is assured that the message was created by the PC franking system.
- 15 5. The content of Field 5 of the message is decrypted with the KTM (C_UnwrapKey method, CKM_KEY_WRAP_WEBSENTRY mechanism). As a result, the appropriate transport key object is generated automatically and stored on the card. In addition, the mechanism once again implicitly checks the integrity of the key as well as the correct sender.
- 20 6. If a key with the same KeyID already exists on the card, it is deleted.
- 25 7. A hash value is formed for the LABEL attribute of the newly imported key using the SHA-1 method and this is returned together with the KeyID and the positive message as the return value.
- 30

8. The user session is ended.

9. A protocol message is created from the return values and it is sent to the ZinS central system.

5

10. Any failed attempts stored for this KeyID (see variant described below) are reset.

A variant of this application case is that one of the routines or the checking of the MACs fails. In this case, further processing is aborted and a return value is returned containing the KeyID, the error code as well as an error message. For the KeyID, the stored number of failed attempts is increased by one. Once this number has reached three, a protocol message is created from the most recently transferred return value and sent to the central payment assurance system.

In case of manual initiation, the result of the import is additionally displayed on the monitor of the security administrator.

Preferably, steps 2 to 7 run directly in the card software (embedded code). This increases the performance as well as the security against being breached.

For the further transport of the data key messages, the server of the central payment assurance system provides another interface by means of which the distribution and the subsequent import of the data keys into the individual crypto-systems of the local payment assurance systems take place.

The interface is realized as a Session Bean; this service is looked up by means of a naming service (Java Naming and Directory Interface - JNDI).

5

The distribution is carried out via the CWMS (computerized work management system) used for the distribution of the P/N list.

10 If a new distribution job is created, the key message is forwarded to all of the currently registered crypto-systems and a protocol entry is written for each case. The crypto-systems is managed via an allocation system. If a distribution job for a data key is already in
15 circulation for which no feedback has been received at the value transfer center PP, then the acceptance of further distribution jobs for data keys is rejected until the point in time of the feedback.

20 The receipt of new key messages is checked at the individual crypto-systems by an ImportController at regular intervals (as a function of the distribution mechanism). When a new message is received, the application case "import transport key" is launched
25 automatically. The return value of this action is checked. If a negative feedback is received, then the import attempt is repeated up to three times.

If the import is still not possible after three
30 attempts, then a protocol message about the failure is sent to the ZinS central system (once again as a function of the transport mechanism). If the import was successful, a positive protocol message is issued.

The protocol messages are processed centrally via the application case "protocol key processing". The release of the transport key is also triggered by this application case.

The import of the data keys is either carried out by a security administrator who initializes the cryptosystem on site or else this import is triggered automatically by the ImportController of the key distribution when the ImportController receives a new data key message.

The key is imported analogously to the import of the transport key, taking into account the special features of a data key message:

1. A registration on the card is carried out with the ID and PIN belonging to the KeyImport user.
2. A hash value is formed for Fields 1 - 7 of the data key message using the SHA-1 method.
3. The signature key is ascertained that owns the KeyID that was indicated in Field 5.
4. This key is used to encrypt the hash value formed under Point 2 (CKM_DES3_CBC_PAD mechanism, the initialization vector IV is filled with zeros) and compared to Field 8. If these two values match, the integrity is ensured and it is assured that the message was created by the PC franking system.

5. The transport key is ascertained that owns the KeyID that was indicated in Field 6.
6. The content of Field 7 of the message is decrypted
5 with the key ascertained under Point 5 (C_Decrypt method, CKM_DES3_CBC_PAD mechanism, the initialization vector IV is filled up with zeros). The result of the decryption is an interim message.
- 10 7. The content of Field 1 of the interim message is decrypted with the KTM (C_UnwrapKey method, CKM_KEY_WRAP_WEBSENTRY mechanism). As a result, the appropriate data key object is generated automatically and stored on the card. In addition,
15 the mechanism once again implicitly checks the integrity of the key as well as the correct sender.
8. If a key with the same KeyID already exists on the card, it is deleted.
20
9. All of the data keys on the crypto-card are read and checked as to whether the value of the generation counter in the LABEL attribute, byte 1, is the same as the newly imported key. If this is
25 the case, these keys are removed from the card. These are keys that - due to an error in the import into a crypto-system of another local payment assurance system - were not released on the value transfer center PP.
- 30
10. A hash value is formed for bytes 2-65 of the LABEL attribute of the newly imported key using the SHA-1

method and this is returned together with the KeyID and the positive message as the return value.

11. The user session with the crypto-card is ended.

5

12. A protocol message is created from the return values and it is sent to the ZinS central system.

13. Any failed attempts stored for this KeyID (see
10 variant described below) are reset.

A variant of this application case is that one of the routines or the checking of the MACs fails. In this case, further processing is aborted and a return value
15 is returned containing the KeyID, the error code as well as an error message. For the KeyID, the stored number of failed attempts is increased by one. Once this number has reached three, a protocol message is created from the most recently transferred return value
20 and it is sent to the central payment assurance system (ZinS central system).

In case of manual initiation, the result of the import is additionally displayed on the monitor of the
25 security administrator.

Preferably, steps 2 to 10 run directly in the card software (embedded code). This increases the performance as well as the security against being
30 breached (especially of the initialization vectors IV as well as of the KTMs).

A cleanup of the data keys is carried out repeatedly, preferably regularly, on as many crypto-systems of the payment assurance system as possible, preferably on all of them, and serves to delete data keys that are no
5 longer needed.

Procedure of the data key cleanup

1. All of the data keys KD located on the card are
10 ascertained and sorted in ascending order according to their ID (CKA_ID attribute).

2. For each key of this list, the following checking procedures are carried out:

15

I. The successor of the key is determined (next-larger ID).

II. If present, the following is verified:

20

1.whether the CKA_END_DATE attribute of the successor indicating the end of the validity of the predecessor is smaller than the current system date. If this is the case, then
25 the currently processed key of the list is deleted.

25

30

2.whether the generation counter of the successor (byte 1 of the CKA_LABEL attribute) is identical to the generation counter of the currently processed key. If this is the case, then the currently processed key of the list is deleted.

The key processing is preferably protocolled on the server of the central payment assurance system (ZinS central server). The keys that are protocolled by this
5 application case are the transport key and the data key.

For each distribution job, a protocol is drawn up indicating to which active crypto-system the key
10 message was sent. For each system and each distribution job, a separate entry is generated that is first set at the status "sent".

After each successful but also after each unsuccessful
15 key processing, the individual crypto-systems generate a protocol message and send it to the central payment assurance system (ZinS central system). This distribution either takes place via JMS queues or by database replication.

20

In the area of the central payment assurance system, after the messages have been received, they are used to update the above-described protocol entries. For this purpose, the status "processing successful" / "not
25 successful" as well as possibly an error code and the message are stored.

Following the processing of the protocol messages, it is checked whether distribution jobs exist that were
30 successfully incorporated by all of the crypto-systems. If this is the case, then the release of each appertaining key, especially in the area of the value transfer center, is initiated. As soon as a system

reports an error, a corresponding message with a negative status is generated in the area of the value transfer center.

- 5 The fact that the key release has been invoked is noted with the distribution job so that no additional releases are carried out for a given job. As long as the note, however, has not been entered, a new attempt is made at regular intervals to contact the release
10 service.

A special variant is present when, after a period of time to be defined, preferably several days, for example, three days, feedback has not been received yet
15 from all of the crypto-systems. After the expiration of this time, a negative release message is sent to the value transfer center.

Preferably, the key allocation system has a user
20 interface that allows an administrator to check the status of a key distribution job. The following data is then displayed for each distribution job:

- the number of systems to which the distribution
25 message was sent
- the number of systems that have reported back a successful processing
- 30 • the number of systems that have not yet reported back the outcome of the processing

- the number of systems that have reported back an unsuccessful processing

Moreover, a detailed listing can be drawn up in which
5 the current status of each system is displayed.

As an alternative, it is possible to display locally the keys stored on the appertaining card.

10 It is advantageous for all of the keys for which distribution jobs were generated to be archived in the area of the central payment assurance system. Preferably, no archiving is carried out on the local systems. There, the keys are stored in the non-volatile
15 memory of the card. Only key messages that were also released are archived.

The key restoration of transport keys and data keys can be launched centrally for a specific crypto-system. In
20 this case, the current keys from the archive are ascertained and sent to the specific crypto-system. For this purpose, protocol messages are likewise generated. With this type of key distribution, only the release message is absent.

25

If the master transport key KTM is lost, then the corresponding crypto-system either has to be sent to the security administrators for new initialization or else the security administrators have to initialize
30 each system on site.

The master transport key KTM is reliably secured against being compromised through special security

measures of the WebSentry card and through the distribution onto several SmartCards as well as through the multi-stage key system.

- 5 Nevertheless, if an exchange of the master transport key is to take place, a new master transport key KTM as well as new signature keys, transport keys and data keys have to be generated.
- 10 These then have to be imported into all of the crypto-systems of the local payment assurance systems. This involves more effort since either all of the crypto-systems have to be transported to the central administration site and back again, or else the
- 15 security administrators have to travel to all of the sites of the local payment assurance systems. Therefore, the use of the security mechanisms according to the invention for the master transport key KTM is especially advantageous.
- 20
- The prior master transport key KTM remains on the card as a so-called "dormant LMK" and has to be appropriately deleted by the security administrator.
- 25 The key handling and the decryption should be present on the crypto-card as embedded code. In this manner, not only a greater degree of security is achieved but also a performance improvement of the crypto-system is expected.

30

Preferably, the crypto-cards contain the standard PKCS#11 functions listed below:

- C_CloseSession
- C_GetSlotList
- C_Init
- C_Initialize
- 5 • C_Login
- C_Logout
- C_OpenSession

along with the presented extensions. Moreover,
10 permanently stored functions should not contain any further extensions of third parties and the extensions required here are exclusively integrated as functions for crypto-cards of the payment assurance system.

15 In order to designate the embedded key-handling methods of the PKCS#11 DLL, these methods are given a prefix. This prefix is defined as **CE_**. In this case, CE stands for **C**rypto **E**xtension.

20 Each embedded method supplies a return value of the type **CK_RV**, which is defined as a fixed component of the pkcs11.h include file. It is advantageous that, during the implementation of the embedded methods, additional error return values are defined and these
25 are provided in a C++ header file for integration. This measure is advantageous in that, through an invocation of an embedded method, various Pkcs11 methods can be invoked which are nested on the hardware. Another advantage here is the completely newly established key-
30 handling within the software of the crypto-cards.

Example for illustrating the method syntax:

CK_RV = CE_MethodName (parameter list)

Within the parameter list, parameters that have to be filled with a result are transferred by means of call
5 by reference. The method name is formed from meaningful word combinations that provide a good idea of what the method does.

The word selection is made in such a way as to allow an
10 association of meaning contents, for example, through the use of English technical terms.

The introduction of two enumeration types serves to verify the functionality of different embedded methods.
15 A distinction is made between the enumeration type for key types and key attributes.

CE_EnumKey = { CE_KT, CE_DT, CE_SE, CE_KA }

20 CE_KA assumes a special position. It is not a key but rather the set of all keys. If this EnumElement is indicated, then the methods implement a functionality that relates to all of the keys contained in the card.

25 CE_EnumKeyAttribute = { CE_ID, CE_LABEL, CE_STARTDATE, CE_ENDDATE }

The necessary defines are to be taken over into the pkcs11.h file. The defined extensions can be located in
30 a separate header file that are enclosed in the pkcs11.h file. Various mechanisms are possible to implement the embedded methods.

In the cryptographic environment, a method is defined that autonomously carries out all of the relevant verifications with the PKCS#11 methods that are available to it.

5

```
CK_RV CE_Decrypt (CK_SESSION_HANDLE session,  
CK_BYTE [ ] message, CK_BYTE* postageID  
CK_BBOOL bOk)
```

10 **Function description:**

The embedded cryptography method receives a 57-byte long data in the parameter message and this data corresponds to the matrix code of the postage indicium.

15 The counting in the following explanation of the work steps starts at one.

1st Step: formation of the initialization vector IV

20 As the initialization vector, the first two bytes are filled with zeros and then bytes f6 - f10 plus byte f14 are appended to the matrix code.

2nd Step: determination of the KD to be used

25 The key phase indicator is contained in byte f14; it indicates which key is to be used. The key phase indicator is stored in the CKA_ID key attribute and thus unambiguously identifies the key. The key handling described further below should allow an efficient finding of the key.

30

3rd Step: decryption of the contained encrypted message
The mechanism employed in CK_MECHANISM is CKM_DES3_CBC. Bytes f15 - f38 are decrypted, that

is 24 bytes, the first 12 bytes of the decrypted result are output by the parameter postageID. Once the decryption has been carried out successfully, the process continues with Step 4, otherwise with
5 Step 5.

4th Step: formation and cleanup of the hash value

A specially constructed 77-byte large data block serves as the basis for the formation of the hash
10 value of the date.

Bytes 1 to 53: corresponding to the first 53 bytes of the matrix code

Bytes 54 to 65: corresponding to the first 12 bytes
15 of the current, unencrypted message part (PostageID)

Bytes 66 to 77: corresponding to the last 12 bytes of the current, unencrypted message part

20 The hash value is formed using SHA-1 and, after this procedure, the first four bytes have to match bytes f54 - f57. If this is not the case, then the date is not valid. If an error occurs during the execution of the hashing or if there is a
25 discrepancy in the hash value, then the procedure is as in Step 5.

5th Step: return of the success indicator

The parameter bOk is filled with "true" if all of
30 the work steps were successful, and with "false" if the hash value cleanup has revealed a discrepancy or if one of the Pkcs#11 methods has caused an error. The return value has to accordingly be

filled with the error message or with CKR_OK if no error has occurred.

5 CK_RV DE_VerifyMsg (CK_SESSION_HANDLE session,
CK_BYTE [] message, int length,
CK_BBOOL bOk)

A general data block message for the verification procedure is shown below:

10

Data block for the verification procedure.				
Item	Length	Byte no.	Designation	Description
1	2	f1 - f2	MsgType	Identification of the key message, constant 'KT' or 'KD'
2	1	f3	Version	Version of the structure of the message, 1 byte starting with the value '01'
3	2	f4 - f5	KeyGeneration	Generation counter in plain text, e.g. '01'
4	1	f6	KeyID	ID of the data key in plain text
5	4	f7 - f11	KeyID	ID of the transport key in plain text
6	4	f12 - f16	SigKeyID	ID of the key used for the signature
7	4	f17 - f21	KTID	ID of the transport key used for the encryption of the interim message (see Item 2)
8	n-22	f22 - f _n	TranspKeyEncr	KT: result of the

			ypt ----- ----- ----- ----- HelperMsgEncr ypt	encryption (wrap) of the transport key ----- ----- ----- ----- KD: result of the encryption of the interim message
9	24	$f_{n+1} - f_{n+24}$	MAC	KT: MAC for the key message; is formed in that an SHA-1 hash value is formed for the Fields 1+2+5+6+8 of the message, and this hash value is encrypted with the signature key (CKM_DES3_CBC_PAD mechanism, the IV is filled up with zeros; ID see item 6). ----- ----- ----- KD: MAC for the key message; is formed in that an SHA-1 hash value is formed for the Fields 1+2+4+3+6+7+8 of the message, and this hash value is encrypted with the signature key (CKM_DES3_CBC_PAD mechanism, the IV is filled up with zeros; ID see item 6).

Unused fields are filled up with zeros. The mode of operation of the method is determined via the parameter MsgType.

- 5 A generated data block is transferred in message for the MsgType KT and KD. The data block is filled up with each of the received messages.

The function allocation CE_VerifyMsg KT for the MsgType
10 KT receives the complete transport key message in the MESSAGE attribute, and its length in the LENGTH attribute. This embedded message serves to ensure the integrity of the transport key message at the recipient. In order to actuate the verification, the
15 following steps have to be performed:

1st Step: formation of the initialization vector IV
The initialization vector is filled with zeros.

20 2nd Step: decryption of the received encrypted message
The mechanism used in CK_MECHANISM is CKM_DES3_CBC_PAD. The bytes of the variable MAC field are decrypted. If an error occurs during the decoding, then the procedure is as in Step 4.

25

3rd Step: cleanup of the hash value

In order to form the hash value of the date, a hash is formed from the Fields 1+2+5+6+8 of the transport key message and it is compared to the
30 hash of Step 2. The hash value is formed using SHA-1. If the hash values are not identical, then the date is not valid. If an error occurs during the execution of the hashing or if there is a

discrepancy in the hash value, then the procedure is as in Step 4.

4th Step: return of the success indicator

5 The parameter bOk is filled with "true" if all of the work steps were successful, and with "false" if the hash value cleanup has revealed a discrepancy or if one of the Pkcs#11 methods has caused an error. The return value has to be filled
10 accordingly with the error message or with CKR_OK if no error has occurred.

After the function allocation CE_VerifyMsg for the MsgType KD, the complete data key message is
15 transferred in the MESSAGE attribute, and its length in the LENGTH attribute. This serves to ensure the integrity of the transport key message at the recipient. In order to actuate the verification, the following steps have to be performed:

20

1st Step: formation of the initialization vector IV
The initialization vector is filled with zeros.

2nd Step: decryption of the received encrypted message

25 The mechanism used in CK_MECHANISM is CKM_DES3_CBC_PAD. The bytes of the variable MAC field are decrypted. If an error occurs during the decoding, then the procedure is as in Step 4.

30 3rd Step: cleanup of the hash value

In order to form the hash value of the data, a hash is formed from the Fields 1+2+4+3+6+7+8 of the data key message and it is compared to the hash of Step

2. The hash value is formed using is carried out using SHA-1. If the hash values are not identical, then the date is not valid. If an error occurs during the execution of the hashing or if there is a discrepancy in the hash value, then the procedure is as in Step 4.

4th Step: return of the success indicator
The parameter bOk is filled with "true" if all of the work steps were successful, and with "false" if the hash value cleanup has revealed a discrepancy or if one of the Pkcs#11 methods has caused an error. The return value has to be filled accordingly with the error message or with CKR_OK if no error has occurred.

These embedded key-handling methods should comprise the key import of a wrapped key and an efficient key management. Keys of the type KS, KD and KT should be imported.

```
CK_RV CE_ImportKey (CK_SESSION_HANDLE session,  
    CK_BYTE_PTR data,  
    CK_ULONG length, CK_BYTE* HashValue,  
    CE_EnumKey KeyType,  
    CK_CHAR [ ] KeyKTID)
```

The function allocation CE_ImportKey is preferably carried out as described below:

The CKM_KEY_WRAP_WEBSENTRY mechanism is used for the wrapping. Accordingly, the same mechanism is needed for the unwrapping. The key obtained is imported into the

crypto-hardware by the unwrapping, whereby a key that is imported a second time, that is to say, with the same key phase, overwrites the old key.

- 5 The wrapped key is transferred to the DATA parameter and its length in the LENGTH parameter. The length of the key depends on the filling of the key attributes. The key type is verified by the KeyType parameter and treated accordingly.

10

The key is then taken into the key management held in the cache operation and, if applicable, the duplicate predecessor is deleted.

- 15 The DT is decrypted by the KT that is identified by the KeyKTID parameter; for all of the other key types, this parameter is not taken into consideration and is filled with zeros.

- 20 The dependence of the CKA_END_DATE attribute is important. It indicates the end date of the key predecessor.

- 25 The key attribute of the imported key contains the random number by means of which the hash value is to be formed using SHA-1. The hash value is returned in the HashValue parameter of the embedded method. The hash value is needed for the key acknowledgement message.

- 30 In case of error during the unwrap mechanism or the hash formation, the appropriate error code is output in the return value, otherwise CKR_OK.

```
CK_RV CE_GetKeyCount (CK_SESSION_HANDLE session,  
CE_EnumKey KeyType,  
int* counter)
```

- 5 The function allocation CE_GetKeyCount shows how many keys of each key type are registered on the card in the key management held in the cache operation. In this manner, the key attributes can be read out in conjunction with the method described below.

10

This method is defined as follows:

```
CK_RV CE_GetAttribute (CK_SESSION_HANDLE session,  
CE_EnumKey KeyType,  
15 CE_EnumKeyAttribute KeyAttribute,  
int pos,  
CK_BYTE [ ] * attribute,  
int* length)
```

- 20 The key type is determined via the KeyType parameter and thus also the table that is to be read from when the different key types on the crypto-hardware are recorded in different lists according to the type of key.

25

The KeyAttribute parameter specifies the attribute that is to be read out and the ITEM parameter indicates the starting point within a table; its maximum value first has to be acquired using the CE_GetKeyCount method for
30 all keys or else for one key type. When the CKA_END_DATE attribute is output, it must be noted that the last current key is theoretically infinitely valid (until a new key of the same type is imported) and, in

its CKA_END_DATE attribute, contains the end date for the predecessor key of the same key type, whereby the CKA_END_DATE of this indicated key is output.

5 The attributes for a date have a fixed length of 8 bytes whereas the CSK_ID and CKA_LABEL attributes have a fixed length of 128 bytes. If the attributes within the keys for these two parameters should be shorter than 128 bytes, then the remaining bytes up to 128 are
10 filled up with zeros. Thus, the user can always provide sufficient memory for the methods in question. In case the user provides a buffer that is too small, the information is trimmed and a message to this effect is transmitted via CK_RV.

15

```
CK_RV CE_DeleteExpiredKeys (CK_SESSION_HANDLE session,  
    CE_EnumKey KeyType,  
    int* counter)
```

20 The function allocation CE_DeleteExpiredKeys searches the card for expired keys and deletes them from the card. Expired keys can be identified by the fact that the system date is older than the CKA_END_DATE of the successor key, see under 2.5.8; this also applies to KT
25 and KS. Selective deletion is made possible through the EnumType and the entire card can be deleted by means of CE_KA (only the LMK is retained). It is important that this method must not be allowed to become active when a key import is carried out. This is preferably
30 controlled in the embedded code and indicated with an appropriate return value. Thanks to this measure, any indeterminate side effects in the internal key management are avoided.

The interface between the payment assurance system and the value transfer center is preferably as narrow as possible in order to rule out manipulation possibilities through side channels.

The structure of the interface between the value transfer center (Postage Point) and the central payment assurance system is shown in Figure 3.

10

The central payment assurance system provides an interface for distributing keys by means of which keys generated in the KeyManagement component of the value transfer center (Postage Point) can be distributed to the crypto-systems of the payment assurance systems.

The value transfer center provides the payment assurance system with a key release interface by means of which the payment assurance system can release the key after the successful distribution and processing of said key.

Since mainly Java is used in both projects, an application interface realization using Session Beans is recommended. In order to uncouple the two systems, the services should be looked up by a naming service using JNDI.

A Session Bean provides the necessary functionality for importing the key data into the ZinS central system. The entire communication is shown in Figure 4.

Figure 4 shows process steps for integration of a data key into the central payment assurance system (ZinS central).

- 5 The process routine ImportKey transfers the data key set to the ZinS central.

Depending on the use of ASN.1 messages, this process routine prepares the received message and causes the
10 message to be stored in the database. The process routine ImportKey then initiates the distribution of the key data to the ZinS local systems by means of CWMS.

- 15 The sequence of importing the data into the database and subsequent distribution of the messages should be observed. This ensures that the received data is secured before work is done to it or with it. The advantage of this approach is that events in the realm
20 of errors can be better reconstructed and, in case of a data loss, access can be gained to the database if need be.

The parameters of the InsertKeyData method are not yet
25 specified since at this point in time, it is not yet clear whether the ASN.1 format is supported. However, the method will have to be equipped with two parameters in order to also be able to sent a detailed message to the Postage Point.

30

In ZinS central, the distribution method of the key allocation system is responsible for the following:

1. archiving the key message on the ZinS central server.
2. distributing the key messages from the Postage
5 Point to the individual mail centers by means of the CWMS interface provided by Vibris. The structure and use of the CWMS service can be found in the manual of this interface.
- 10 3. after the distribution and import into the individual mail centers has been completed, an appertaining return message is generated.

The data is transmitted from the value transfer center
15 (Postage Point) in the ASN.1 format. The appertaining return response is likewise expected in the ASN.1 format. Instead of this format, however, other formats can, of course, also be used. The particular formats are adapted for use by a suitable parser.

20

Preferred data formats in ASN.1 are the following:

Distribution message for the transport key

```
TransportKeyMessage ::= SEQUENCE
{
25  MsgType OCTET STRING, ( fix 'KT' )
    Version OCTET STRING, ( 0x01 )
    KeyID OCTET STRING, ( CKA_ID )
    SigKeyID OCTET STRING, ( CKA_ID of SignatureKey used
    for MAC )
30  TransKeyEncrypt OCTET STRING, ( TransportKey wrapped
    with LMK )
    MAC OCTET STRING ( MAC over all previous elements )
}
```

Distribution message for the data key

PostageKeyMessage ::= SEQUENCE

{

- 5 MsgType OCTET STRING, (fix 'KD')
 Version OCTET STRING, (0x01)
 KeyID OCTET STRING, (CKA_ID)
 KeyGeneration OCTET STRING, (0x01 ascending)
 SigKeyID OCTET STRING, (CKA_ID of SignatureKey used
10 for MAC)
 TransKeyID OCTET STRING, (CKA_ID of TransportKey)
 DataKeyEncrypted OCTET STRING, (PostageKey wrapped
 with LMK
 and
15 encrypted with TransportKey)
 MAC OCTET STRING (MAC over all previous elements)
 }

Also see section 2.4.6.

20 **Release message**

KeyExchangeReceipt ::= SEQUENCE

{

- KeyID OCTET STRING, (CKA_ID of received key)
 KeyLabelHash OCTET STRING, (SHA-1 hash of CKA_LABEL of
25 received key)
 State BOOLEAN, (TRUE/FALSE to enable/dismiss key)
 Message OCTET STRING (Description of success/failure)
 }

- 30 The data structures can have a different set-up.
 However, a set-up according to the embodiments
 presented is especially advantageous since they allow
 the transmission of all of the information that is to

be taken into consideration and they account for a small data transmission effort. In particular, the data is transmitted via CWMS to the local payment assurance systems which are preferably located in mail centers of
5 the postal service provider.

When the ASN.1 format is used, the data would first have to be parsed into the internal key message, which would be dispensed with if the messages defined in this
10 document were used directly.

The data packets received through the CWMS correspond to the key messages previously defined in this document. These messages are then subjected to the
15 VerifyMsg method of the embedded code after they have first been adapted to the data table above title "Attributes of the data key KD". After the verification, either the import of the key is started via the embedded code or else an appropriate error
20 message is generated. Compare key messages and pages 12 to 14 in terms of the import data structure of the CE_ImportKey embedded code method. The deletion of the old keys as well as the updating are carried out automatically by the above-described embedded code
25 methods.

Every day, the CE_DeleteExpiredKeys method is invoked which removes the expired keys from the crypto-hardware daily, whenever this is necessary. During the import,
30 the CA_ImportKey method ensures that duplicate keys are deleted and that newer ones replace them.

The embedded code methods are bound to the application by the Java class CryptoAdapters. CryptoAdapters offer all functions (similarly named) that the embedded code and other PKCS#11 methods offer.

5

By means of JNI, a DLL (CryptoAdapter.DLL) is used that has statically linked the DLL provided by Zaxus. The security risk is further minimized through the static linking.

10

The C++ implementation of the JNI interface also provides error handling for each requested session; in this context, see Stage 3 under "multithreading" in the PCFM Design Document. The worker concept is supported in that the crypto-hardware is initialized in the multithreading mode and, after the login, each worker gets its own session (getSession, C_GetSession), as a result of which this worker is registered in the DLL and error handling is established specifically for the worker. The mainstream of the DLL is addressed by the login and likewise has its own error handling for the execution of Pkcs#11 methods and the embedded methods.

15
20

Figure 6 shows an overview of the implementation. Preferably, the key list and the code that is provided by the embedded methods are removed. Otherwise, the set-up of the structures must be selected to be identical.

25
30 Connection of the DLL

A preferred encapsulation and an advantageous error handling are shown in Figure 7 by way of an example.

An implementation of the key lists is not needed if the key lists are held completely in the embedded code of the crypto-hardware. The GetAttribute methods decrease to the one single invocation of the embedded
5 CE_GetAttribute method; by the same token, the invocations for the import and its implementation are reduced since this is automatically carried out by the embedded code after the transfer of the necessary data.

10 An expanded error constant list is provided in the embedded code.

The key messages from the Postage Point are stored in ZinS central in a database in order to allow a later
15 replacement of a defective local payment assurance system. First of all, it is required that the message as such be incorporated into the database and secondly, that administrative information be received.

20 Consequently, the following database tables are needed.

TransportKeyData	
SDItemNo	INT
Key data record	See table on page 29

Key data stores the key message in the way in which it was received by the Postage Point; in case of a failure
25 of a local payment assurance system, the archiving of the key message serves to coordinate the failed local payment assurance system with the other remaining local payment assurance systems by importing the archived key. Before the storage, the message has to be parsed
30 by ASN.1 in order to transmit it to the local systems.

The shared form of the data records is preferred as the storage modality; this corresponds to the table that describes the data block that is used for the CE_VerifyMessage embedded method.

5

TransportKeyManagement	
SNItemNo	Number
ReceiptDate	DATE (/ timestamp)
CompletionDate	DATE (/ timestamp)
DistributionDate	DATE (/ timestamp)
EndStatus	VARCHAR(20)
StatusText	VARCHAR(20)

This data record has a central significance. First of all, it serves to evaluate the feedback received from the local payment assurance system in the mail centers, preferably all of the mail centers that are integrated into the mailing system of the postal service provider, as well as to conduct a capable error analysis and to alarm of the system operator whenever the time is exceeded during the distribution procedure.

15

By the same token, this table involves administrative information processing. This makes it possible to use the entry SNItemNo to allocate and to evaluate the appertaining TransportKeyData table and the TransportKeyReplayMessage of the individual (83) mail centers, if applicable,.

20

TransportKeyReplayMessage	
SNItemNo	Number

MailCenterNo	Number
MessageDate	DATE (/ timestamp)
MessageText	VARCHAR(520)

In this table, the individual key response messages of the local payment assurance system are archived in the mail centers of the postal service provider as a
5 function of the import procedure.

Claims:

1. A method for verifying the authenticity of a
5 postage indicium generated using a franking key and
applied onto a mailpiece, whereby cryptographic
information contained in the postage indicium is
decrypted and used for verifying the authenticity
of the postage indicium,
10 **characterized in that**
a data key (KD) is generated and transmitted from a
central payment assurance system to local payment
assurance systems.
- 15 2. The method according to Claim 1,
characterized in that
the local payment assurance systems import the data
key and transmit a result of the import to the
central payment assurance systems (ZinS central
20 system).
3. The method according to Claim 2,
characterized in that
the result of the import is transmitted as a data
25 record.
4. The method according to Claim 3,
characterized in that
the data record contains a key.
30
5. The method according to one or more of Claims 2 to
4,
characterized in that

the central payment assurance system (ZinS central system) checks the result of the import and transmits it to a value transfer center (Postage Point).

5

6. The method according to Claim 5,
characterized in that
the result is checked by decryption of the key.

- 10 7. The method according to one or more of Claims 1 to 6,

characterized in that,
once the data key has been successfully imported into essentially all local payment assurance systems (ZinS local), the data key is released at
15 the value transfer center (Postage Point) and is subsequently used for producing new postage amounts.

- 20 8. The method according to one or more of Claims 1 to 7,

characterized in that,
the data key is a symmetrical key.

- 25 9. The method according to one or more of the preceding claims,

characterized in that
the new data key is transmitted from the value transfer center (Postage Point) to the central
30 payment assurance system.

10. The method according to Claim 9,
characterized in that

the value transfer center encrypts the data key with a transport key (KT).

11. The method according to Claim 10,
5 **characterized in that**
the transport key is encrypted with a master transport key (KTM).
12. The method according to one or more of Claims 9 to
10 11,
 characterized in that
the data key is generated in the area of the value transfer center.
13. The method according to one or more of Claims 9 to
15 12,
 characterized in that
the data key is provided with key identification information.
- 20 14. The method according to Claim 13,
 characterized in that
the key identification information is transmitted in encrypted form.
- 25 15. The method according to one or more of Claims 9 to
14,
 characterized in that
the data key is provided with a generation counter
30 and/or is transmitted together with the generation counter.
16. The method according to Claim 15,

characterized in that

the data key is transmitted to a cryptographic element, especially to a crypto-card, and in that, as soon as the cryptographic element receives the data key, it deletes all of the data keys that have the same generation counter value as the transmitted data key.

17. The method according to one or more of Claims 9 to 16,

characterized in that

the data key is provided with an end date for the preceding data key and/or it is transmitted with the end date.

15

18. The method according to Claim 17,

characterized in that

the franking key is transmitted to a cryptographic element and in that the cryptographic element checks whether other data keys have an end date and whether the end date stored for the successor data key is older than a date stored in the payment assurance system.

19. The method according to one or more of the preceding claims,

characterized in that

the data key or at least part of the data key forms a component of a franking key for the generation of postage indicia.