RZ 3113 (# 93159) 03/08/99
Computer Science/Mathematics 14 pages

Research Report

E-Cash Withdrawal using Mobile Telephony

C. Binding, S. Hild, L. O'Connor

IBM Research Division
Zurich Research Laboratory
8803 Rüschlikon
Switzerland

IBM Research Division
Almaden · Austin · Beijing · Haifa · T.J. Watson · Tokyo · Zurich

E-Cash Withdrawal using Mobile Telephony

C. Binding, S. Hild, L. O'Connor

IBM Research Division, Zurich Research Laboratory, 8803 Rüschlikon, Switzerland

Abstract

This paper proposes a protocol for adding value to an electric purse, suitable for implementation over a wireless network. The protocol is developed with respect to the Deutsche Geldkarte electronic purse and its *withdrawal* (or value loading) function. We propose extensions to the architecture to allow secure and convenient e-cash withdrawal in a wireless application environment such as GSM/SMS or the Wireless Application Protocol (WAP) where open wireless networks are used for communication purposes. We expect that other electronic purse schemes can be adapted to this protocol, assuming the Deutsche Geldkarte scheme to be sufficiently generic.

1 Introduction

Various electronic cash systems have been trialled or are being introduced on a wide scale. They are intended for payment of small monetary amounts of up to the equivalent of 200.- US Dollars and are based on usage of smart cards to store some monetary value.

In Switzerland, all EC-direct debit cards issued since 1997 by banks are equipped with the *CASH* system based on the Proton technology from Banksys (Belgium) [9]. VISA has been experimenting with VisaCash in several locations, most noteworthy New York city [19]. MasterCard has trialled its MasterCard Cash system in Australia [8]. The payment card industry is currently elaborating the Common Specification for E-Purse system (CEPS), showing the long-term interest in such technology [1]. All these systems are based on smart cards acting as portable devices capable of securely storing value and other parameters.

Germany's *Deutsche Geldkarte* has been piloted in various communities and will be issued country-wide in 1998 [12, 19]. Its architecture is shown in Fig. 1 and may be considered a generic architecture for e-cash systems based on the maintenance of a card counter (in conjunction with some currency denomination and user information) to reflect the balance of the user's purse. The main entities of the architecture are the user's smart card, an automatic teller machine (ATM), the point-of-sale (PoS) terminal, and the underlying banking infrastructure. A secure, private (or virtual private) network is assumed between the entities: the user's bank where he has an account and where the purse's shadow account is maintained, the clearing center which settles the e-cash related transactions, the merchant's bank which maintains the merchant's account, the PoS terminal, and the ATM.

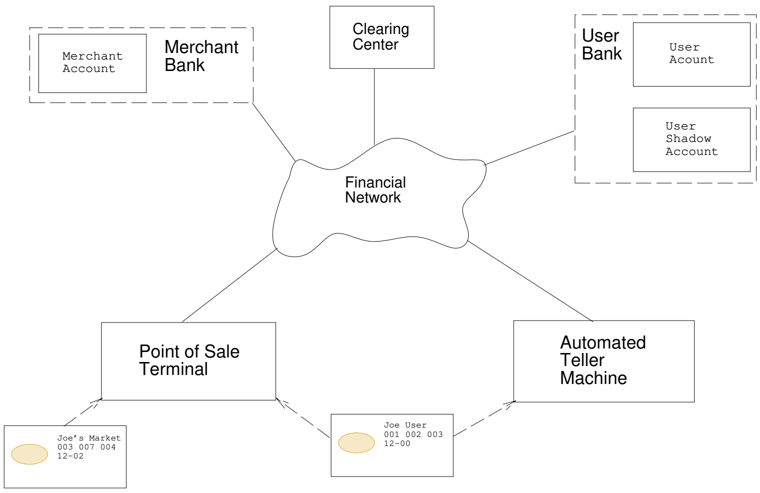

Figure 1: A generic e-Cash System Architecture

The user withdraws (respectively deposits) e-cash at the ATM via an on-line transaction which involves the clearing center, the user's bank account and the purse related shadow account. The user identifies himself against the smartcard by entering a PIN code during these transactions. During a withdrawal operation, the user's account is debited the given amount which is credited to the purse's shadow account and the purse itself; a deposit reverses the value flow.

Payment is done by transferring e-cash from the user's card onto a merchant's payment terminal. A special merchant card is used in this transaction as an authentication device. During payment, the user does not have to indicate his PIN: he is only prompted for the desired amount, indicates "OK" and withdraws the cash card from the merchant's PoS terminal. The purse's shadow account is not queried on-line. The settlement between the user and merchant account occurs during end-of-day processing. At that time, the merchant transmits all e-cash transactions of the day to the clearing center which then settles the accounts by transferring value from the user's shadow account to the merchant's account.

Currently, the security of the Geldkarte scheme is guaranteed through various authentication and encryption techniques:

- *PIN identification*: only the authenticated, legitimate owner of the Geldkarte card can unlock access to the card.
- *Card and ATM verification*: the card and ATM authenticate each other by running a challenge-and-response authentication protocol, preventing the use of forged cards or fake terminals within the system.
- *Transaction management*: the transaction (withdrawal or deposit) is controlled by the trusted and secure ATM; the user can only interrupt the protocol executed by the ATM at pre-defined control points.
- *Private networks*: the connection between both, the ATM and the merchant's PoS terminal into the bank infrastructure is secured using encryption of the existing financial network.

One of the fundamental assumptions of the system's security is that the ATM used for withdrawal is trusted. In particular, various master keys from the issuing bank are kept by the ATM. In addition, sensitive user information from the card, including the PIN, is handled by trusted I/O hardware. Our proposals center on the amount of trust that the user and the banking institution ascribe to a given mobile telephone. We note that it is impossible to migrate the Deutsche Geldkarte protocol unchanged to a mobile phone without placing an unreasonable amount of trust in the phone.

The Geldkarte scheme is considered an *off-line* transaction since at the time of payment the user's shadow account – the binding reference – is not queried. Unless the system has been tampered with, the value stored on the card never exceeds the shadow account value since the card account can only be augmented via a secure transaction involving the ATM. Thus, there is no risk of overdrawing the user's real account.

Although Deutsche Geldkarte, and e-cash in general, has not yet found wide-spread acceptance, we expect its popularity to increase in the future as more PoS terminals become e-cash compliant. For example, parking meters, phone boxes, vending machines, or cafeteria cash registers are ideal targets for e-cash based schemes. We believe that one factor which will greatly influence the acceptance of such e-cash schemes is the pervasive availability of the ATM function, or the ability to load value onto the card at arbitrary times and places. Currently, the user cannot download additional value into his card at the PoS terminal, or in fact any other terminal except the (trusted) ATM. This inconvenience could be overcome if a mobile telephone[1] can be used to provide the ATM function. Then, for example, payment at a parking meter would always be possible even if the user does not have the exact monetary change or a sufficiently loaded e-cash card.

Migrating the ATM functionality to a mobile telephone then requires altering various security assumptions of the Deutsche Geldkarte environment, including:

- *Untrusted ATM*: mobile telephones are currently not tamper-proof devices. Engineering a reasonably tamper-proof end-user device is technically feasible, but involves high production costs and thus may not be realistic.
- *Open, insecure network*: a fundamental security aspect of e-cash schemes is secure connections (often at the physical layer) as provided within the currently deployed banking infrastructure. Mobile communications do not provide an environment with end-to-end security and it is the responsibility of the mobile application to provide this service if required.
- *Transaction interruptions*: we envision e-cash cards to be easily inserted into and removed from the mobile telephone. Given the small and decreasing size of handsets, the mobile telephone can not "swallow" the e-cash card as ATMs currently do. Thus the user can, at any point in time, interrupt an ongoing transaction by withdrawing the e-cash card from the mobile telephone.

[1]Here we use a mobile phone as an example of a mobile device.

In the remainder of this paper, we examine the Deutsche Geldkarte protocol in detail (section 2) and propose protocol and application extensions that may be used to migrate the withdrawal (value loading) application to a mobile handset (section 3). In the present paper we do not consider migrating the payment application. The reason for this is that we still envisage e-cash payments to be made using PoS terminals and the usefulness of e-cash appears more enhanced by making the withdrawal payment mobile as opposed to the payment protocol. Section 4 relates our proposal to some mobile execution platforms that can be used to implement our proposals. Finally, section 5 discusses our findings.

2 Details of the Deutsche Geldkarte Scheme

This section provides a detailed description of the Deutsche Geldkarte's *withdrawal* application to illustrate currently deployed e-cash technology [2, 5]. The download of e-cash onto the purse is divided into two phases: preparation to withdraw and the withdrawal itself, which are described further below. In what follows we will simply refer to the Deutsche Geldkarte smart card as the "card" or the "Geldkarte", and the automated teller machine (ATM) as the "terminal".

2.1 Key management

We begin by examining the underlying cryptographic mechanisms used by Geldkarte to provide privacy, authentication and integrity services, and the associated keys that realize these services. Cryptographic operations are based on either DES or triple-DES [10], and integrity checks are also derived from DES based message authentication codes (MAC) [10]. Security services are derived from a collection of secret parameters (keys) pre-installed in the banking infrastructure and the Geldkarte. In particular, a series of *master keys* are selected and fixed for some time, from which all other keys, such as session keys, are derived. As explained below, a master key is also known as a key-generating key (KGK), and the main master keys required for the withdrawal application are KGK_{INFO} (card/terminal authentication), KGK_{PIN} (PIN verification) and KGK_{LD} (card/bank authentication during money loading) [3]. Each key-generating key is 16 bytes in length.

The master directory of the card stores card-specific keys derived from some of the above master keys, identity information of the card holder in the elementary file *EF_ID*[2], account information data, random number generation data, the card's software version information, etc. An e-purse specific sub-directory *DF_PURSE*[3] stores additional key material, authentication data, PIN data, the purse's value, and protocol log information.

For each specific functionality, a particular card and function-dependent key is used. For example, the card specific key K_{LD} is used for e-cash withdrawal (or loading), and it is derived from the KGK_{LD} master key during card issuance. The derivation of a specific key from a master key is based on data contained in the card's elementary file *EF_ID* and a publically known initial value IV. The actual derivation of a card-specific key KK is given by

$$KK = P(E_{KGK}^{DES-ECB}(H(IV, CID))) \tag{1}$$

where

- KK: is a card-specific function key.
- P: is a parity adjustment function ensuring that each key byte is of uneven parity (required for DES keys) ;
- $E_{KGK}^{DES-ECB}$: represents the triple-DES decryption in ECB mode using key KGK [10].

[2] In smart card technology, an *elementary file* represents a data file stored in the smart card resident file system.
[3] In smart card technology, a *dedicated file* represents a directory entry in the smart card resident file system.

- H: is the hash function defined in ISO 10118-2 (Matyas-Meyer-Oseas hash function [10]) generating a 16-byte hash value from input values whose lengths are an integer multiple of eight.
- *CID*: is the zero-padded content of the card's elementary file *EF_ID* such that the length of *CID* is an integer multiple of eight.

2.2 Preparation to for the Withdrawal Protocol

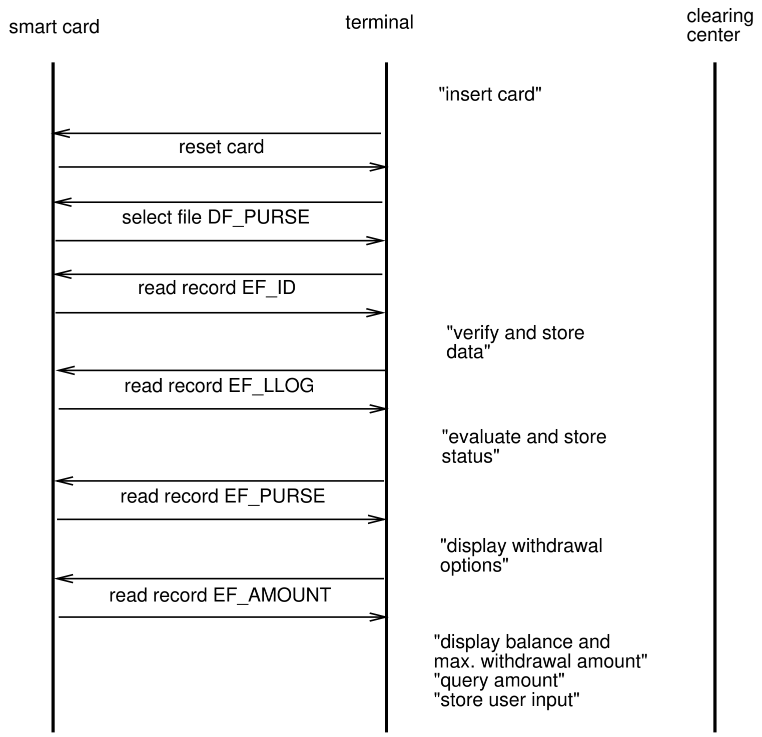

Figure 2: Withdrawal Preparation

The preliminary phase of the withdrawal protocol is illustrated in Fig. 2 and involves the following steps which are initiated by and executed under the control of the terminal:

1. Generation of a *reset card* command to which the card answers with its *answer to reset.*
2. *Select*ion of the purse application, represented through the dedicated file *DF_PURSE.*
3. Obtain the card's *EF_ID*, which contains the identity of the user and his card. It is verified by the terminal using a partial checksum for data integrity and value checking for validity dates, currency, country code, etc. and stored for latter usage.
4. The transaction log stored on the card in elementary file *EF_LLOG* is read. This controls whether a previous withdrawal transaction is resumed or a new transaction shall be started[4].
5. Information regarding the type of the purse and how value can be loaded onto the purse is read off the card.[5]
6. The last step reads the elementary file *EF_AMOUNT* which contains the information on the purse's currently stored value and the maximal amount which can be downloaded onto the purse.

[4]For the remainder of our discussion we shall assume that a new transaction is started.

[5]For the remainder of our discussion, we assume that value is debited from the user's bank account and not via some other value transfer mechanism which is also a possibility in the Geldkarte architecture.

We have omitted the discussion of error handling here for sake of clarity of the presentation.

2.3 The Withdrawal Protocol

The transaction to withdraw cash from the user's bank account and load it onto his e-purse implies communication between the terminal and the clearing center. The withdrawal transaction of the Geldkarte uses a protocol which requires one round-trip message between the terminal and clearing center. It is illustrated in Fig. 3. The following steps are executed during the transaction executed under control of the terminal. We illustrate the withdrawal protocol including the optional external authentication and secure messaging[6] for the user's PIN between terminal and card. Otherwise, the protocol starts with step 6.

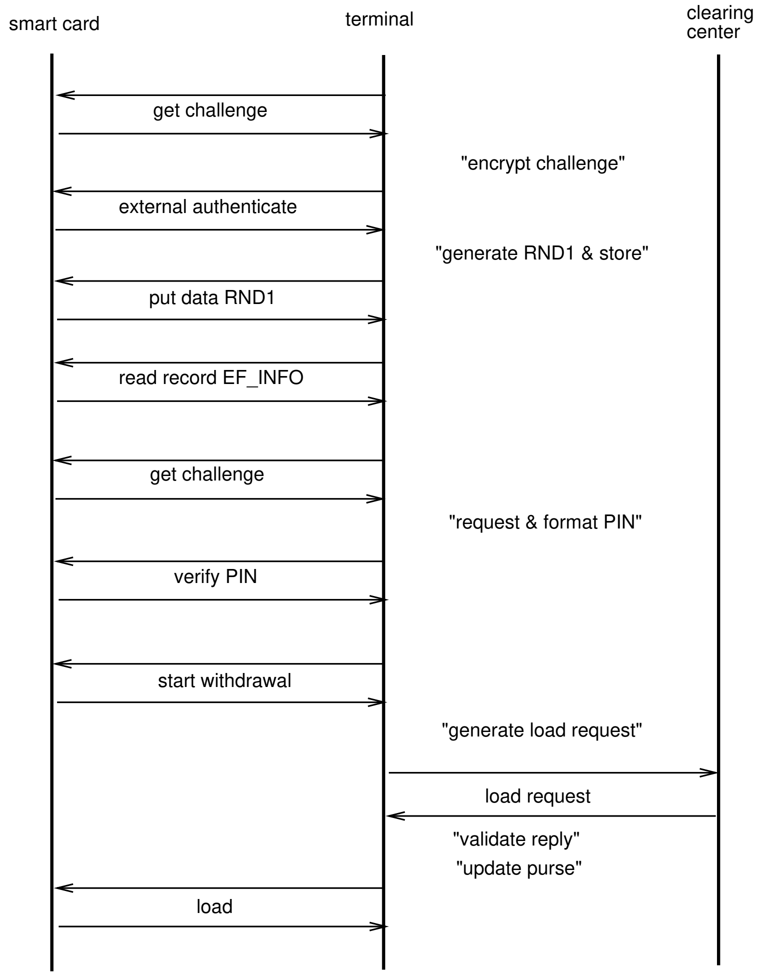

Figure 3: Withdrawal Protocol

1. The card is prompted for a random challenge $RND0$ used in step 2 below.
2. The *external authentication* command is issued to the card to authenticate the terminal to the card. $RND0$ is used to form a challenge as follows

$$ENCRND = E_{K_{INFO}}^{DES-ECB}(RND0, IV)$$

for some initial vector IV.

[6] Secure messaging denotes the keyed MACing of a message to be exchanged between terminal and card followed by the encryption of that message and its checksum. DES-CFB MAC and DES-CBC encryption is used here [4].

The key K_{INFO} used to generate the value *ENCRND* is a card-specific key derived from the master key KGK_{INFO} as described in (1). The master key is stored in the secure, tamper-proof terminal.

3. A random number *RND1* is generated by the terminal and transferred onto the card via the *put data* command.

4. The card's elementary data file *EF_INFO* is read by the terminal using secure messaging for which *RND1* is used as the initial control vector and K_{INFO} as key.

5. The card is prompted via *get challenge* for a random value *RND2*.

6. The user is requested to enter his personal identification number (PIN) which is transmitted onto the card using either secure messaging or PIN transmission in the clear.

   When secure messaging is used, *RND2* becomes the initial control vector and K_{PIN} the key to compute the DES-CFB MAC. K_{PIN} is derived from the master key KGK_{PIN} as in (1), which is stored in the terminal.

7. Depending on the card's state obtained in step 4 of the withdrawal preparation phase described above, the withdrawal process is started or resumed. (We shall assume here that a new withdrawal process is started.)

   The card replies to the *start loading* command with the relevant information for the clearing center to process the withdrawal. This includes the message ID, a transaction sequence number maintained by the card, a repetition counter, the transaction amount, the purse's value, the account number, logging information of the last transaction, the maximal transaction amount, the maximum purse value, and the loading key K_{LD}'s version.

   During this phase, the card-specific loading key K_{LD} is used to encrypt information which allows user authentication. The information that is encrypted here is the user's shadow account number (5 bytes) and the bank's identification number (4 bytes). Thus, encryption is used here to authenticate the information to the bank, not to provide confidentiality which is guaranteed through the secure banking network.

   K_{LD} is also used here to compute a DES-based MAC for the generated message, which is appended to the information listed above.

8. The terminal formats the data and transmits a message to the clearing center. The only (partially) encrypted field remains the end-user's account: all other data is passed in the clear at the application level. It is the banking network which provides transport layer data confidentiality.

   The message sent to the clearing center is protected with a MAC using a transaction key K_T which is in turn derived from K_{MAC} and a terminal maintained 8 byte long key index *KI* incremented at the beginning of each transaction.

9. The clearing center processes the request and reply. We here assume a successful reply, and failure at this step may trigger a transaction to undo prior, erroneous transactions.

   The reply message from the clearing center contains information which will be passed to the card in the next step. This card related information is protected via a MAC based on usage of K_{LD} known to the clearing center.

   An additional MAC is computed over the entire message sent to the terminal, based on K_T as in the above case of the request message.

10. The terminal issues the *load* command to the card. This request contains the amount to be credited to the purse and is secured via a DES-CBC message authentication code (MAC) using K_{LD}, known to the card and the clearing center (but not the terminal).

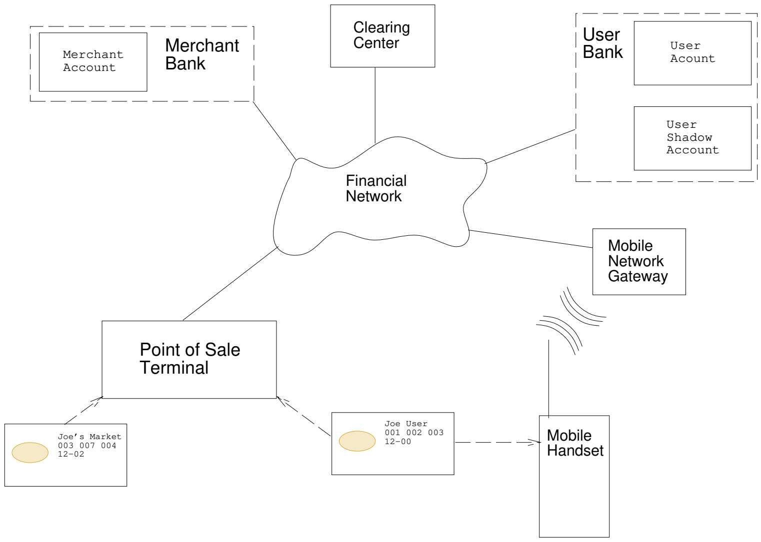

Figure 4: Mobile e-Cash Solution

11. If a failure occurs during the *load* an automatic undo-transaction is started to unroll the load transaction.

From the above description of the protocol, we can state the following findings:

1. The terminal is assumed to be tamper-proof: various master keys are stored by the terminal and used at various points to derive card specific keys (KGK_{INFO}, KGK_{PIN}).

2. The card does not store any master keys, but some card (and thus user) specific keys such as $K_{INFO}, K_{PIN}, K_{LD}$.

3. The terminal generates a transaction key used for data integrity. This key is based on a sequence number maintained by the bank terminal and a terminal specific key K_{MAC}.

4. Data transmitted between bank terminal and clearing center is not fully encrypted, only checksummed for message authenticity, at the application level. The banking network's lower layers are used to provide full data confidentiality.

3 Securely mobilizing Deutsche Geldkarte

This section describes solutions for migrating the Geldkarte bank terminal function to a mobile telephone (or ME for *M*obile *E*quipment). Figure 4 shows the envisioned system architecture. A mobile handset with two smart card slots acts as the e-cash loading terminal. One of these slots is permanently used by the ME's Subscriber Identification Module (SIM) card; the other slot can be used for application cards such as the e-purse scheme discussed here. The ME communicates over a mobile network with a gateway into the banking network for which we continue to assume a wired and secure network. All other components of the system remain unchanged.

The immediate concern is to determine what level of trust the card holder and bank are prepared to attribute to the ME. In the original Geldkarte architecture, the terminal is trusted to perform the following functions: card/terminal authentication, secure PIN delivery to the card, and terminal/bank authentication. In the first two cases, the function requires knowledge in the terminal of the appropriate master key, which are, respectively KGK_{INFO} and KGK_{PIN}. The terminal is also trusted with detailed information about the state of the Geldkarte application. Furthermore, the terminal's I/O system is trusted to securely handle the user's PIN and the user given transaction amount.

From the user's point of view the PIN is the most valuable quantity to be kept secret, while for the bank the most valuable quantities to be kept secret are the master keys. If the ME reveals a user's PIN then it is possible that unauthorized transactions could be made if the card is stolen or lost; however, knowledge of the master keys can be used to fake the terminal function, and potentially subvert the security of the Geldkarte system as a whole. Given this, our approach in migrating the terminal function to an ME is first to avoid the storage of master keys on the ME, and then to limit the risk of a user PIN being revealed by the ME.

Our solutions are distinguished by the level of trust that the user and bank ascribe to the ME. We shall not explicitly consider the case where the ME is trusted to provide an exact replica of the terminal functions. Indeed, we do not believe that this is a credible solution as master keys must be placed on the ME and, although technically feasible, the manufacturing costs of such a device appear prohibitive.

In the solution discussed in section 3.1, we however consider the ME to be tamper-proof with respect to storing a so-called *trust level token* and to provide secure I/O handling for PIN and transaction amount entry. The untrusted solution is discussed briefly in section 3.2.

Common to both solutions is the possible interruption of Geldkarte transactions due to unexpected user intervention. In section 3.3 we propose a method for adding state to Geldkarte transactions which allows for recovery from, for example, the user removing the card before the withdrawal protocol has completed.

3.1 Tamper-proof trusted mobile phone

We stress that the notion of trust and tamper resistance refer to the mobile phone keeping the user-supplied PIN secret, and not to trusting it with bank master keys or establishing a secure channel from the phone to the bank. The user may be prepared to make more valuable transactions on such mobile, trusted phones as compared to untrusted handsets. Similarly, the bank may also alter the characteristics of transactions it is prepared to accept given knowledge of the trust level it can assume about the mobile phone hosting the transaction.

Banks may restrict or strongly suggest users to make transactions from only such tamper-proof mobile phones. Thus, the first phase of a mobile enhanced Geldkarte transaction is for the card to determine the trust level of the hosting mobile phone. We propose to have the phone manufacturer embed a tamper resistant *trust level token* (TLT) into the handset. Well known TLTs can then be recognized by the cash card, using some cryptographic keying relationship between the phone, card and bank. We propose that public key cryptography (see [10] for definitions and examples) be used by the manufacturer to embed a signed TLT into the phone, which is to be verified by the card at the time of the transaction.

Let PM be the phone manufacturer, and let $Cert_{CA}(PM)$ be the public key certificate of PM. For each tamper-proof phone with identifier ME_{ID}, the PM creates and signs a certificate $Cert_{PM}(ME_{ID})$ for the phone where the certificate contains a field indicating that the phone is tamper-proof. The certificate $Cert_{PM}(ME_{ID})$ and the private key $K^{-1}_{ME_{ID}}$ corresponding to the certificate's public key $K_{ME_{ID}}$ are securely loaded into the tamper-proof phone and stored there.

When the Geldkarte is inserted into the mobile phone, the card first queries the phone for its trust level, TLT. The following protocol is run when the card is inserted:

1. The card generates a random number R and sends the pair (C_{ID}, R) to the handset, where C_{ID} is the identifier of the card.
2. The handset signs (ME_{ID}, C, R) using its private key $K^{-1}_{ME_{ID}}$, and sends to the card $Cert_{PM}(ME_{ID})$, $Sgn_{ME_{ID}}(ME_{ID}, C, R)$.
3. The card validates the certificate and checks the signature and extracts the TLT from the certificate. If the signature is not correct or the TLT indicates the phone is not tamper-proof then the card exits.

Here we have assumed that the card can verify the signature produced by the phone which implies that the card is able to verify $Cert_{PM}(ME_{ID})$ by having a copy of the public key of the certification authority (CA) that issued the manufacturer's certificate. This appears to be a reasonable assumption since first, we can assume a smart card to provide sufficient storage to hold such a root-CA key. Second, we can postulate that such a root-CA for all handsets exists; for instance this could be a national equipment certification agency.

If the card cannot perform the trust level verification of the phone then the bank can perform it, using the following protocol:

1. The card generates a random number R and encrypts R with K_{INFO} to obtain $E_{K_{INFO}}(R)$ and sends this along with card information C_{ID} to the phone.

2. The phone signs $E_{K_{INFO}}(R))$ with its private key $K^{-1}_{ME_{ID}}$ and forwards its certificate $Cert_{ME}$ (including TLT) and the signature $Sgn_{ME_{ID}}(E_{K_{INFO}}(R)))$ to the bank.

3. The bank retrieves the necessary certificates to verify the signature, and if it is correct, the bank sends to the phone $E_{K_{INFO}}(R+1\|TLT))$. The bank can determine K_{INFO} knowing C_{ID}.

4. The phone sends $E_{K_{INFO}}(R+1\|TLT))$ to the card.

5. The card verifies that this token is the encryption of $R+1$, and concludes that the phone is tamper-proof since only the bank could produce this encryption, and it trusts the bank verify the phone's signature and TLT.

Note however that this involves additional communication overhead with the bank. We thus strongly advocate our first approach with TLT verification by the card.

The usage of secret key technology can also be envisioned here: A secret key K_{TLT} can be introduced, known only to a card and a phone. Knowledge of this key or a key derived from it can be used to demonstrate the trust level of the phone. This however implies that the phone's secret key must be made known to the e-cash card; in practice an impossible task given the number of handsets and cards involved and the turnover rate in phones and cards.

The above protocol has replaced the *external authentication* phase of the protocol given in section 2. We do not use any system wide master keys, but also assert a somewhat weaker trust level of the "terminal". This trust level asserts the reliable handling of user input and output to the card, in particular PIN and transaction amount.

We now proceed to detail the mobile version of the withdrawal protocol, shown in Fig. 5. Since the handset no longer maintains any master keys and is not a trusted cryptography device, it can not perform the terminal's role in generating a transaction key K_T and encrypting messages sent to the bank over the banking network. Instead, we have to rely on the trusted cash card itself to perform these tasks. Our protocol achieves this by using the card's loading key K_{LD} to derive transaction keys. The following steps are executed:

1. The card verifies that the mobile phone is tamper-proof using the above scheme.

2. The user is requested to enter his personal identification number (PIN) which is transmitted onto the card in the clear.

3. The card and bank share information that varies from transaction to transaction, i.e. a transaction counter $LSEQ$. The card generates a transaction key K_{TRANS} from the card's loading key K_{LD} using $LSEQ$ as data that is shared with the bank.

4. This key, K_{TRANS}, is used to create a cyphertext which contains all the relevant information needed by the clearing center to process the withdrawal request. The cyphertext contains the information of steps 7 and 8 given in the protocol of section 2.3. Concatenated with this cyphertext is the (clear-text) card identity C_{ID}.

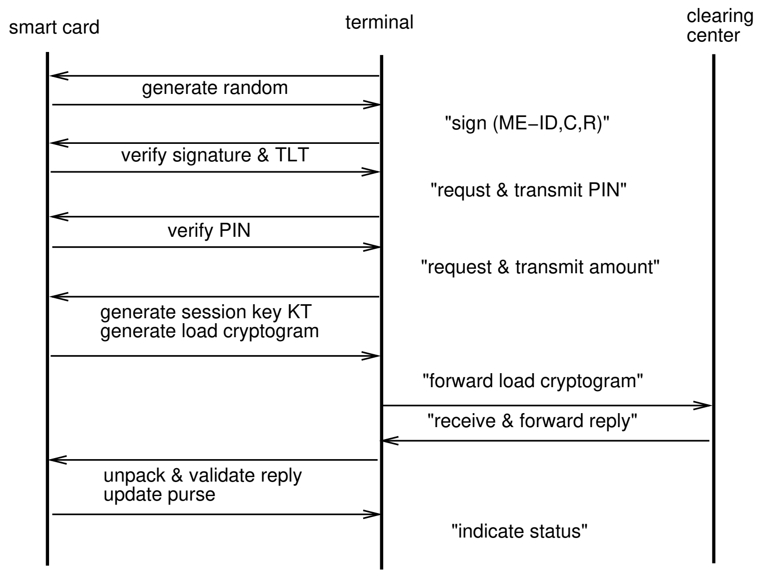

Figure 5: Mobile Withdrawal Protocol

5. The handset opaquely forwards this cyphertext to the clearing center.

6. The clearing center forms K_{TRANS} from its copy of K_{LD} and $LSEQ$, which it determines from C_{ID}. It then decrypts the cyphertext containing the load request. The contents of the transaction are examined and executed if there are no errors. The bank returns a reply encrypted in K_{TRANS}.

7. The phone forwards this reply opaquely to the card.

8. The card decrypts the reply and processes the bank instructions. If a failure occurs during the *load* an automatic undo-transaction is started to unroll the load transaction.

It is possible to use public key cryptography in the above protocol to establish a transaction key between the card and the bank. The card can carry a copy of the bank's certificate, and simply select a random transaction key K_{TRANS} which it can transmit to the bank encrypted under the bank's public key. The bank's reply can also be encrypted in K_{TRANS} and signed if further authentication is required.

3.2 Insecure mobile telephone

We consider an insecure handset a mobile phone which does not securely store a trust level token or has a corrupted TLT. In this case the extraction of TLT from the phone will either fail or produce an unverifiable value. At this point the card will need to decide if the transaction is to proceed, and if so then the protocol outlined from step 2 above is executed.

3.3 Interruption of transaction

Unlike the normal case when a bank terminal is used, the mobile user can abort a Geldkarte transaction at arbitrary times, by pulling out the e-purse, turning off the mobile handset, loosening the mobile connection, and so on. The solution here is already available in the current Geldkarte implementation and consists in maintaining state information for the ongoing transaction. We shall introduce following states on the purse:

1. S_n: the committed state reached after the last successfully completed transaction.

2. S'_{n+1}: the pre-committed state of the transaction while waiting for a reply from the clearing center.

3. S_{n+1}: the committed state of the purse after successful completion of transaction n.

Within the clearing center we maintain following state:

1. O_n: committed state after transaction $n-1$

2. O_{n+1}: committed state after transaction n

Figure 6 illustrates the state transitions of the e-purse as well as the clearing center's state transition and the message flow. The e-purse must be able to redo the last, not fully completed transaction. This is easily achieved by maintaining a transaction sequence number and a state variable.

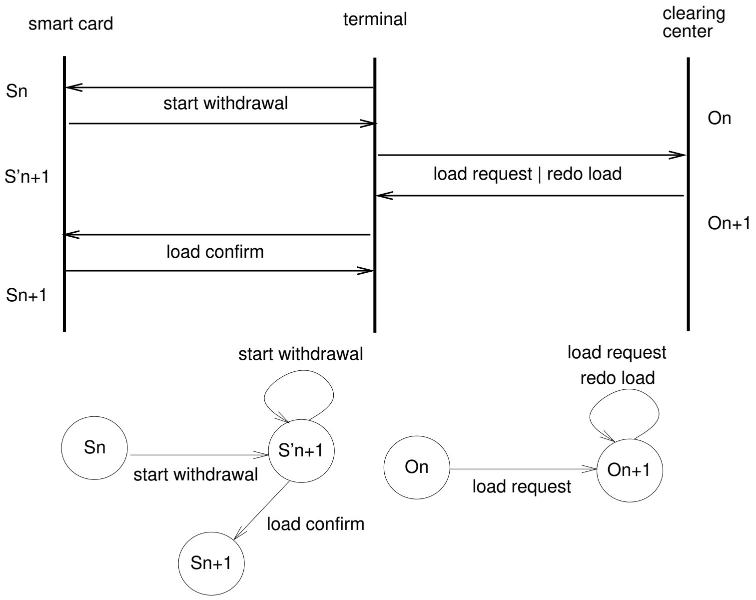

Figure 6: Transaction state diagram

While the purse's state is in transition (S'), it cannot be used for any other transaction, except to resume or restart a withdrawal transaction. Thus, it is not in the user's interest to interrupt voluntarily the withdrawal transaction. If however the transaction is interrupted, the clearing center is not affected, as the ongoing transaction is resumed at the next possible connection between card and clearing center.

4 Mobile Execution Platform

In the previous sections, we have described the security relevant aspects of an e-cash withdrawal protocol based on the Deutsche Geldkarte architecture. This section examines two industry standard mobile execution platforms which can be used to implement the above protocols.

Current mobile telephone technology, examplified here by the wide-spread GSM system [11], uses a end-user device which does not provide an extensible mobile execution environment. The only possibility of extending the built-in functionality of a handset, also referred to as Mobile Equipment (ME), is to rely on the ME's Subscriber Identifcation Module (SIM) which is a standardized smart card [7]. The SIM provides a platform to execute arbitrary computation and can interact with the ME through the SIM toolkit protocol [6] to access the ME's man-machine interface (MMI). In the latest release of the standards, support for a second smart card slot within the ME is provided. We expect that for user convenience as well as branding reasons, applications will involve a second smart card that can be inserted into the ME.

Although the SIM card evidently is a computing device, it has various shortcomings. First, smartcard technology remains limited in its performace and space characteristics. This restricts the complexity of the applications with respect to the size of their executables and data sets. Second, the SIM toolkit is limited in its functionality and awkward to use, with the SIM being a slave device of the ME.

An application protocol as described Section 3 requires authentication between terminal and smartcard. In our case, we require the handset to indicate some proof of its certified trust level to the smartcard[7]. This can, for obvious reasons, not be implemented with a smartcard alone and requires some functionality provided by the ME which goes beyond the GSM SIM toolkit standard. However, given such a modification to the standard GSM phone, our proposed solution can be implemented with an extended SIM toolkit functionality; awkward as it might be.

One further, open issue of using the SIM toolkit is the handling of the two smart cards implied in our solution: the traditional SIM card as well as the e-cash application card. How the handset's SIM card can interact with the application card is currently still unclear. The only possible solution we see is the merging of SIM and application functionality on a single card which for various reasons such as marketing and user convenience is not desireable.

Another, potentially more attractive and more powerful mobile execution platform to implement our proposal is the Wireless Application Protocol (WAP) industry standard [14, 15]. It provides a WWW-like application paradigm, consisting of a browser for hyperlinked documents, formatted in the Wireless Markup Language (WML) [16], augmented with a light-weight scripting environment [17]. Within WAP, communication is provided through the WAP protocol stack which can be hosted on various wireless bearers, such as GSM's SMS, USSD, the emerging GPRS, or the future UMTS.

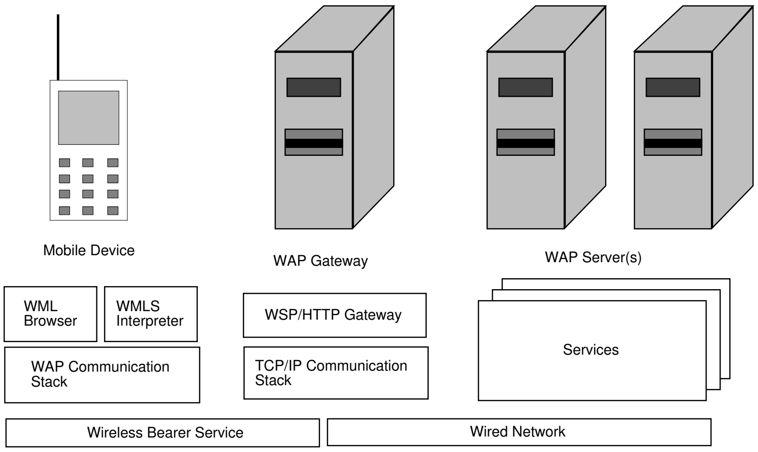

Figure 7: WAP Architecture

The WAP client execution platform consists of the (micro) browser, the WMLS interpreter, and a set of standardized WMLS libraries [18]. The libraries allow access to the ME hosted smartcards, be it the SIM or any additional smartcard. WMLS scripts are referenced either by WML document *task* entries or from some WMLS script.

Using WAP, the handset executes the transaction, acting as master to the smart card and as the communicating entity of the clearing center. The proposed trust level assertion can be expressed using WAP's scripting mechanisms and proposed APIs to smart card functionality and persistent, secure storage. The proposed WAP APIs to access smart cards allow for access to any number of cards [13] from a WAP/WMLS application.

[7]Thereby providing some limited external authentication.

Unlike the SIM toolkit based approach, it appears that a WAP based implemetnation of our proposal does not require modifications to the WAP environment. For this reason and due to our belief that ultimately WAP will represent a more flexible and powerful application environment for mobile, personal, and pervasive devices we favor the WAP based approach.

5 Conclusion

This report has provided an in-depth analysis of the current Deutsche Geldkarte system, which we consider as a generic embodiement of similar e-cash schemes. Based on the Deutsche Geldkarte system architecture we have presented protocols which would enable the secure e-cash withdrawal in a mobile environment on a device which is not fully tamper-proof and communicates over non-secured mobile networks.

Our protocols rely on a fully trusted, user-owned secure token, based on well-known smart card technology. Our protocols support the assertion of a trust-level for the ME as well as secure communication with a clearing center without generating additional traffic between the mobile user and the centralized clearing center. Interruptions of transactions are handled by preserving sufficient state information on the smart card for protocol resumption if necessary.

We believe our solutions to be implementable using either a modified SIM application toolkit or the application platform of the Wireless Application Protocol which we consider a more powerful and flexible environment for the e-cash withdrawal application.

The popularity of an e-cash solution could be vastly enhanced if end-users were enabled to have access to the e-cash download function from their mobile telephone handsets. We expect such systems to be implemented shortly and believe that they will be readily accepted by the end-user population.

References

[1] Common Smart Card and Electronic Purse Standard (ceps). World-Wide Web. http://www.protonworld.com.

[2] Deutscher Sparkassenverlag GmbH. *Schnittstellenspezifikation für die ec-Karte mit Chip: Die elektronische Geldbörse: Börsenkarte*, August 1995.

[3] Deutscher Sparkassenverlag GmbH. *Schnittstellenspezifikation für die ec-Karte mit Chip: Key-Management*, August 1995.

[4] Deutscher Sparkassenverlag GmbH. *Schnittstellenspezifikation für die ec-Karte mit Chip: Datenstrukturen und Kommandos*, Januar 1997.

[5] Deutscher Sparkassenverlag GmbH. *Schnittstellenspezifikation für die ec-Karte mit Chip: Geld-Karte Ladeterminals*, Januar 1997.

[6] ETSI, Sophia-Antipolis, France. *Digital cellular telecommunications system (Phase 2+): Specification of the SIM Application Toolkit for the Subscriber Identity Module - Mobile Equipment (SIM-ME) interface (GSM 11.14)*, November 1998. Version 7.1.0.

[7] ETSI, Sophia-Antipolis, France. *Digital cellular telecommunications system (Phase 2+): Specification of the Subscriber Identity Module - Mobile Equipment (SIM-ME) interface (GSM 11.11)*, November 1998. Version 7.1.0.

[8] Walter E. Greenberg. Mastercard cash: Helping shape the future of money with a viable alternative to cash. In *Proceedings of CardTech/SecurTech*, pages 39–46, May 1996.

[9] Armand Linkens. The electronic purse is live in Belgium. In *Proceedings of CardTech/SecurTech*, pages 59–70, May 1996.

[10] Alfred J. Menezes, Paul C. van Oorschot, and Scott Vanstone. *Handbook of Applied Cryptograhpy*. CRC Press, 1997.

[11] Michel Mouly and Marie-Bernadette Pautet. *The GSM System for Mobile Communications*. Mouly & Pautet, 1992. ISBN 2-9507190-0-7.

[12] Gerhard Neuhierl. The german banking chip card project. In *Proceedings of CardTech/SecurTech*, pages 47–57, May 1996.

[13] Nokia Mobile Phones. *WAP Smart Card WMLScript extension library*. WAP Forum, September 1998. Proposed Extension Library.

[14] WAP Forum. *Wireless Application Protocol: Architecture Specification*, April 1998. version 1.0.

[15] WAP Forum. *Wireless Application Protocol: Wireless Application Environment Overview*, April 1998. version 1.0.

[16] WAP Forum. *Wireless Application Protocol: Wireless Markup Language Specification*, April 1998. version 1.0.

[17] WAP Forum. *Wireless Application Protocol: WMLScript Language Specification*, April 1998. version 1.0.

[18] WAP Forum. *Wireless Application Protocol: WMLScript Standard Libraries Specification*, April 1998. version 1.0.

[19] Konrad Wrona, Ralf Keller, Ian Herwono, and Fiona Williams. Electronic Commerce in Deutschland und weltweit: Neue Zahlungssystem und ihre Anwendungsbereichte. *PIK*, 21(1):3–10, 1997.