A Payment Scheme for Mixes Providing Anonymity[1]

Elke Franz, Anja Jerichow, Guntram Wicke

Dresden University of Technology, Department of Computer Science, D-01062 Dresden
{efl, jerichow, wicke}@inf.tu-dresden.de

Abstract. Mixes allow the users of electronic data networks to communicate with each other without identifying themselves or uncovering their relationship. Several security applications suggest the usage of mixes. However, such concepts are often not commercially used in practice. One reason may be that providers are only willing to offer anonymous communication if their payment is arranged.

We present an anonymity service provided by mixes and describe a payment scheme which allows its anonymous, secure payment. The scheme aims to achieve security for all: the user, the provider and the bank. Detailed protocols for the payment of mixes are described.

Keywords: anonymous communication, mix service, tick payment, payment protocols, charging

1 Introduction

In the last decade the discussion about anonymity and privacy has become more and more important. This requires anonymous communication, i.e. it is to keep secret who is communicating with whom at which time and maybe from which location. The usage of mixes [Chau_81] is often suggested for providing anonymity in telecommunication networks, mobile communication systems and the internet [PfPW_91, FJKP_96, GüTs_96, SyRS_96, LoEB_97].

We decide between two networks providing the communication service and the anonymity service. For payment of these services there must be two different schemes: one for paying the provider of the communication service and one for paying the provider of the anonymity service.

This paper presents an anonymity service provided by mixes and describes different approaches of payment schemes for the mixes. There are some anonymous, secure payment schemes on the market like ecash. The special about our approach is that we use the real messages sent to the mixes to transmit the payment data in a very efficient

[1] Parts of this work were supported by the German Science Foundation (DFG), the Gottlieb Daimler- and Karl Benz-Foundation and the German Ministry of Education, Science, Research and Technology (BMBF).

way. Section 2 gives an overview about mixes. Suitable payment schemes for the anonymity service are discussed in Section 3. We suggest the usage of an interactive micropayment scheme. To transfer small amounts, tick payment is used. Section 4 discusses several payment protocols, from which we describe one protocol in detail. Necessary protocols between users, network and bank are explained in Section 5. Section 6 concludes and gives an overview about further work.

2 The Usage of Mixes for Anonymity

2.1 Requirements and Definition of the Mix Service

There are several ways to provide anonymous communication in a network, e.g. broadcast for recipient anonymity, DC network for sender anonymity [Chau_88] and mixes for protection of the communication relation [Chau_81, PfWa_86]. In this paper we focus on mixes. A mix is a node in a communication network. Mixes collect and store incoming messages and change their order and appearance by encryption before forwarding them. Due to the functionality of a mix it therefore ensures unlinkability of incoming and outgoing messages. Thus the users stay anonymous while communicating.

The unlinkability of senders and recipients of messages is already achieved by a single mix. However, this mix knows the sender and the recipient of every message. If it is not trustworthy, the anonymity of the users is not guaranteed. Unfortunately, the user cannot be sure whether a mix is trustworthy. A general rule to improve this situation is the following: The more mixes are used the more likely is the usage of at least one trustworthy mix whereas independent providers should operate them.

The attacking model defines the strength of the system against possible attacks. The *attacking model for mixes* says: An attacker is allowed to observe all data traffic in the entire network and to control all mix nodes but one. By this model an eavesdropper cannot trace a message through the mix network.

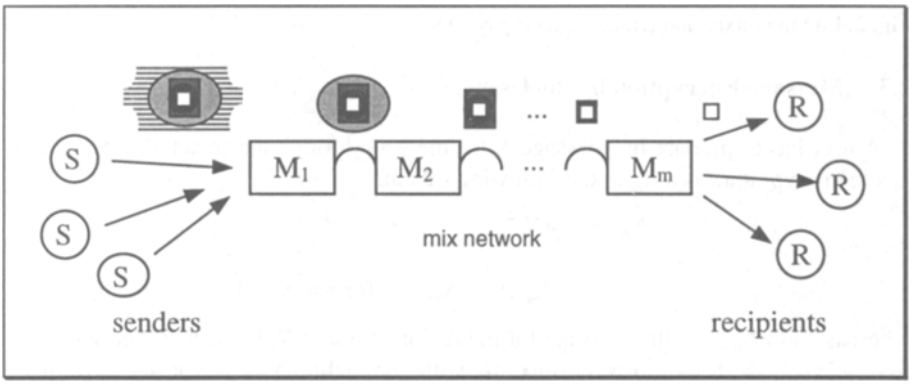

Fig. 1. Picturing the different appearances of one message at different points of the mix network

As mentioned above, payment must be realized for both the communication and the anonymity service. Payment of the communication service provider is already solved in current networks. In this paper we will answer the question of how to pay the provider of the anonymity service. We define the mix service to be the anonymity service whereas each mix is an independent economic unit.

2.2 Payment: Provider of a Mix Chain vs. Provider of a Mix

Mixes can be linked together in any possible way according to the requirements. As a result of the different feasible offers of the mix service to the user we differ between the two following ways for payment (Fig. 2):

First, one provider offers a fixed mix chain (case a): The different owners of the mixes in his chain are under his contract. He accepts the payment from the users via one mix selected by him. This is sufficient due to the fixed number of mixes in such a mix chain. Therefore, the service price is predetermined. The mix owners receive their money by this provider as stated in the contract.

The second possibility is that each mix owner acts as its own provider (case b): In that case, the mixes build a sequence but are not in a fixed chain. The users have to pay each mix separately for its service.

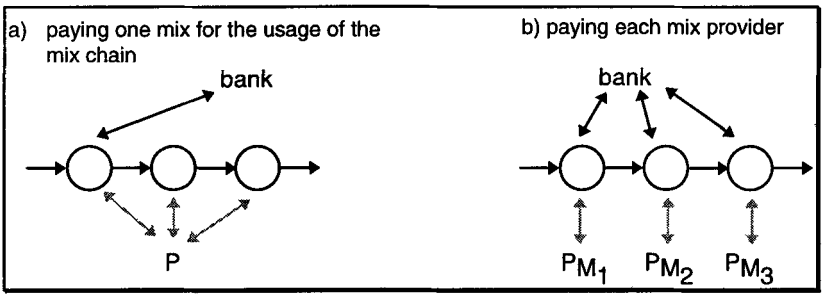

Fig. 2. Linking mixes and arranging their payment

2.3 Message Encryption by the Users

A user has to prepare his message N for the way through the selected mixes, i.e. he has to encrypt it according to the following scheme:

$$N_R := c_R(N) \tag{1}$$

$$N_i := c_i(z_i, A_{i+1}, N_{i+1}) \quad \text{for } i = m,\ldots,1 \tag{2}$$

whereas $N_R (=N_{m+1})$ is the message for the recipient R and N_1 is the message sent to the first mix. A_i is the address of mix M_i with the public key c_i for its asymmetric cryptosystem.

For every mix M_i a random string z_i must be included. Otherwise, an attacker could encrypt a message again with the public key of a mix and would be able to compare

the result with the former input. A mix must also test for replays to avoid the bridging of itself by re-sending a message.

3 Suitable Payment Schemes

3.1 An Interactive Micropayment Scheme

There are several approaches for payment. A delivered service can be pre-paid, pay now or post-paid: *Pre-payment* means that a user has paid in advance before using the service. If he can use the service without paying immediately, we call it *post-payment. Pay now* means paying immediately.

If post-payment is used, the mix provider has to trust that he receives his money afterwards. On the other hand, the user has no guarantee of fulfillment of the service if pre-payment is used. Thus, after only one protocol step the advantage of the provider would be considerable large.

Instead of paying the full service in one step, we therefore suggest:

- The message is split into small blocks.
- The mix service receives payment for each delivered message block.

Nevertheless, there is an advantage of one party over the other but we keep it as small as possible by this approach.

For this protocol, it is necessary that mixes and users interact during the message transfer. A user sends content and payment data together in one message block. The mix separates this data again in order to get its payment. Due to the splitting in message blocks and the necessary interaction we call this scheme *interactive micropayment scheme.*

Note, we do not distinguish between charging and paying (for this discussion see Sect. 5).

3.2 Using Tick Payment for the Mix Service

We suggest the usage of tick payments [Pede_96] as a mechanism for the interactive micropayment scheme. Originally, the idea of ticks was used for authentication [Schn_96]. Tick payment is a very efficient way for transferring small amounts since a tick means that only some additional bits are added to each message.

The principle of ticks is based on a one-way function. This one-way function is a length preserving permutation f but nevertheless called hash function in the following. To start the calculation, a random number a is chosen. The hash function is applied n times to this random value and thus a chain of n hash values is computed:

$$f^1 = f(a); \quad f^2 = f(f^1(a)) = f(f(a)); \ldots ; \quad f^n = f(f^{n-1}(a)) = f(f(f(\ldots f(a)))) \qquad (3)$$

Each hash value f^i is named a tick. For simplification, we will call f^{n-i} as tick t_i.

This principle can easily be used to pay in small amounts, e.g. for the small information blocks transmitted between a customer and a merchant (e.g. the provider).

Thus neither the merchant nor the customer will gather an important advantage over the other party.

At the beginning the customer creates a chain of hash values. After that he makes out a check and sends it to the merchant. Fig. 3 shows the necessary data that a check must include.

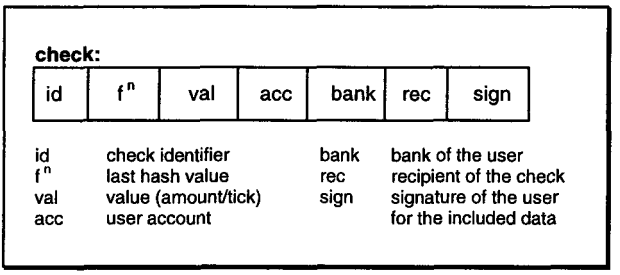

Fig. 3. Necessary data for the check

It must be prearranged which amount of money is represented by every tick. The maximum number of usable ticks n determines the maximum value of the check which is equal to n times the amount per tick.[2] However, it is not necessary to use the whole possible amount of money.

The message is transmitted in small steps. For each data block that the customer receives from the merchant he sends a tick. The merchant must test the correctness of the tick by applying the hash function f to a received value t_i and comparing the result with the last valid tick t_{i-1}.

If the transmission is complete, the merchant can cash the check at the bank whereas the received ticks must be submitted together with the check. He gets the actual value of the check by multiplying the amount per tick and the number of received ticks. The bank can also verify the ticks by applying the hash function. Based on the prearranged amount per tick and the quantity of received ticks the merchant gets his money.

In the following section we will apply the principle of ticks and describe different micropayment schemes for the payment of the anonymity service offered by mixes. We change our macroscopic view that one message is paid with one check to a view in detail: One message block is paid with one tick.

4 Basic Protocols

4.1 Assumptions

To discuss the payment protocols the following assumptions are made:

[2] There is also another possibility as suggested in [Pede_96]: If a trapdoor hash function is used, any number of ticks for a chosen value can be generated by the sender. In the following, hash functions without trapdoor are used.

- One check is used for payment of many message blocks.
- Each message block requires separate payment.
- Each message block includes payment data.
- The encryption scheme for sender anonymity is used: The sender prepares the message for the mix network and, therefore, he also pays for the anonymity service.
- The payment is either made to one distinguished mix of the mix chain whereas any mix can be used for this task, or every mix is paid when using its service. We assume for the former case that the mix chain provider defines the mix that is responsible for payment and informs the users about this decision.

4.2 Discussion of Different Approaches

There are different approaches for implementing the suggested interactive micropayment protocols resulting from the following aspects:

- Delivery of the message can take place before or after payment.
- Payment can be triggered by the sender or the recipient of the message.

If payment before delivery of the message is used, a message block is first paid and then proceeded. If payment after delivery is used, the mix will not get the payment for its service before the next node has received the data, i.e. first the message block is proceeded and afterwards it is paid. Note, if the recipient triggers payment, the used tick must be redirected via the recipient. The following protocols describe the steps between the user and the mix network for all combinations.

Payment before Delivery of the Message Block (Table 1).
If the *sender triggers the payment*, he includes the check and the first tick in the first block of the message. The mix network stores this check. Each following message block only includes the appropriate tick. Thus the anonymity service is immediately paid for and further steps are not necessary.

If the *recipient triggers the payment*, the first block of the message includes the check and the first tick. To allow only the recipient to trigger the payment, all ticks must be encrypted with his public key. The mix stores the check and sends the encrypted tick to the recipient of the message. The recipient decrypts the tick and sends it back to the mix network. If the mix network has received the decrypted tick, i.e. its payment, it sends the message block. The following blocks are handled in exactly the same way. Therefore, a mix is not able to get its payment before the recipient has sent back the decrypted tick.

In comparison to the protocol described above, two additional steps are necessary for the transfer of each message block.

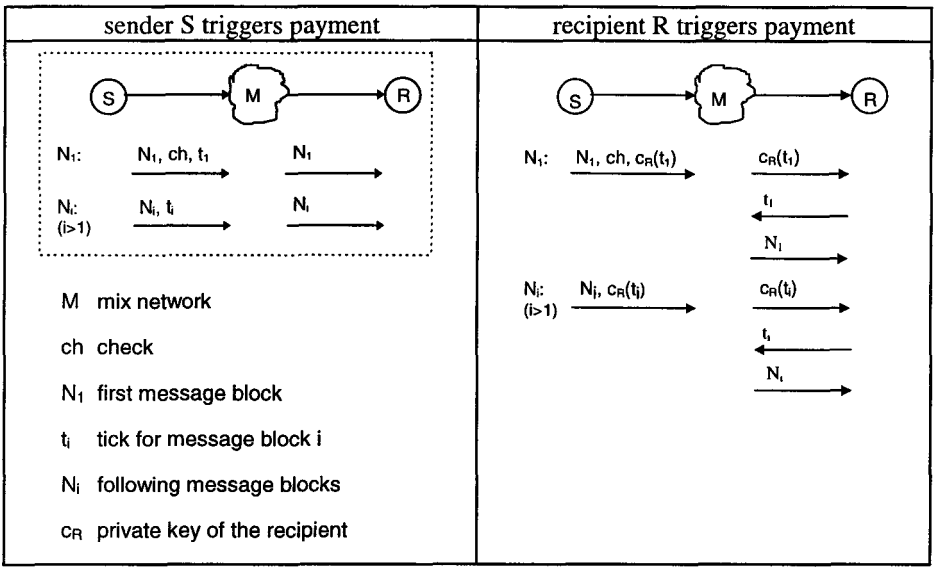

Table 1. Basic protocols for payment before delivery

Payment after Delivery of the Message Block (Table 2).
If the *sender triggers the payment* he must send the first message block together with the check. The mix network stores the check and sends the message block to the recipient who then sends an 'ok' to the sender of the message. Hence, the sender can be sure that his message block is arrived, i.e. he has guarantee of message receipt. He then sends the mix network the tick. The same applies for the following blocks, payment is triggered by the sender after he has received the 'ok'.

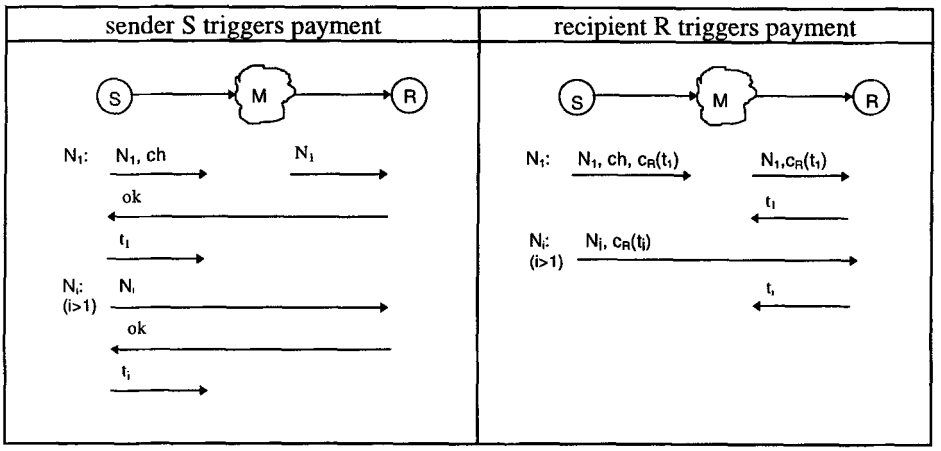

Table 2. Basic protocols for payment after delivery

To improve efficiency, the tick could always be sent together with the following message block. However, the delay by the mix network slows down the transfer rate.

If the *recipient triggers the payment*, each message block includes an encrypted tick again. The first message block also includes the check. After receiving a block, the recipient sends back the decrypted tick to the mix network. By this, the sender cannot be sure that the recipient has got the message blocks. However, the mixes will be interested in sending the messages to the recipient to get their payment. We, therefore, can assume that the mix will correctly process each message block.

The scheme for payment before delivery where the sender triggers the payment is the most efficient of all discussed approaches. That is why we will describe this scheme very detailed now. However, if the guarantee of message receipt is considered as more important, than the second protocol group should be applied.

4.3 Detailed Protocols for Payment before Delivery

Payment for the Provider of a Mix Chain (Fig. 4).

Here we have a distinguished mix of the mix chain for accepting the payment. For this protocol, it is suitable to use the first mix since there is no need to pass the check and the ticks through the whole mix network. Moreover, there is only a minimum extension of the message. Only the message prepared for the first mix includes payment data that are removed after passing this mix.

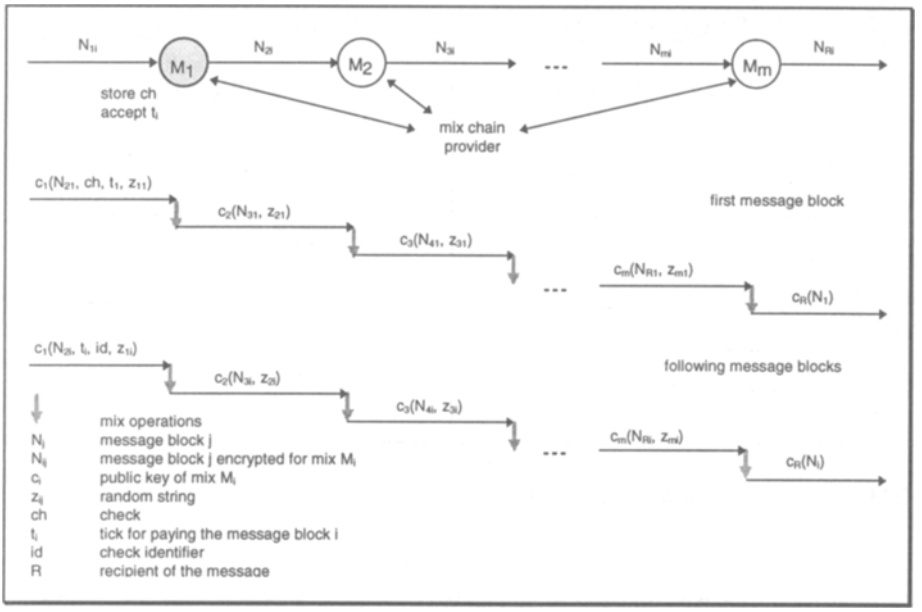

Fig. 4. Basic protocol for payment of a mix chain provider

The payment runs as follows: The sender includes a check *ch* and the first tick t_1 in the first message block N_{11}. The check, which contains an identifier, is stored by the mix. Each following message block also includes a tick. Moreover, it includes the identifier *id* to assign the tick to the appropriate check. Before passing a message block to the next node, the mix can verify the tick as described in Sect. 3.2. A tick t_i represents the payment for the message block N_{jj}.

Payment Scheme for each Mix Provider within a Chain (Fig. 5).
If the user wants to select the mixes himself he has to pay *each* mix provider of this chain separately. The principle is the same as described in the previous section.

The only difference, the first message block now contains a check and the first tick for each provider. Each mix must store its check. The following message blocks include only the appropriate ticks and the check identifier.

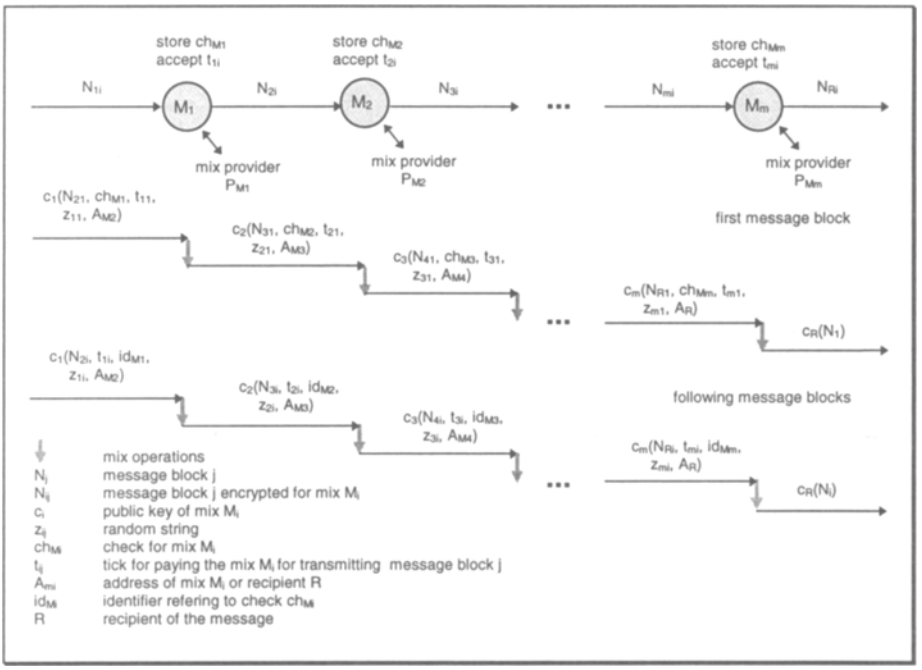

Fig. 5. Basic protocol for payment of each mix provider of a chain

Both protocols are very efficient because no further processing steps are needed. The sent ticks can immediately be verified. One disadvantage to point out is that the sender cannot be sure whether the mix really transmitted his data to the recipient.

5 Charging: Protocols between User, Network and Bank

5.1 General

Former sections discuss the protocols between user and network, i.e. the service providers of the mixes. Such protocols are necessary to transmit the payment data to the providers. In this section we consider the protocols which are necessary to cash a check at the bank (see [BCMM_95] for details).

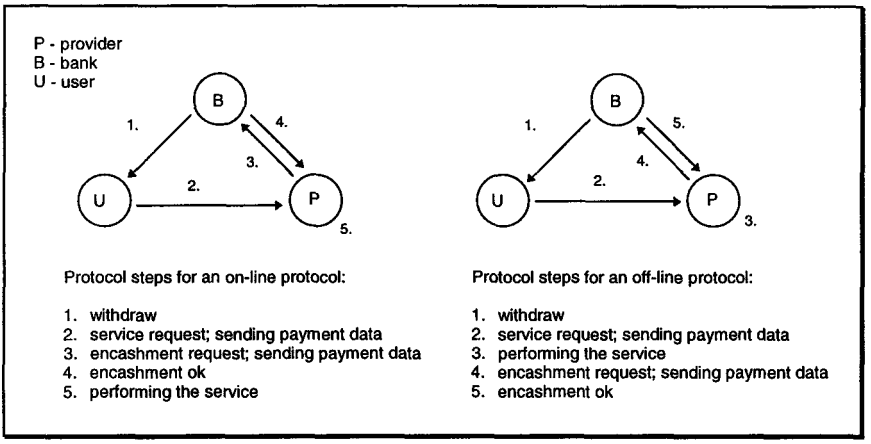

Fig. 6. Protocols between users, providers and bank

In general, there are two possible protocols to perform the payment: on-line and off-line protocols (Fig. 6). If an on-line protocol is used, the mix does not perform the requested service, i.e. the transmission of the appropriate message, before it has received the encashment ok from the bank. Thus it can be sure that it really will get the payment from the users. But there is an important disadvantage: The request of the mix for all messages slows down the transmission of messages.

The other possibility is to use an off-line protocol. The mix immediately processes the received messages, which includes the payment data, i.e. it sends the message to the next node. Afterwards it starts an encashment request to the bank. Thus the transmission of messages will not be slowed down. Therefore we choose this variant for paying the anonymity service.

One problem we have not considered yet is that the mix provider cannot be sure whether it really will get valid payment for his service. One possibility to avoid this problem in off-line protocols is the usage of tamper proof hardware for the generation of checks and ticks. This device prevents double spending. Therefore, no on-line protocol is needed for double spending detection.

However, this means that every user must own a tamper proof hardware device.

5.2 Using a Tamper Proof Hardware Device (TPH)

The task of the TPH is to achieve trustworthiness in the payment process for all, i.e. the user, the bank and the provider. Trustworthiness in the TPH means

- *for the bank*: It is not possible that a user overdraws his accounts while creating checks. The bank is able to recognize if one of its customers has been creating such a check with his TPH, and will only accept those checks.
- *for the user*: A check once made out with the TPH is valid and can be therefore correctly billed on his account. The user anonymity is kept against both the provider and the bank.

- *for the provider*: He can recognize whether a check is valid. He knows that checks generated with a TPH will be accepted by the bank and that he will get the correct amount for them. The checks are not usable by others.

From these points we derive the following requirements on a TPH:

- The tamper proof and the correct functionality of the TPH must be guaranteed.
- The identity of a user cannot be recognized from a check (untraceability), i.e the user is kept anonymous. Moreover, unlinkability of different checks of one user must be guaranteed.
- It must be possible to decide whether a check was created with a TPH in order to avoid the usage of a forged check. This must be possible for the bank as well as for the provider.
- The bank must be able to test the correct identity of the provider who wants to cash the user's check.

More details about building and using a TPH can be found in [PPSW_96].

5.3 Sketching the Charging Process

For the charging process, the user's TPH must communicate with both the bank and the provider. These parties can take for granted that the TPH correctly executes all necessary functions due to its functionality.

Every user has a 'normal' bank account and an electronic purse, i.e. an account managed by the TPH. The TPH consists of two parts: one for performing user actions, e.g. making out checks, and one for performing bank actions, e.g. signing.

Before the user can send messages through the network he must prepare his messages, i.e. generating and attaching checks and ticks. Of course, this is only possible if the device is loaded.

Loading money on the TPH. Loading means to withdraw money from the bank account. The user selects the amount which is to transfer and sends his inquiry to the bank. Provided that the user has enough money on his bank account, the bank transfers the required amount. The TPH stores it at the user's TPH account. If the account is empty, the TPH must be loaded again.

Only the withdraw process between bank and TPH must be performed on-line. The check itself can be created off-line.

Creating the check. Hereby, the TPH makes out the check according to Fig. 7 and generates the ticks. As mentioned in Sect. 5.1, the maximum amount of a check is limited. It can be calculated from the possible number of ticks and the amount per tick.

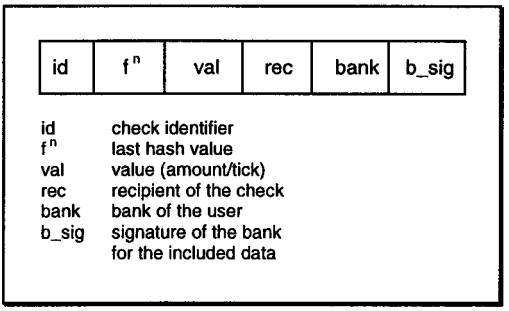

Fig. 7. Necessary data for a check created by the TPH

The user is now able to send his message together with the necessary payment data. The provider of the mix or the mix chain who receives it can cash in the check.

Signing the check by the bank. The check included in the message must be signed by the bank. This warrants that the provider and the bank are able to test the correctness of a check. Thereby, blind signatures [Chau_85] must be used since the user is to be kept anonymous against the bank (as mentioned in the requirements of Sect. 5.2). I.e. the checks will be blinded before they are signed by the bank transaction part of the TPH. The signing key of the bank is stored in the bank and in the user part of the TPH.

This procedure can be performed off-line in the TPH because the correct withdraw from the bank account is already done. Moreover, due to the functionality of the TPH it is guaranteed that a check can only be in the range of the balance of the TPH account.

Testing the correct identities of bank and provider. The provider can verify the correctness of the check by testing the bank signature. If he has also received the ticks for the check, he can cash it at the bank. The bank must make sure that the check really belongs to the recipient, i.e. to the one provider who claims to have received it. This can be easily performed:

If the check includes the public key of the provider as a sign for the recipient, the bank could use a challenge-response protocol to check the identity of the provider. To realize this protocol, the bank sends a random number which the provider must encrypt with his private key. After receiving the encrypted number the bank can verify the identity of the provider. For that the bank decrypts this value with the public key of the provider and compares it with the random number.

If the bank has also successfully tested its own signature included in the check, it can be sure that this check is correct and can transfer the money to the provider's account. Thus, both the bank and the provider can trust the anonymous payment of the user.

6 Summary and Outlook

In this paper we introduced several protocols for payment of an anonymity service provided by mixes. This can be done very efficiently if the transfer of a message is related to the transfer of the payment data. I.e. the message can drop the 'money' required by the mix while passing it. By using an interactive micropayment scheme, i.e. applying the idea of tick payment to mixes, we keep the advance of one party over the other as small as possible. We described in detail the protocols for payment before delivery of the message block whereas payment is triggered by the sender.

Other protocols which have the property of payment after delivery must be researched as well. The reader may ask, why does the mix not send a request for payment before it transfers a message block to the next network node. We considered this approach to be too time inefficient and, therefore, did not follow up it.

For all protocols described here we applied asymmetric encryption schemes. In further investigations, the application of hybrid encryption schemes has to be examined as well.

The discussion of boundary conditions, especially the relations between user, network and provider, must be considered to allow secure anonymous payment.

The described protocols are based on hash functions without trapdoor. It may be that trapdoor hash functions are also suitable. Furthermore, perhaps they are more efficient for paying a mix provider. In future work such functions must therefore be investigated.

We did not consider fall back security, i.e. protocols for identifying a double spender in case of a corrupted tamper proof hardware device.

In this paper we assumed that the sender pays for the anonymity service and, thus, keeps his anonymity due to the used encryption scheme. If the recipient wants to maintain anonymity, the protocols will differ from the here suggested. Thus, this is another topic of further investigation. Sharing the payment between the sender and the recipient may rise other problems. Especially attacks resulting from the sharing are of special interest. On one hand, the users must not pay less than the agreed amount of money. On the other hand, it must be avoided that the provider can get more money than negotiated before.

The suggested protocols can be applied to every application that uses mixes. We explained the general approach of anonymous communication by mixes and how the payment of such a service may be performed. Depending on the application and the used communication network, e.g. mixes in the internet for anonymous web access and mixes in ISDN for anonymous telephony, the protocols must be adopted to the context in which they are used. However, we made clear that payment can be organized anonymously for the usage of the anonymity service, i.e. the mix network.

We say 'thank you' to Andreas Graubner and Andreas Pfitzmann for all the helpful comments and discussions.

7 Literature

BCMM_95 Antoon Bosselaers, Ronald Cramer, Rolf Michelsen, Stig Mjølsnes, Frank Muller, Torben Pedersen, Birgit Pfitzmann, Cristian Radu, Peter de Rooij, Berry Schoenmakers, Matthias Schunter: Functionality of the Basic Protocols; CAFE Public Report IHS8341, CWI Amsterdam, October 7, 1995; CAFE (Esprit 7023) Deliverable IHS 8341 (confidential), September1,1995.

Chau_81 D. Chaum: Untraceable Electronic Mail, Return Addresses, and Digital Pseudonyms; Communications of the ACM 24/2 (1981) 84-88.

Chau_85 D. Chaum: Security without Identification: Transaction Systems to make Big Brother Obsolete; Communications of the ACM 28/10 (1985) 1030-1044.

Chau_88 D. Chaum: The Dining Cryptographers Problem: Unconditional Sender and Recipient Untracebility. Journal of Cryptology 1/1 (1988) 65-75.

GüTs_96 C. Gülcü, G. Tsudik: Mixing Email with BABEL; Proc. Symposium on Networking and Distributed System Security, San Diego, IEEE Comput. Soc. Press, 1996, pp 2-16.

FJKP_96 Dogan Kesdogan, Hannes Federrath, Anja Jerichow, Andreas Pfitzmann: Location management strategies increasing privacy in mobile communication; 12th IFIP International Conference on Information Security (IFIP/Sec '96), Chapman & Hall, London 1996, 39-48.

LoEB_97 T. Lopatic, C. Eckert, U. Baumgarten: MMIP - Mixed Mobile Internet Protocol; CMS'97 - Communications and Multimedia Security, IFIP TC-6 and TC-11, 22-23 Sept. 1997 in Athens (Greece).

Pede_96 Torben P. Pedersen: Electronic Payments of Small Amounts. 1996 Security Protocols Workshop, 10.-12.4.1996, Cambridge, UK, proceedings to appear in LNCS, Springer-Verlag.

PfPW91 A. Pfitzmann, B. Pfitzmann, M. Waidner: ISDN-MIXes - Untraceable Communication with Very Small Bandwidth Overhead. 7th IFIP International Conference on Information Security (IFIP/Sec '91), Elsevier, Amsterdam 1991, 245-258.

PfWa86 A. Pfitzmann, M. Waidner: Networks without user observability – design options; Eurocrypt '85, LNCS 219, Springer-Verlag, Berlin 1986, 245-253; Extended version in: Computers & Security 6/2 (1987) 158-166.

PPSW_96 Andreas Pfitzmann, Birgit Pfitzmann, Matthias Schunter, Michael Waidner: Mobile User Devices and Security Modules: Design for Trustworthiness; IBM Research Report RZ 2784 (##89262) 02/05/96, IBM Research Division, Zürich, Feb. 1996.

Schn_96 Bruce Schneier: Applied Cryptography Second Edition: protocols, algorithms, and source code in C. John Wiley & Sons, Inc., p. 53, 1996.

SyGR_97 Paul F. Syverson, David M. Goldschlag, Michael G. Reed: Anonymous Connections and Onion Routing; 1997 IEEE Symposium on Security and Privacy.