Publication date: October 16, 1994. This paper will appear in: Proceedings of the Internet Society 1995 Symposium on Network and Distributed System Security, San Diego, California, Februari 16-17, 1995.

Electronic Cash on the Internet

Stefan Brands

Centrum voor Wiskunde en Informatica
Kruislaan 413, NL-1098 SJ Amsterdam

Abstract

It is generally realized that the Internet will not be able to offer full-fledged electronic marketplace capabilities without a suitable electronic mechanism for processing payments. The electronic payment mechanism that is presented offers a variety of features that are believed to be particularly appealing in this respect.

To participate, an Internet user must interface to his computer a tamper-resistant device with an ordinary 8-bit microprocessor, typically a PCMCIA card, and install some software. Internet service providers do not need special hardware. Payments can be made off-line, and are untraceable and unlinkable. Multi-party security is guaranteed without parties having to trust other parties. Transaction processing speeds are such that even modestly equipped computers will be able to meet the performance levels required by demanding Internet payment applications. One particularly interesting such application is click-and-pay ability when travelling World-Wide-Web links.

1 Introduction

The Internet is currently witnessing an explosive growth. At the time of writing this paper, an estimated 23 million people in over 130 countries have access to the Internet, and a million more are joining each month. The Internet encompasses more than 15,000 autonomous networks, and over 1,800,000 hosts world-wide. Although the main Internet user groups still reside in business, governmental and educational organizations, the general public is rapidly catching up. Small companies that offer access to the Internet at an affordable price pop up at an accelerating rate. If anything resembles the digital information highway of the future, then it must be the Internet.

It is not surprising that the awareness of the business potential of the Internet is also rapidly growing. By serving as a virtual marketplace, the Internet can dramatically change the way business is conducted. Parties that are thousands of miles apart and have never met will be able to conduct transactions from behind their personal computers. Organizations and individuals will be able to offer information, goods and services by, for instance, creating their own home pages for popular user interfaces to the World-Wide-Web, needing to invest only in the cost of a server. Small companies will be able to compete with large companies, because product information can rapidly be widely disseminated while the associated distribution costs are negligible.

Many of the obstacles that prevent the Internet from offering the capabilities of a full-fledged electronic marketplace have already been removed. Browsing mechanisms, secure methods for downloading information, and user-friendly interfaces, such as Mosaic for the World-Wide-Web, are currently being standardized.

Undoubtedly the biggest of the remaining challenges is to implement an appropriate mechanism for processing Internet payments. It is generally expected that the bulk of the transactions over the Internet will be of low value, involving small payments for access to on-line magazines, reports, newspapers, pictures, shareware, hobbyists' information, and so on. For such purchases, payment mechanisms that are paper-conducted will not be cost-effective, and are hence unacceptable. Without the ability to perform payments by a mere click of a mouse button, Internet users will be reluctant to make small payments when travelling World-Wide-Web links. In the longer run, other services such as video-on-demand and video game rental may become available, but these are still unlikely to exceed a limit of a few dollars per payment. It seems self-evident that the business potential of the Internet can only be exploited to the full by implementing a payment mechanism that is completely electronic.

The purpose of this paper is to present such a mechanism. The presented mechanism offers a variety of features that seem particularly appealing for processing Internet payments, and that are lacking in many of the currently considered proposals. High-speed payments can be made off-line, without being traceable or even linkable, and multi-party security is guaranteed without requiring parties to trust other parties.

This paper is organized as follows. First, in Section 2, the features that have been incorporated into the proposed payment system are described and motivated. The system itself is presented in Section 3. On the basis of three figures, that represent the system gradually being built up, it is described what measures have been taken to ensure that the system has the features listed in Section 2. In Section 4, a mathematical embodiment of the proposed payment system is provided. The analysis of correctness of the embodiment is discussed in Section 5. This is followed by a comparison of the presented embodiment to another embodiment of the system, in Section 6. Next, in Section 7, it is shown how to optimize the described embodiment by using pseudo-randomness. The performance of the resulting embodiment is evaluated in Section 8. As can be seen from the evaluation, the performance levels that can be attained by modestly equipped computers should easily meet the requirements of demanding applications such as click-and-pay ability when travelling Word-Wide-Web links. In Section 9, several extensions are described, including ways to further improve efficiency by trading off with privacy of payments, and to convert between different currencies at payment time. A discussion of ways to improve efficiency by trading off with unlinkability of payments is provided in Section 10.

2 System features

In the past few years a great many Internet payment systems ([1, 11, 13, 14, 15, 20], amongst others) have been proposed that are not based on public-key cryptography. As a result, none of these systems offers multiparty security or privacy of payments (the acclaimed properties of security and privacy, if any, depend completely on trust in a third party, which typically is under control of the currency provider). Although each of these systems certainly has its specific merits, be it in terms of ease of operation or in terms of efficiency, they will be inadequate when used on a large scale, once people have become accustomed to the idea of electronic payments over the Internet. Only systems based on public-key cryptographic techniques (possibly in combination with tamper-resistance) will be able meet the necessary security and privacy levels for large-scale use. For this reason, the system that is presented in this paper is based on public-key cryptographic techniques.

The following description lists the features of the system, and motivates them.

Privacy. Dealing in personal information is a multi-billion dollar industry nowadays. With the advent of a widely used virtual marketplace, an extremely powerful tool becomes available to parties that are interested in gathering personal information. A currency processing mechanism that enables detailed monitoring of the flow and contents of all transactions will certainly be of interest to those parties. Signs indicate, however, that Internet users will be very reluctant to use such a mechanism.

The Internet payment system that is proposed in this paper has been designed such as to guarantee full privacy of payments. This means that the payments of an Internet user are untraceable and unlinkable, even when the bank and the service providers match their databases in an attempt to link or trace payments. Each user can ensure his own privacy of payments, but is not required to do so. The fact that the claimed privacy properties are publicly verifiable may certainly be of help to create public trust in the system, which will be a prerequisite for wide-scale public acceptance of any Internet payment system.

Off-line payments. If an Internet payment system is implemented to serve a relatively small community of Internet users, an on-line payment system may be a satisfactory solution. In this respect, the on-line privacy-protecting payment system [8] that has recently been announced by DigiCash B.V., and which is currently being tested on the Internet, may very well turn out to be appropriate. However, imagine the queueing problems that may arise if such a system is used on a large scale. If the central computer is down, or the communication lines to the central computer are slow or broken, then the whole system lies flat on its back for some period of time.

Another obvious disadvantage is the fact that the service providers will incur the cost of on-line verification. Imagine a popular server, that offers pictures at a few dimes each, having to make an on-line connection to the central computer of the bank for every single payment.

Furthermore, in a privacy-protecting on-line cash system the on-line action by the central computer of the bank cannot consist of a settlement of two accounts, but must necessarily involve the checking of the transaction data against a central database that must be updated at each payment (to prevent double-spending of electronic cash). This implies that the central computer must verify each payment *serially*, because parallel verification of payments will not detect payments that are made at the same time with the same electronic money. Again, one can easily picture the queueing problems when hundreds of thousands of Internet users pay to travel World-Wide-Web links. This problem can by no means be solved by the use of multiple on-line checking centers: to prevent spending of the same electronic money at different places, each payment would have to be recorded by all the checking centers.

The system proposed here has been designed to have off-line payment capability. Internet service providers can accumulate payments, and deposit the aggregate value at the bank at suitable times when network traffic is low.

Multi-party security. Since electronic cash is just digital information, it should be infeasible to spend the same electronic cash multiple times. Although privacy-protecting off-line electronic cash systems have been proposed in which such a fraud will enable the bank to trace the perpetrator afterwards, when the corresponding payment transcripts are deposited, this measure by itself obviously provides an insufficient security level.

Instead, the system that is described in this paper guarantees prior restraint of double-spending electronic cash. A participant in the system is provided by the bank with a tamper-resistant device, typically a PCMCIA card, that must be interfaced to his computer. Although this hardware requirement may seem to be a disadvantage, it is believed that in the longer run the advantages will significantly outweigh the objections. Firstly, off-line systems that do not use tamper-resistance cannot offer prior restraint of double-spending. Secondly, once people get accustomed to paying with electronic cash, they will want the ability to use the same payment system for all their payments. The system that is presented here is portable to many different platforms: the protocols are not hardware dependent. To make payments in public areas, the user can take his tamper-resistant device and insert it into, say, his palmtop computer. Thirdly, in a privacy-protecting Internet payment system without tamper-resistant devices the electronic cash of the user must be stored on his computer, implying that other parties, such as system managers, can spend the electronic money of the user.

Of course, the hardware equipment of the Internet users should be inexpensive, otherwise they will not want to incur the cost associated with opening an Internet bank account. Cryptographic co-processors are currently rather expensive, and probably will remain so for quite a while. This has been taken into consideration in the embodiment of the system that is presented in Section 4: ordinary, widely available, 8-bit micro-processors can be used, such as are typically used for smart cards.

New technology may make tamper-resistant devices more vulnerable to attacks. For this reason, the security of the proposed system is only partly dependent on the tamper-resistance of the user-devices. A mechanism has been built in that ensures that the contents of a compromised tamper-resistant device cannot be double-spent without the owner being traced afterwards by the bank. Since breaking tamper-proofness requires the capabilities of a national laboratory, the attack will not be worthwhile.

It is important that no party in the system needs to trust another party to ensure his security. This has been taken into account in the proposed system. The bank need not trust the service providers and the users; a service provider is guaranteed to get its money from the bank at deposit time if only it follows the payment protocol; no user or service provider can repudiate a payment, due to the use of digital signatures; users can disprove false claims by the bank of having double-spent electronic money; and there are no subliminal channels in the protocols that enable the tamper-resistant devices to leak or receive information.

The acclaimed security properties are publicly verifiable, because no secret algorithms are used. This may surely help in building public trust and hence acceptance.

Efficiency. Convenience of making payments is highly desirable. To make small purchases over the Internet, the actions required of the parties involved in a transaction should be minimal. This pertains not only to the physical efforts required of a party (how much typing must he do to perform the transaction), but also to the speed by which the transaction is processed. Payments should be instantaneous, preferably without requiring interaction. Only then will Internet users be willing to pay make such small purchases as retrieval of a weather forecast or an electronic holiday brochure, or pay for travelling World-Wide-Web links.

In the embodiment that is described in Section 4, any specified amount can be transferred by sending a mere 143 bytes to the service provider; a mouse button click suffices to execute the payment. No interaction is required, making the embodiment ideally suitable for tagging electronic cash to e-mail messages. The required computational effort to prepare a payment is virtually negligible: even if the tamper-resistant device of the user has a simple 8-bit micro-processor, this preparation will not take more then a few hundredths of a second. The service provider must temporarily store 160 bytes for each payment (it must provide this at deposit time), and so can handle over 65,000 payments a day by a 10 Megabyte hard disk, without having to contact the bank. The data of one billion transactions can be stored by the bank on 50 Gigabytes of hard-disk space; if an Internet bank has 100,000 account holders that all make 30 payments a day, it can accommodate the payments of almost one year before it has to refresh its public key (or buy new hard disk space).

Open system. The system that is presented in this paper is an open system. Internet users and service providers can join or leave the system at any time without the remaining participants needing to know about this. Different Internet banks can join at any time, without this preventing account holders at different banks from making secure off-line payments (see Section 9). The bank of the other party need not even be known to the parties involved in a transaction. Since the same system can be implemented for other payment platforms, Internet users can use the electronic cash that they use on the Internet also in other environments.

Other features. Another feature that is offered by the proposed system is conversion between different currencies at payment time. It is unlikely that the introduction of one global Internet currency will succeed for a payment system that is to be used by tens of millions of Internet users.

Further features that are offered, but are not described here, are: anonymous accounts can be used, while maintaining the ability for the bank to trace double-spenders; off-line transferability of payments can be incorporated if one is willing to use more sophisticated micro-processors for the tamper-resistant devices of the users; and the same electronic tokens can be used to represent digital pseudonyms.

In particular, the employed techniques can be used to implement mechanisms for off-line processing of credentials other than electronic cash. Credentials gathered by a user can be maintained and updated in one electronic token. The user can demonstrate possession of subsets of credentials, and a variety of relations between different credentials, without revealing *any* additional information.

For a description of the application to anonymous accounts, transferability, digital pseudonyms and credentials, the interested reader is referred to [4].

3 The system

In this section, the payment system is presented. The actions of the parties in the system are described in terms of secret and public keys, and signatures. In the mathematical embodiment that is described in Section 4 appropriate mathematical choices will be substituted for the objects and actions that are treated here only generically.

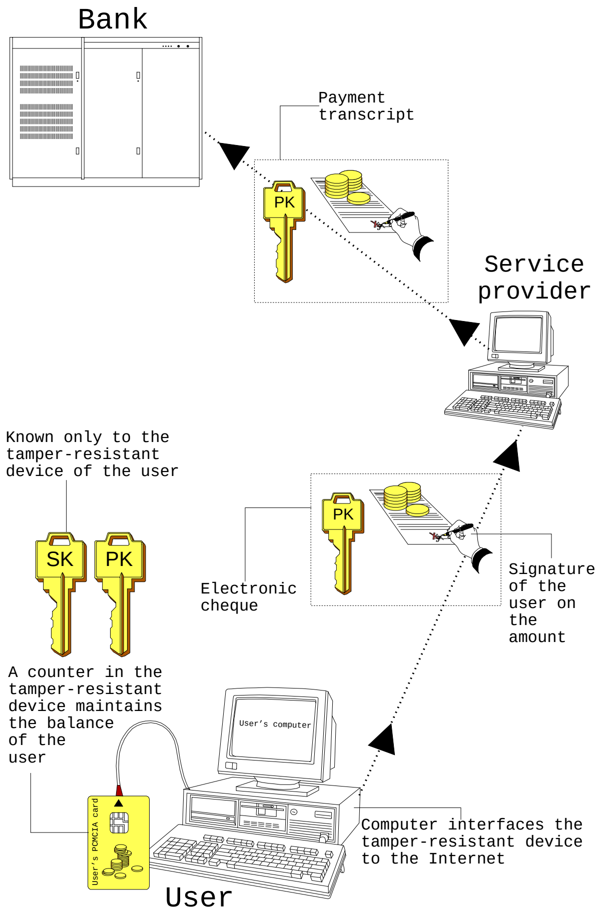
Some familiarity with the basics of public-key cryptography should suffice to follow the discussion. To increase readability, the discussion has been divided into three subsections: first, a stripped version of the system is discussed, to introduce the system; then, it is shown how to incorporate security; and, finally, privacy of payments is added.

The system can be seen as a hybrid form of three systems, the first of which has been proposed by Even, Goldreich and Yacobi [12], the second by Bos and Chaum [2], and the third by Chaum, Fiat, and Naor [9]. Each of these systems by itself would not be appropriate for use on the Internet, though. The first system [12] is an off-line cheque system that offers no privacy of payments, and whose security depends *completely* on tamper-resistant devices; the second system [2] is a privacy-protecting off-line cheque system whose security depends completely on tamper-resistance; and the third system [9] is a privacy-protecting coin system that only offers traceability of double-spent coins after the fact.

3.1 A simplified version

Figure 1 represents a simplified version of the system. Depicted is one of each of three types of participants. At the top left, the computer of the Internet bank is depicted; at the right-hand side the computer of an Internet service provider is shown; and the equipment of an Internet user is depicted in the lower left corner. The computer of the user is interfaced to a tamper-resistant device, depicted here for explicitness as a PCMCIA card. For the moment, the user's computer will be assumed to play a passive role, in that it only serves as a suitable interface between the tamper-resistant device and the Internet.

The tamper-resistant device has been issued to the user by the Internet bank. It keeps track of the cash balance held by the user, by means of a counter, and can perform computations. The security will only be partly dependent on the tamper-resistance of the device: as will be shown later, the profit that an attacker can make by breaking open his tamper-resistant device, and extracting its contents (which will require



Bank
Payment transcript
PK
Service provider
Known only to the tamper-resistant device of the user
SK
PK
Electronic cheque
PK
Signature of the user on the amount
A counter in the tamper-resistant device maintains the balance of the user
User's computer
User's PCMCIA card
Computer interfaces the tamper-resistant device to the Internet
User

Figure 1: A simplified symbolic representation.

the capabilities of a national laboratory), will hardly be worthwhile.

The tamper-resistant device increases the counter at withdrawal time by the amount of money that is withdrawn by the user from his account at the bank, and decreases it at payment time by the amount of electronic money that is transferred from the user to the service provider.

To transfer electronic money from the user to the service provider such that the service provider can verify its validity off-line, electronic money must be recognizable, without help of the bank, as having been created by the bank. To this end, a secret key has been installed by the Internet bank into the memory of the tamper-resistant device. When a specified amount is to be transferred to the service provider, the corresponding public key is sent by the tamper-resistant device, via the computer of the user, to the service provider, together with a digital signature of the tamper-resistant device on the amount. The secret resp. public key is shown in Figure 1 as a key labeled by SK resp. PK. Since the user does not know the secret key of the tamper-resistant device, this signature can only be produced when the tamper-resistant device cooperates. Of course, it has been programmed by the bank to do so only when its counter exceeds the specified amount. After having computed the signature, the tamper-resistant device decreases the counter by the amount that it signed.

Because the public key has been transferred along to the service provider, the service provider can verify the digital signature on the amount. If it is correct, the service provider accepts and provides the requested service to the user.

At a suitable moment later on, the service provider can deposit the payment transcript, consisting of the public key and the signature on the amount, at the bank. This payment transcript is indicated in the figure by the rectangle across the communication line between the service provider and the bank. The bank in turn verifies the payment transcript, by verifying the signature on the amount. If the signature is valid, the bank credits the account of the service provider by the amount that is specified by the signature.

Note that the bank need not issue tamper-resistant devices to the service providers, which clearly is interesting from an economic point of view.

3.2 Incorporating multi-party security

The system that has been outlined thus far is not secure. Since anyone can generate a pair consisting of a public key and a matching secret key (at least in the currently known practical schemes), the public key that is sent to the service provider at payment time must be recognizable by the service provider as being a *valid* public key. Valid in this context means that only some tamper-resistant device knows the corresponding secret key. Because the secret information held by each tamper-resistant device must be unique for security reasons (otherwise, the compromise of one tamper-resistant device is fatal to the system), it is impractical for service providers to maintain a list of all valid public keys, in particularly so because the system is supposed to be open.

The solution for this key management problem consists of requiring the user at payment time to also provide the service provider with a certificate of the bank. This certificate is a digital signature of the bank on the public key of the tamper-resistant device. It is represented in Figure 2 by the object denoted by cert(PK). The certificate enables service providers to verify the

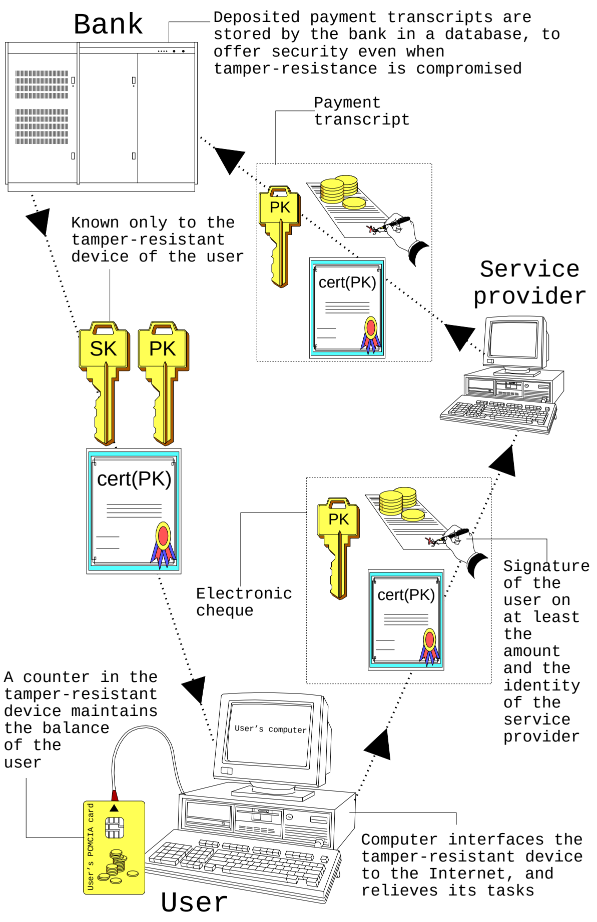

Bank
Deposited payment transcripts are stored by the bank in a database, to offer security even when tamper-resistance is compromised
Payment transcript
PK
cert(PK)
Known only to the tamper-resistant device of the user
Service provider
SK
PK
cert(PK)
PK
cert(PK)
Electronic cheque
Signature of the user on at least the amount and the identity of the service provider
A counter in the tamper-resistant device maintains the balance of the user
User's computer
User's PCMCIA card
Computer interfaces the tamper-resistant device to the Internet, and relieves its tasks
User

Figure 2: Symbolic representation with security.

validity of public keys off-line, if only they store the public key of the bank. Of course, the genuineness of the public key of the bank is very important, and so anyone should be able to convince himself of its validity. Hereto, the bank should widely disseminate it in a variety of ways. To not distract from the essentials, the secret and public key of the bank are not depicted in Figure 2.

The system described thus far is similar to that presented by Even, Goldreich and Yacobi [12], with the distinction that the service provider is not represented by a tamper-resistant device that increases its balance by the received amount; instead, the service provider deposits the payment transcript to get his account credited by the bank.

If the user manages to break open his tamper-resistant device, then he can forge money by computing the signatures at payment time by himself; the financial gain is due to the fact that the counter in effect is disabled. To limit the financial damages resulting from such an attack, the bank could take the following simple measure. It keeps track for each account holder how much money is withdrawn, and how much is spent. The bank can easily do this, since each deposited payment transcript uniquely identifies the tamper-resistant device that computed the signature (because its public key is included in the payment transcript), and hence identifies the account holder that made the payment.

However, this particular method for providing increased security will not be used here, since it cannot be extended to offer privacy of payments. Instead, another method is used, which is not as efficient but has the advantage that it can be extended to offer full privacy of payments (as will be shown in the next subsection). The method consists of the bank requiring *each* payment to be made with respect to a unique public key. Each tamper-resistant device is given a multitude of secret keys by the bank, and is programmed to assist for each secret key only once in computing a signature on an amount (the consequences of using another limit than 1 are discussed in Section 10). For practicality, such secret keys (with corresponding certified public keys) can be downloaded from the bank by means of a so-called certificate issuing protocol between the bank and the tamper-resistant device. Since the communication from the bank to the tamper-resistant device is through the computer of the user, the certificate issuing protocol must prevent the user from learning the secret key.

This measure offers a high level of security for the bank, for the following reason. An attacker, who manages to extract the secret keys of his tamper-resistant device, can only make an undetectable profit by using each of the keys once to compute a signature, because the bank knows (and of course keeps track of) the set of secret keys held by each tamper-resistant device. To continue making a profit without being detected, the attacker must keep on withdrawing new certified keys from the bank. By imposing a fairly low maximum on the amount that may be signed with respect to one certified public key, and limiting the number of certified keys that may be downloaded in a certain time span at withdrawal time, the profit that an attacker can make from breaking his tamper-resistant device is severely limited. As a result, the expected profit will hardly make the attack on the tamper-resistant device worthwhile, given the current state of tamper-proofing technology. It is this measure, which still applies after we have incorporated full privacy of payments, that gives the bank the added security.

Note that if an attacker manages to break open, and extract the secret keys from, the tamper-resistant device of *another* party, then double-spent certified keys will be traced by the bank to the incorrect party. (In the embodiment described in Section 4, additional measures are taken that make this attack extremely difficult.) The bank should hence require the participants in the system to protect their tamper-resistant devices, and inform the bank in case of theft. The secret keys of the stolen device can then be black-listed; by requiring service providers to verify payments on-line against this black-list (which they may download from the bank), no significant damage can be done. Note that the requirement of having to report a stolen device also means that the person traced by the bank can be held, at least to some extent, liable for the detected fraud.

The information that is transferred from the user to the service provider can be though of as being an electronic cheque: the public key that is part of this information can be used once only, and the amount that is being transferred is filled in at payment time, by the signature that is made with respect to the public key.

To prevent different service providers from depositing the same payment transcript multiple times without knowing who is liable (a service provider, or the user that managed to extract the associated secret key from his device?), the signature at payment time must be made not only on the amount, but also on some information that is uniquely associated with the service provider, and such information as date and time of the transaction. If the service providers are honest, then the information that is signed by the payer's device clearly is different for each payment. The bank can hence in all fairness demand this to be always the case. By also requiring a description service provider to be included, the same payment transcript cannot be deposited by two service providers by coincidence.

The system described thus far offers a high level of security for the service providers and the bank. A service provider that verifies at payment time the certificate on the transferred public key, and the signature with respect to that public key on the amount, is guaranteed to be credited at deposit time with the specified amount. The required use of a unique certified key for each payment ensures that the expected profit of an attacker, who manages to extract the secret key of a tamper-resistant device, will be severely limited in practice.

The user is also guaranteed a high level of security. Service providers are not able to double-deposit electronic cheques of users, because two identical payment transcripts indicate to the bank that the service provider(s) are cheating. Because only the user has access to his tamper-resistant device, not even parties that have complete access to the computer of the user, such as system managers in a LAN, can spend the electronic money of the user. If suitable PIN or biometric verification on the device is used, then this holds even if the device is stolen or found to be lying around somewhere. Non-repudiation of payments can be ensured by letting the service provider at payment time return a digital signature on information specific to the transaction (such as date and time, identity of the service provider, and the transferred amount). To not distract from the essentials in the figures, this returned digital signature and the corresponding certified key pair of the service provider are not displayed in Figure 2.

Until now, the role of the computer of the user has been a passive one. It merely served to interface his tamper-resistant device to the Internet. By letting the computer play an active role, part of the burden of the tamper-resistant device can be transferred to the computer. It is immediately clear that the computer, instead of the tamper-resistant device, can store the public keys and the certificates. Additional improvements may be achieved with respect to the computation of the signature on the amount, since all currently known digital signature schemes require a significant computational effort. As will be appreciated, the mathematical embodiment that is presented in Section 4 is such that even this burden is almost completely moved to the computer of the user (except for an insignificant, but inevitable, part): the tamper-resistant device never needs to perform public-key cryptographic operations.

3.3 Incorporating privacy of payments

There is one important issue that has not been addressed yet: privacy of payments. The high level of security for the bank has been achieved at the expense of a complete loss of privacy of payments. Even if the bank would be willing to rely for its security only on the tamper-resistance of the issued devices, it still has the full tracking abilities of each and every payment, no matter how small. Although the privacy issue is partly a political one, various signs indicate that Internet users are unwilling to accept for wide-scale use a privacy-compromising payment mechanism.

As will be shown next, privacy of payments can be incorporated without decreasing security and efficiency. A symbolic representation of the final system,

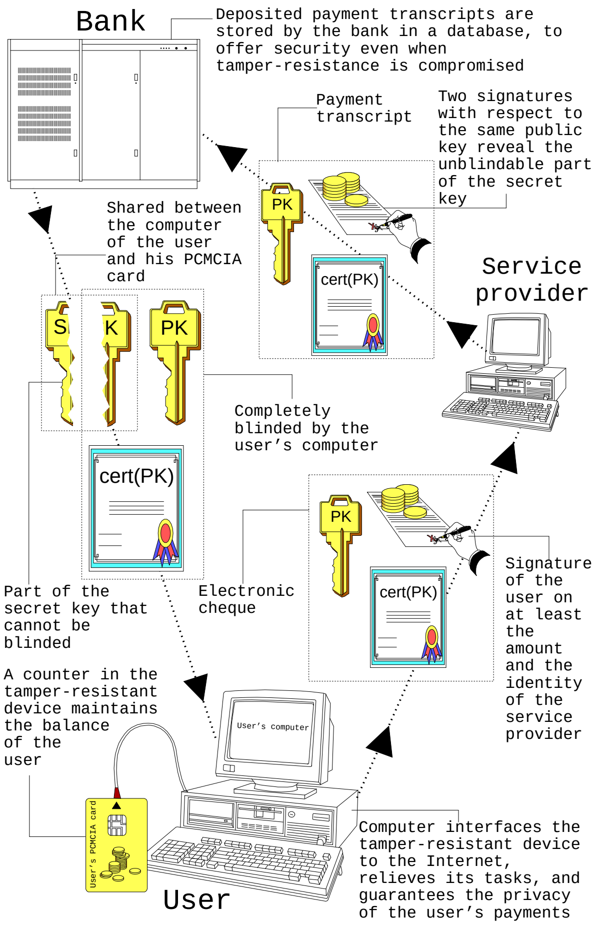

Bank
Deposited payment transcripts are stored by the bank in a database, to offer security even when tamper-resistance is compromised
Payment transcript
Two signatures with respect to the same public key reveal the unblindable part of the secret key
PK
cert(PK)
Service provider
Shared between the computer of the user and his PCMCIA card
S
K
PK
cert(PK)
Completely blinded by the user's computer
PK
cert(PK)
Signature of the user on at least the amount and the identity of the service provider
Part of the secret key that cannot be blinded
Electronic cheque
A counter in the tamper-resistant device maintains the balance of the user
User's computer
User's PCMCIA card
User
Computer interfaces the tamper-resistant device to the Internet, relieves its tasks, and guarantees the privacy of the user's payments

Figure 3: Symbolic representation of the final system.

offering both security and privacy, is shown in Figure 3.

Consider first the well-known approach of letting the bank issue the tamper-resistant devices in such a manner that it does not know which user receives which device. In this way, the secret and public keys can no longer be associated by the bank with users, only with tamper-resistant devices. This approach, which has been applied in for instance the Mondex scheme [1, 19] (which is based only on conventional cryptographic techniques), has the following significant drawbacks:

1. It actually cannot be said to offer privacy at all, for the following two reasons. Firstly, it will be difficult to run the distribution process in a manner that randomly distributes the tamper-resistant devices over sufficiently many users, and users will still have to trust the provider to properly conduct the distribution process.

   Secondly, all payments made with the same tamper-resistant device are still linkable by the bank at deposit time, due to the fact that the bank knows which public keys correspond to which tamper-resistant device. This implies that if a tamper-resistant device is ever linked to the identity of its holder, for instance because the holder is required to identify himself in a certain transaction, then not only is the privacy of all his future payments lost, but also that of all the payments that he conducted in the past. Users will not be willing to increase the unlinkability of their payments by swapping devices with other users from time to time, assuming PIN verification on the devices.

2. The security for the bank also reduces significantly. The secret keys that are stored in a tamper-resistant device can now be used multiple times by an attacker that manages to break open the device. Due to the fact that the tamper-resistant device is not associated to the user, the bank cannot trace the attacker afterwards. As mentioned before, a successful attacker of a tamper-resistant device can realistically hope to make a large profit by automating the multiple spending of the compromised keys, before service providers will refuse his payments due to blacklisting.

   In sum, the compromise of one tamper-resistant device by an attacker certainly does not limit the damage to that particular device, even though each tamper-resistant device uses unique keys. (It is worthwhile to note that this fact also applies to secret-key systems, and is often overlooked.)

A system that guarantees privacy by using anonymous accounts has the same problems.

As realized first by Chaum [6], the most satisfactory way to ensure privacy of payments is to destroy the relation between the information that the bank sees when it certifies a message (which in the system presented here is a public key), and the information that is transferred to the service provider in the corresponding payment protocol. Because it does not make sense to let the bank do this (one would need to trust it in that case, but then there would be no point in incorporating privacy-protecting measures in the first place), the certificate receiver should be able to perform the destruction by himself. In that way, users can guarantee their own privacy of payments, without having to trust other parties.

As Chaum further demonstrated, by a particular example [6], such *blind* issuing protocols can be designed by using cryptographic technique. To explain the concept of blinding a signature issuing protocol, consider the set of all pairs consisting of a message and a corresponding signature. Blinding the issuing protocol means that the receiver, when performing the protocol with the signer, ends up with one pair from this set, such that the signer has no information whatsoever about the pair. It is easy to see that this technique can guarantee that payments cannot be traced to withdrawals (not even if service providers cooperate with the bank), and payments of the same user cannot be linked (so identification in one payment does not compromise the privacy of other payments).

In the system outlined here, it is not immediately clear that the user can indeed guarantee his privacy in this way. Because the bank must ensure that certified keys are downloaded by the user in such a way that only his tamper-resistant device learns the corresponding secret keys, the certificate issuing protocol actually is performed between the tamper-resistant device and the bank (with the user's computer merely acting as an interface). Since the tamper-resistant device has been issued to the user by the bank, the user can trust his device no more than it can trust the bank to faithfully performing the blinding. Chaum [7] also argued that the user-controlled computer may be used to moderate the communication between the tamper-resistant device and the bank in such a way that the blinding is performed correctly. Moreover, the computer may be able to ensure that the bank and the tamper-resistant device cannot send messages to one another by using a subliminal channel in the certificate issuing protocol. This latter property also applies to the communication between the service providers and the tamper-resistant device. The embodiment in Section 4 is such that the computer of the user can moderate all communication.

In all, we have actually identified three different roles that are performed by the user's computer: not only is it used to interface the tamper-resistant device to the Internet, it can also take over part of the burden of the tamper-resistant device (both computation and storage), and serve to moderate the communication between the tamper-resistant device and the outside world (service providers, the bank) to ensure privacy.

As we have seen, the blind signature issuing technique solves the privacy problem. Users can guarantee themselves that their payments are untraceable and unlinkable; they do not have to trust other parties for this. The particular combination of measures that has been discussed thus far has been proposed first, in the form of a particular mathematical embodiment, by Bos and Chaum [2].

The problem of lost security is *not* addressed by the blind signature technique, since the blinding ensures that multiple spending of compromised keys cannot be traced to a tamper-resistant device. To overcome this, a very special kind of blind issuing protocol must be used, called a one-show blind signature protocol. The use of such a protocol can ensure that a certified public key, which has been issued by the bank in a blinded way, can be traced to party that it has been issued to *if and only if* more than one signature is computed with respect to the key. This idea was proposed by Chaum, Fiat, and Naor [9], and was shown to be viable by an exemplary electronic coin system.

The one-show blind signature idea consists of letting the bank certify the public key of a user in such a way that the user can perfectly blind the public key and the certificate thereon, but not part of the corresponding secret key. This situation is depicted in Figure 3 by a secret key that has been separated into two parts, and a rectangle around the objects that can be fully blinded by the user. Correspondingly, the signature scheme employed by the user must be such that one signature does not reveal this part of the secret key, whereas two signatures do. As will be clear, the one-show blind signature technique actually consists of designing two different protocols, one for issuing and one for showing, that must act securely in concert.

Observe that in Figure 3 the unblindable part of the secret key is indicated to be shared between the computer of the user and his tamper-resistant device. This additional measure ensures that attempts of the bank to falsely accuse a user of having spent the same keys multiple times, can be (mathematically) disproven by that user. Why this measure actually works to offer protection against such framing will be explained in greater detail in Subsection 4.2. Suffice it to say here that the unblindable part of the secret key may be held entirely by the tamper-resistant device in case such a mathematical disproof is not believed to be necessary (it may suffice for the user to disprove the false claim by showing that his tamper-resistant device is still intact).

4 A mathematical embodiment

A mathematical embodiment of the proposed system will now be presented. The described embodiment is based on the restrictive blind signature issuing technique in combination with the representation problem in groups of prime order, both of which I introduced in [3] to construct efficient off-line coin systems. Using the cut-and-choose technique of Chaum, Fiat and Naor [9] to obtain the one-show blind signature property would have lead to inefficient systems, even more so when combined with techniques for implementing the property of prior restraint of double-spending.

Additional techniques that have been applied are due to, amongst others, Chaum and Pedersen [10], Okamoto [16], Okamoto and Ohta [17], and Schnorr [18]. Chaum and Pedersen described protocols embodying the concept of moderation described in [7]; the signature protocol that they presented as part of these protocols (an "ordinary" blind signature protocol) has been used as the basis of a restrictive blind signature issuing protocol. The general technique of Okamoto and Ohta [17] to design payment protocols has been applied to a protocol of Okamoto [16] in order to design the payment protocol. Finally, the Schnorr identification scheme has been used to ensure that the tamper-resistant device, when signing an amount, does not leak information that enables the user to make other signatures, with respect to the same certified public key, on his own.

A fair amount of background in public-key cryptography will be necessary to understand the mathematical exposition that now follows. At this point, non-cryptographically oriented readers may wish to skip over to Section 8, in which the performance of this embodiment is evaluated.

4.1 The objects

In accordance with the presentation in Section 3, it will now be shown how the secret and public keys, and the signature schemes of the tamper-resistant device and the bank, are embodied mathematically.

All arithmetic is performed in a group G_q of prime order q for which polynomial-time algorithms are known to multiply, determine equality of elements, test membership, randomly select elements, and for which no feasible methods are known to compute discrete logarithms. Various types of such groups are known in the literature, and so no particular choice will be made.

The bank generates independently at random three numbers $g_0, g_1, g_2 \in G_q \setminus \{1\}$, and a number $x \in \mathbb{Z}_q$. The bank also determines a collision-free hash function $\mathcal{H}(\cdot)$ that maps its inputs to $\mathbb{Z}_{2^k}$, where k is an appropriate security parameter. The function $\mathcal{H}(\cdot)$ is chosen such as to (presumably) make the Schnorr signature scheme [18] secure.

The objects depicted in Figure 3 are embodied as follows:

PK

A public key that is issued by the bank to the user is a pair $(h_i', a_i') \in G_q \times G_q$.

cert(PK)

The number x is the secret key of the bank, and the corresponding public key consists of the tuple (g_0, g_1, g_2, h) and the descriptions of G_q and $\mathcal{H}(\cdot)$. A certificate of the bank on the public key (h_i', a_i') of the user is a triple (z_i', c', r') such that

$$c' = \mathcal{H}(h_i', a_i', z_i', g_0^{r'} h^{-c'}, (h_i')^{r'} (z_i')^{-c'}).$$

SK

The secret key that corresponds to the public key (h_i', a_i') of the user is a pair $((\beta_1, \alpha_1), (\beta_2, \alpha_3))$, such that $h_i' = g_1^{\beta_1} g_2^{\alpha_1}$ and $a_i' = g_1^{\beta_2} g_2^{\alpha_3}$. This secret key is shared between the user (computer) and his tamper-resistant device in the following way. The numbers α_1 and α_3 are known to the user; the number β_1 is of the form $\alpha_1(x_{i1} + x_{i2}) \bmod q$, where α_1 and x_{i2} are known only to the user, and x_{i1} is known only to the tamper-resistant device; and the number β_2 is of the form $\alpha_1 w_i + \alpha_2 \bmod q$, where α_2 is known only to the user, and w_i only to the tamper-resistant device.

S

The bank uses the number $h_i g_2$ as input to the certificate issuing protocol (which will be specified in the next subsection), where $h_i = g_1^{x_{i1}+x_{i2}}$. Observe that h_i is a joint public key of the user and his tamper-resistant device: neither the user nor the tamper-resistant device (or the bank, for that matter) know the corresponding secret key.

By raising $h_i g_2$ to the power α_1 (resulting in h_i'), and performing appropriate additional actions, the user can derive, from the information revealed by the bank, a certificate on (h_i', a_i'). If α_1 is randomly chosen, then this guarantees that h_i is uncorrelated to h_i'. At the same time, the user will not be able to do *more* than this blinding. He cannot set h_i' to $h_i^{\alpha_1} g_1^{s_1} g_2^{s_2}$, for some known pair $(s_1, s_2) \neq (0, 0)$, such that he can derive a certificate on (h_i', a_i'). In other words, the unblindable part of the secret key that corresponds to public key (h_i', a_i'), and that is uniquely associated with the user, is equal to $\beta_1 \alpha_1^{-1} \bmod q$. If $\alpha_1 \neq 0 \bmod q$, then this is equal to $x_{i1} + x_{i2} \bmod q$. Note that if $\alpha_1 = 0 \bmod q$, then $h_i' = 1$, *i.e.*, this fact is indicated by the value of the public key.

A signature with respect to public key (h_i', a_i') on a number, `spec` (which depends amongst others the amount that is being transferred), is a pair (r_1', r_2) such that

$$g_1^{r_1'} g_2^{r_2} (h_i')^{-d} = a_i',$$

where $d = \mathcal{H}(h_i', a_i', \texttt{spec})$. Note that:

- the tamper-resistant device and the computer of the user together possess the secret information to compute a signature on any `spec`: $r_1' = d\beta_1 + \beta_2 \bmod q$ and $r_2 = d\alpha_1 + \alpha_3 \bmod q$.
- due to the special form of β_1 and β_2, the computer and the tamper-resistant device can jointly compute r_1 and r_2 such that the part contributed by the tamper-resistant device is uncorrelated to the signature. This prevents subliminal channels from the tamper-resistant device to the service provider. Namely, the tamper-resistant device first computes $r_1 = dx_{i1} + w_i \bmod q$, and the computer computes r_1' by multiplying r_1' by α_1 and adding $d\alpha_1 x_{i2} + \alpha_2$ modulo q to it. Note that the computer can compute r_2 by itself.
- if (r_1', r_2) is a signature on `spec`, and (r_1'', r_2') is a signature on a number that differs modulo q from `spec`, then $x_{i1} + x_{i2} \bmod q$ can be computed from the signatures:

  $$x_{i1} + x_{i2} = (r_1' - r_1'')/(r_2 - r_2') \bmod q,$$

  as long as $\alpha_1 \neq 0$. This is equal to $\beta_1 \alpha_1^{-1}$, the unblindable part of the secret key, and so from this the bank can compute h_i and trace the party that signed twice with respect to (h_i', a_i').

4.2 The actions

The mathematical embodiments of the keys and the signatures of the respective parties have been described in the preceding subsection. The description that now follows focuses on the cryptographic protocols needed to obtain and issue these keys and signatures.

To facilitate readability, from now on the Internet bank will be denoted by $\mathcal{B}$, the user by $\mathcal{U}_i$, and the service provider by $\mathcal{S}_j$. The computer of $\mathcal{U}_i$ is denoted by $\mathcal{C}_i$, and his tamper-resistant device by $\mathcal{T}_i$. In the descriptions of the protocols, it is implicitly assumed that a party halts the execution of a protocol if it does not accept at a certain stage. Assignment is denoted by the ":=" symbol.

All random numbers that are generated in the protocols are assumed to be truly random. In Section 7, it is shown that the use of pseudo-random generators by $\mathcal{T}_i$ and $\mathcal{C}_i$ can lead to a significant performance improvement with respect to storage for $\mathcal{C}_i$ and computational effort for $\mathcal{T}_i$.

Opening an account. $\mathcal{U}_i$ installs on his computer, $\mathcal{C}_i$, a software program for performing the protocols. As explained in Subsection 3.3, this software program protects the privacy of payments of $\mathcal{U}_i$. Obviously, it need not be trusted by $\mathcal{U}_i$: $\mathcal{U}_i$ may write it himself, download it from public domain, or buy it on the free market. The software program contains the public key of $\mathcal{B}$ (which may have to be downloaded and refreshed on occasion in case $\mathcal{B}$ builds expiration dates into certificates).

When $\mathcal{U}_i$ opens an account with $\mathcal{B}$, the following procedure takes place.

Step 1. $\mathcal{C}_i$ generates independently at random a secret key $x_{i2} \in \mathbb{Z}_q$, and stores it. $\mathcal{C}_i$ sends $g_1^{x_{i2}}$, which will from now on be denoted by h_{i2}, to $\mathcal{B}$, together with an appropriate verifiable description of the identity of $\mathcal{U}_i$. (Alternatively, $\mathcal{U}_i$ may be required to show up in person at some local office. Apart from the identification purpose, this also has the advantage that the tamper-resistant device need not be send by mail.)

Step 2. $\mathcal{B}$ verifies that the description of the identity corresponds to $\mathcal{U}_i$. It then generates independently at random a secret key $x_{i1} \in \mathbb{Z}_q$ for $\mathcal{U}_i$. $\mathcal{B}$ lists this number in its so-called account database, together with at least a balance variable that keeps track of the amount of money that $\mathcal{U}_i$ has in its account with $\mathcal{B}$, and the description of $\mathcal{U}_i$'s identity.

$\mathcal{B}$ then issues to $\mathcal{U}_i$ a tamper-resistant device $\mathcal{T}_i$ which has stored in non-volatile memory at least the following items: the numbers x_{i1} and g_1, and a description of G_q; code to perform its role in the protocols; and a counter variable, from now on denoted by `balance`, that keeps track of the amount of money that is held by $\mathcal{U}_i$.

$\mathcal{B}$ makes $g_1^{x_{i1}}$, which will from now on be referred to as h_{i1}, known to $\mathcal{U}_i$; this is the public key of $\mathcal{T}_i$. $\mathcal{B}$ then computes $h_i = h_{i1} h_{i2}$ (the joint public key of $\mathcal{T}_i$ and $\mathcal{U}_i$) and stores h_i in his account database along with its other information on $\mathcal{U}_i$. Observe that $\mathcal{B}$ does not know the joint secret key, $x_{i1} + x_{i2} \bmod q$, of $\mathcal{T}_i$ and $\mathcal{U}_i$.

Finally, $\mathcal{B}$ computes $(h_i g_2)^x$, which will henceforth be denoted by z_i, and makes z_i known to $\mathcal{U}_i$.

Step 3. $\mathcal{U}_i$ installs h_{i1}, h_i and z_i in the software program that runs on $\mathcal{C}_i$.

If $\mathcal{U}_i$ later on double-spends a certified key (for which he first needs to break open his tamper-resistant device), then $\mathcal{B}$ will be able to trace him by computing h_i.

The secret keys of $\mathcal{T}_i$ and $\mathcal{U}_i$ serve different purposes:

- The secret key, x_{i1}, of $\mathcal{T}_i$, prevents $\mathcal{U}_i$ from computing by himself signatures with respect to the joint public key (and blinded forms thereof). This ensures that $\mathcal{U}_i$ cannot double-spend certified keys that have been issued by $\mathcal{B}$.

- The secret key, x_{i2}, of $\mathcal{U}_i$ serves to prevent $\mathcal{B}$ from falsely accusing him of having double-spent a certified key. As explained, if $\mathcal{U}_i$ manages to break open $\mathcal{T}_i$ and extract its secret key then he can spend the same certified keys over and over again. After the deposits of the corresponding payment transcripts, $\mathcal{B}$ will then be able to compute the secret key corresponding to the joint public key of $\mathcal{T}_i$ and $\mathcal{U}_i$, and trace $\mathcal{U}_i$ by searching its account database for the matching public key. If $\mathcal{U}_i$ follows the protocols and does *not* double-spend, then he never leaks information that enables $\mathcal{B}$ to feasibly compute his share of the joint secret key. Therefore, $\mathcal{U}_i$ can disprove a false claim by $\mathcal{B}$ because $\mathcal{B}$ is not able to show the secret key, x_{i2}, of $\mathcal{U}_i$. Vice versa, the fact that $\mathcal{B}$ knows this joint secret key (and hence the share of $\mathcal{U}_i$) is a proof that $\mathcal{U}_i$ has double-spent.

  The secret key of $\mathcal{U}_i$ also ensures that an attacker, that steals his tamper-resistant device, and manages to extract its contents, will not be able to double-spend the certificates. For that, it must know x_{i2}.

For even greater security. $\mathcal{U}_i$ can remember (part of) x_{i2} in the form of a password, and type it in on his computer only when a payment session is to be started. In that case, the task of an attacker to (double-)spend the certified keys of $\mathcal{U}_i$ is extremely difficult.

The withdrawal protocol. Because $\mathcal{T}_i$ keeps track of the balance of $\mathcal{U}_i$ by means of a counter, $\mathcal{U}_i$ should only be allowed to pay an amount when the balance at payment time exceeds the amount to be paid; otherwise, $\mathcal{U}_i$ can pretend to have lost his tamper-resistant device, once the balance has become negative. Therefore, the balance in the tamper-resistant device will on regular occasions have to be up-dated by means of a withdrawal protocol.

Various designs for a suitable withdrawal protocol are conceivable. Note that no public-key cryptographic techniques are needed, since the contents of $\mathcal{T}_i$ are known to $\mathcal{B}$; the withdrawal protocol can hence be based on conventional cryptographic techniques, which are much more efficient.

In the particular realization of a withdrawal protocol that is shown here, $\mathcal{T}_i$ is assumed to have in common with $\mathcal{B}$ a secret key `key`. This secret key, and a sequence number, `seq`, (which has been set to some initial value, such as zero), have been stored by $\mathcal{B}$ before issuing $\mathcal{T}_i$ to $\mathcal{U}_i$. In addition, the description of a one-way function $f(\cdot)$ has been stored by $\mathcal{B}$ in $\mathcal{T}_i$; this function may take the form of a block cipher, such as DES in encipherment mode, taking the secret key as the DES-key and the additional arguments as input. The function $f(\cdot)$ may even be kept secret by $\mathcal{B}$, for greater security. Of course, in practice $f(\cdot)$ may be deterministically related to $\mathcal{H}$, and `key` to x_{i1}, for greater storage efficiency in $\mathcal{T}_i$.

To withdraw an amount, `amount`, $\mathcal{T}_i$ and $\mathcal{B}$ perform the following withdrawal protocol (see Figure 4), with $\mathcal{C}_i$ acting merely as interface:

Step 1. $\mathcal{B}$ decreases the balance, `balance`', of $\mathcal{U}_i$ by `amount`. It then increases `seq` by one, and transfers $v := f(\texttt{key}, \texttt{seq}, \texttt{amount})$ to $\mathcal{T}_i$, by sending it to $\mathcal{C}_i$.

Step 2. $\mathcal{T}_i$ receives v from $\mathcal{C}_i$. It then computes $f(\texttt{key}, \texttt{seq}, \texttt{amount})$, and compares it for equality with v. If equality holds, it increases `seq` by one, and `balance` by `amount`.

Although another user cannot transfer money from the account of $\mathcal{U}_i$ with $\mathcal{B}$ to his own tamper-resistant device, he can decrease the balance of $\mathcal{U}_i$ with $\mathcal{B}$ in this manner. Hence, $\mathcal{T}_i$ should first identify itself to $\mathcal{B}$ before the protocol is executed. Hereto, a similar protocol, with the roles of $\mathcal{T}_i$ and $\mathcal{B}$ interchanged, can be used.

Remarks. 1. To prevent $\mathcal{B}$ from using a subliminal channel to send information to $\mathcal{T}_i$, $\mathcal{C}_i$ should perform a more active role. To ensure that the number

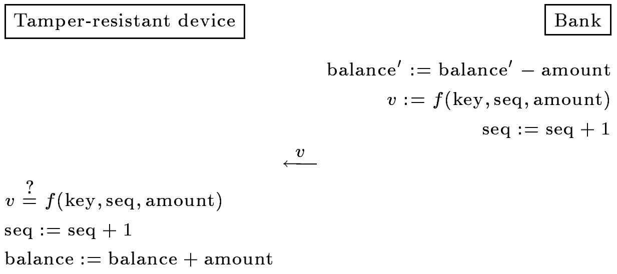

Tamper-resistant device
Bank
$\text{balance}' := \text{balance}' - \text{amount}$
$v := f(\text{key}, \text{seq}, \text{amount})$
$\text{seq} := \text{seq} + 1$
$\xleftarrow{v}$
$v \overset{?}{=} f(\text{key}, \text{seq}, \text{amount})$
$\text{seq} := \text{seq} + 1$
$\text{balance} := \text{balance} + \text{amount}$

Figure 4: Withdrawal protocol (computer interface not shown).

transferred by $\mathcal{B}$ in Step 1 cannot serve as a subliminal channel, $\mathcal{C}_i$ should require $\mathcal{T}_i$, before passing v on to it, to provide a commit on $f(\texttt{key}, \texttt{seq}, \texttt{amount})$. After $\mathcal{C}_i$ has passed v on to $\mathcal{T}_i$, $\mathcal{T}_i$ must then open the commit to reveal that has been able to compute v all along. For efficiency, the commitment function can be based on a block-cipher, taking for instance $f(\texttt{key}, \texttt{seq}, \texttt{amount})$ and a random number as inputs.

2. The increase of the sequence number, $\texttt{seq}$, serves to prevent a replay attack by $\mathcal{U}_i$. The increment by one has been chosen for explicitness. Alternatively, $\mathcal{B}$ may specify the new sequence number to be determined from the old sequence number according to a more complicated relation, which may in addition be kept secret by $\mathcal{B}$.

3. Instead of letting $\mathcal{B}$ keep track of the value of $\texttt{seq}$ for each tamper-resistant device, $\mathcal{C}_i$ can inform $\mathcal{B}$ at withdrawal time of the current value of $\texttt{seq}$.

4. Instead of letting $\mathcal{U}_i$ pay to increase the counter, the bank can let him pay for the certified keys that $\mathcal{U}_i$ will withdraw. More specifically, $\mathcal{U}_i$ can be required to pre-pay the maximum amount that may be signed with respect to one certified public key. The withdrawal protocol then becomes superfluous, while $\mathcal{T}_i$ at payment time must accumulate the unspent parts (*i.e.*, maximum amount minus transferred amount) by increasing its counter. A protocol similar to that shown above, but with the roles of $\mathcal{T}_i$ and $\mathcal{B}$ interchanged, can be used by $\mathcal{U}_i$ to deposit the accumulated unspent cash to his account. The choice not to let $\mathcal{U}_i$ pay for downloading certified public keys, but for increasing his counter, has been made merely for explicitness.

The certificate issuing protocol. Payment of an amount requires $\mathcal{U}_i$ to provide the service provider with a signature on the amount (and additional data). As explained, this signature is made with respect to a public key that must have been certified by $\mathcal{B}$, and that may (and can, as long as $\mathcal{T}_i$ is not compromised) be used only once. In accordance with the fourth remark in the preceding paragraph, no cost is charged for downloading certified public keys. Each certified key can be used to sign any amount below a predetermined maximum; payment amounts that are in excess of this maximum amount must be paid by using more than one certified key. To prepare for the withdrawal of a certificate, $\mathcal{T}_i$ and $\mathcal{C}_i$ perform the following off-line pre-processing (see Figure 5):

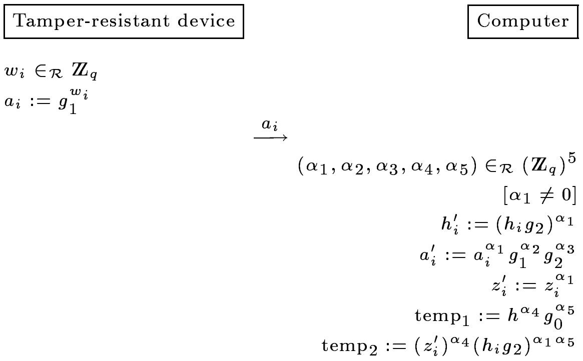

Tamper-resistant device
Computer
$w_i \in_{\mathcal{R}} \mathbb{Z}_q$
$a_i := g_1^{w_i}$
$\xrightarrow{a_i}$
$(\alpha_1, \alpha_2, \alpha_3, \alpha_4, \alpha_5) \in_{\mathcal{R}} (\mathbb{Z}_q)^5$
$[\alpha_1 \neq 0]$
$h_i' := (h_i g_2)^{\alpha_1}$
$a_i' := a_i^{\alpha_1} g_1^{\alpha_2} g_2^{\alpha_3}$
$z_i' := z_i^{\alpha_1}$
$\text{temp}_1 := h^{\alpha_4} g_0^{\alpha_5}$
$\text{temp}_2 := (z_i')^{\alpha_4} (h_i g_2)^{\alpha_1 \alpha_5}$

Figure 5: Pre-processing for certificate issuing protocol.

Step 1. $\mathcal{T}_i$ generates independently at random a number $w_i \in \mathbb{Z}_q$, and sends $a_i := g_1^{w_i}$ to $\mathcal{C}_i$. $\mathcal{T}_i$ stores w_i for later use in the payment protocol.

Step 2. $\mathcal{C}_i$ generates independently at random a vector $(\alpha_1, \alpha_2, \alpha_3, \alpha_4, \alpha_5) \in (\mathbb{Z}_q)^5$, such that $\alpha_1 \neq 0 \bmod q$. It then computes $h_i' := (h_i g_2)^{\alpha_1}$, $a_i' := a_i^{\alpha_1} g_1^{\alpha_2} g_2^{\alpha_3}$, $z_i' := z_i^{\alpha_1}$, $\texttt{temp}_1 := h^{\alpha_4} g_0^{\alpha_5}$, and $\texttt{temp}_2 := (z_i')^{\alpha_4} (h_i g_2)^{\alpha_1 \alpha_5}$.

$\mathcal{C}_i$ stores (h_i', a_i') and $(\alpha_1, \alpha_2, \alpha_3)$ for later use in the payment protocol, and temporarily stores $\texttt{temp}_1$, $\texttt{temp}_2$, α_4, and α_5.

As will be shown in Section 7, the computational burden of performing the exponentiation in Step 1 can be completely moved from $\mathcal{T}_i$ to $\mathcal{B}$.

The actual withdrawal of the certificate is done by means of the following on-line certificate issuing protocol between $\mathcal{C}_i$ and $\mathcal{B}$ (see Figure 6):

Step 1. $\mathcal{B}$ generates at random a number $w \in \mathbb{Z}_q$, and sends $a := g_0^w$ and $b := (h_i g_2)^w$ to $\mathcal{C}_i$.

Step 2. $\mathcal{C}_i$ computes the challenge number $c' := \mathcal{H}(h_i', a_i', z_i', a\,\texttt{temp}_1, b^{\alpha_1}\,\texttt{temp}_2)$. It stores c', and sends $c := c' + \alpha_4 \bmod q$ to $\mathcal{B}$.

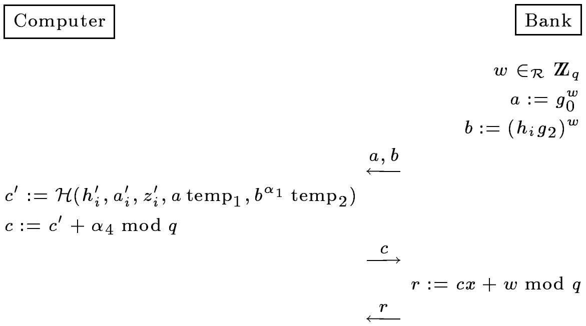

Computer
Bank
$w \in_{\mathcal{R}} \mathbb{Z}_q$
$a := g_0^w$
$b := (h_i g_2)^w$
$\xleftarrow{a,\, b}$
$c' := \mathcal{H}(h_i', a_i', z_i', a\, \mathrm{temp}_1, b^{\alpha_1}\, \mathrm{temp}_2)$
$c := c' + \alpha_4 \bmod q$
$\xrightarrow{c}$
$r := cx + w \bmod q$
$\xleftarrow{r}$

Figure 6: On-line part of certificate issuing protocol.

Step 3. $\mathcal{B}$ sends the response number $r := cx + w \bmod q$ to $\mathcal{C}_i$.

Note that no participation whatsoever of $\mathcal{T}_i$ is needed in the on-line part of the certificate issuing protocol. $\mathcal{B}$ can perform the issuing protocol in parallel for different Internet users, and so there is no processing bottleneck.

$\mathcal{C}_i$ can now go off-line again, and perform the following post-computation:

> $\mathcal{C}_i$ verifies r by verifying that $g_0^r h^{-c} = a$ and $(h_i g_2)^r z_i^{-c} = b$. If both verifications hold, it computes $r' := r + \alpha_5 \bmod q$, and stores r'. The numbers $\texttt{temp}_1$, $\texttt{temp}_2$, α_4, and α_5 can be erased; they are no longer needed.

The payment protocol. To pay to $\mathcal{S}_j$ an amount, `amount`, $\mathcal{T}_i$ and $\mathcal{C}_i$ perform the following pre-processing (see Figure 7):

Step 1. $\mathcal{C}_i$ determines the specification, denoted by `spec`, of the payment. This number is a concatenation, in a standardized format, of the `amount` that is to be transferred, the `time` and `date` of transaction, and an identification number that is uniquely associated with $\mathcal{S}_j$. Additional data fields may be included in `spec`. $\mathcal{C}_i$ then sends (h_i', a_i') and `spec` to $\mathcal{T}_i$.

Step 2. $\mathcal{T}_i$ verifies that w_i is still in memory, and that `balance` exceeds `amount` ($\mathcal{T}_i$ can read this value from `spec`). If this is the case, it computes $d = \mathcal{H}(h_i', a_i', \texttt{spec})$ and $r_1 := dx_{i1} + w_i \bmod q$. It then decreases `balance` by `amount`, erases w_i from memory, and sends r_1 to $\mathcal{C}_i$.

Step 3. $\mathcal{C}_i$ also computes $d = \mathcal{H}(h_i', a_i', \texttt{spec})$, and verifies that $g_1^{r_1} h_{i1}^{-d} = a_i$. If this is the case, $\mathcal{C}_i$ computes $r_1' := \alpha_1(r_1 + dx_{i2}) + \alpha_2 \bmod q$ and $r_2 := d\alpha_1 + \alpha_3 \bmod q$.

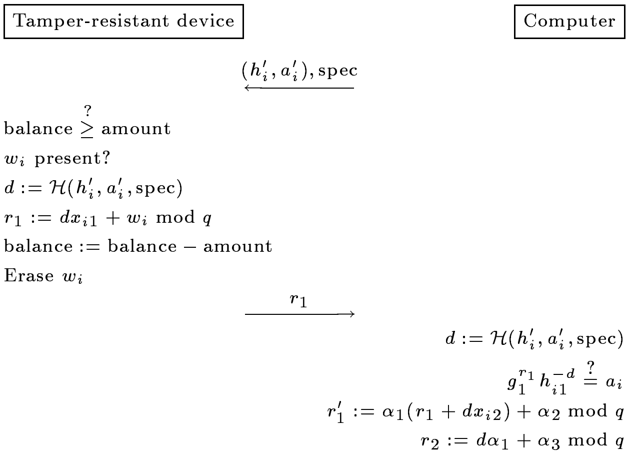

Tamper-resistant device
Computer
$\xleftarrow{(h_i', a_i'),\, \mathrm{spec}}$
$\mathrm{balance} \overset{?}{\geq} \mathrm{amount}$
w_i present?
$d := \mathcal{H}(h_i', a_i', \mathrm{spec})$
$r_1 := dx_{i1} + w_i \bmod q$
$\mathrm{balance} := \mathrm{balance} - \mathrm{amount}$
Erase w_i
$\xrightarrow{r_1}$
$d := \mathcal{H}(h_i', a_i', \mathrm{spec})$
$g_1^{r_1} h_{i1}^{-d} \overset{?}{=} a_i$
$r_1' := \alpha_1(r_1 + dx_{i2}) + \alpha_2 \bmod q$
$r_2 := d\alpha_1 + \alpha_3 \bmod q$

Figure 7: Pre-processing for payment protocol.

Note that it has been assumed that $\mathcal{U}_i$ (or $\mathcal{C}_i$) can determine `spec` without assistance of $\mathcal{S}_j$. When, say, travelling World-Wide-Web links by an interface such as Mosaic, this assumption is very plausible; date and time can be looked up (they may even be entered manually by $\mathcal{U}_i$ by using the keyboard of $\mathcal{C}_i$, although typically $\mathcal{C}_i$ can retrieve this information locally), and the identification number of $\mathcal{S}_j$ can be taken to be the ftp address of the service provider. Alternatively, identification addresses may be retrieved from a local server, and frequently visited addresses can be stored on $\mathcal{C}_i$. For the same reason, this assumption is justified in a situation where the user wishes to tag his payment along with an e-mail message to another party (that in effect plays the role of a service provider). In that case, the e-mail address can serve as the identification number.

The actual payment is done by means of the following on-line payment protocol between $\mathcal{C}_i$ and $\mathcal{S}_j$ (see Figure 8):

> $\mathcal{C}_i$ sends $(h_i', a_i'), (z_i', c', r'), (r_1', r_2)$ to $\mathcal{S}_j$.

$\mathcal{S}_j$ computes d in the same way as did $\mathcal{C}_i$ and $\mathcal{T}_i$, and accepts the transferred information if and only if $h_i' \neq 1$, $c' = \mathcal{H}(h_i', a_i', z_i', g_0^{r'} h^{-c'}, (h_i')^{r'} (z_i')^{-c'})$ and $g_1^{r_1'} g_2^{r_2} (h_i')^{-d} = a_i'$.

It is conceivable that $\mathcal{S}_j$ expects another `time` field value to have been used by $\mathcal{U}_i$, since it can be expected that the granularity of the `time` field in `spec` is such that small disturbances in clock synchronization between $\mathcal{U}_i$ and $\mathcal{S}_j$ result in different values. As will be

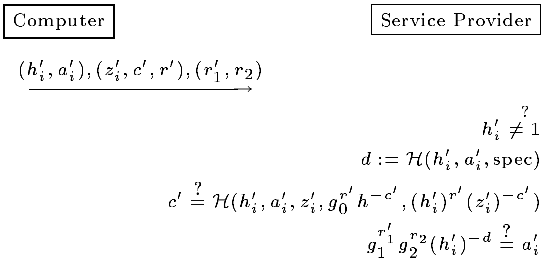

Computer
Service Provider
$(h_i', a_i'), (z_i', c', r'), (r_1', r_2)$
$h_i' \overset{?}{\neq} 1$
$d := \mathcal{H}(h_i', a_i', \mathrm{spec})$
$c' \overset{?}{=} \mathcal{H}(h_i', a_i', z_i', g_0^{r'} h^{-c'}, (h_i')^{r'} (z_i')^{-c'})$
$g_1^{r_1'} g_2^{r_2} (h_i')^{-d} \overset{?}{=} a_i'$

Figure 8: On-line part of payment protocol.

appreciated, this is not a problem: in the on-line part of the payment protocol, $\mathcal{U}_i$ can send the chosen value for the `time` field along. Given that in the deposit protocol $\mathcal{B}$ will accept the payment transcript if (h_i', a_i') and `spec` are not identical to that of a payment transcript that $\mathcal{S}_j$ deposited before, $\mathcal{S}_j$ need merely check that $\mathcal{U}_i$ does not re-use the same certificate later on for the same suggested value for `time` and `date`. To this end, $\mathcal{S}_j$ can for example accept any suggested values for `time` and `date` that are within a certain interval (representing a time span). It then should check at each payment that the transferred payment data has not been received before (within this time span). A time span of, say, 15 minutes should be more than sufficient for most practical applications.

In any case, such measures are not needed if $\mathcal{S}_j$ specifies the `time` and `date` values that must be used by $\mathcal{U}_i$. Although in that case the payment protocol requires interaction, this is no disadvantage in a situation in which $\mathcal{U}_i$ is logged in anyways at the service provider's. If $\mathcal{C}_i$ does not perform the verification of the contribution, r_1, of $\mathcal{T}_i$ on-line, the computational efforts for $\mathcal{T}_i$ and $\mathcal{C}_i$ are so small that the entire interaction need not take more than a fraction of a second.

In Section 7, optimizations are discussed that reduce the amount of data that must be transferred from $\mathcal{C}_i$ to $\mathcal{S}_j$.

The deposit protocol. At a suitable time, preferably when network traffic is low, $\mathcal{S}_j$ sends the payment transcript, consisting of $(h_i', a_i'), (z_i', c', r'), (r_1', r_2)$ and `spec`, to $\mathcal{B}$.

$\mathcal{B}$ verifies that `spec` has been formed correctly by $\mathcal{S}_j$. If this is the case, it searches its so-called deposit database to find out if it has stored (h_i', a_i') before. There are two possible situations:

1. (h_i', a_i') is not yet in the deposit database. $\mathcal{B}$ then computes $d = \mathcal{H}(h_i', a_i', \texttt{spec})$, and verifies the payment transcript by verifying that $h_i' \neq 1$, $c' = \mathcal{H}(h_i', a_i', z_i', g_0^{r'} h^{-c'}, (h_i')^{r'} (z_i')^{-c'})$ and $g_1^{r_1'} g_2^{r_2} (h_i')^{-d} = a_i'$. If these verifications hold, $\mathcal{B}$ stores (h_i', a_i'), `spec`, and (r_1', r_2) in its deposit database, and credits the account of $\mathcal{S}_j$ by `amount`.

2. (h_i', a_i') is already in the deposit database. In that case a fraud has occurred. If `spec` of the already stored information is identical to that of the new payment transcript, then $\mathcal{S}_j$ is trying to deposit the same transcript twice.

   Otherwise, $\mathcal{B}$ verifies the transcript as described in situation 1. If the verification holds (the payment transcript is valid), then the certified public key (h_i', a_i') must have been double-spent with overwhelming probability. Since $\mathcal{B}$ now has at its disposal a pair (r_1', r_2) from the new transcript and a pair, say (r_1'', r_2'), from the already deposited information, it can compute $(r_1' - r_1'')/(r_2 - r_2') \bmod q$. $\mathcal{B}$ then searches its account database for joint public key $g_1^{(r_1' - r_1'')/(r_2 - r_2')}$. Since the identity of the corresponding account holder is known to $\mathcal{B}$, appropriate legal actions can be taken. The number $(r_1' - r_1'')/(r_2 - r_2') \bmod q$ serves as the proof of $\mathcal{B}$ that the traced user has compromised his tamper-resistant device and has double-spent the certified public key (h_i', a_i').

5 Correctness.

An analysis of correctness (security and privacy) is omitted here, since the results that can be proven are highly similar to those provided by me in [3]. Informally, the main results of this analysis are as follows. It can rigorously be proven that the unlinkability and untraceability of payments is guaranteed unconditionally for users that follow the protocols. As for security, forging certified public keys that can be used to sign amounts is as hard as breaking the Schnorr signature scheme [18]. If $\mathcal{T}_i$ cannot be compromised physically, then its response r_1 in the payment protocol cannot be used by attackers to spend the corresponding certified key a second time (without assistance of $\mathcal{T}_i$), if the Schnorr identification scheme is secure. A successful false claim by the bank of a user having double-spent a certificate requires the bank to break the Discrete Log problem.

The only aspect for which no rigorous proof is known (under some standard intractability assumption, preferably security of the Schnorr signature scheme) is to prove that the supposedly unblindable part of the secret key indeed is unblindable; a proof is known only when assuming a non-standard, though plausible, intractability assumption [4].

6 Other mathematical embodiments.

In [5], I presented another mathematical embodiment of the system in Section 3. This other embodiment offers the same features, except for the ability to mathematically disprove false accusations of double-spending. The computational requirements for all parties are reduced in this embodiment by a factor of about a half. The on-line computational effort for $\mathcal{U}_i$ is only one modular multiplication of two 64-byte numbers, *i.e.*, about .5 % of that in the embodiment of Section 4. Another important advantage is that it can rigorously be proven [4] that an account holder indeed cannot blind the unblindable part of the secret key that pertains to his account, assuming only that the Schnorr signature scheme [18] is secure.

Whether the embodiment of [5] is preferable to the one that has been described in Section 4 is unclear, because the proof of unblindability only works for serial executions of the certificate issuing protocol (it exploits the fact that not even the bank needs to know its own secret key when performing the protocol with respect to *one* account holder). Indeed, it can be shown [4] that two *parallel* executions of the issuing protocol for two *different* account holders can be misused, by two different users that first break both their tamper-resistant devices, and then make smart choices for the two respective challenge values in the parallel executions of the certificate issuing protocol. The resulting certified public key can be used to sign multiple messages without this resulting in traceability to either one of the two users.

In [4], I also describe a minor modification that suffices to make parallel executions of the withdrawal protocol immune to attacks. This modification consists of letting the bank choose (part of) the blinding-invariant number associated with an account holder randomly anew in *each* execution of the protocol, making it known to the account holder only after it has received his challenge. At the start of the protocol, it instead makes g_1 raised to this number known to the account holder. This causes some additional storage burden for the account holders. A suitable combination [4] of this measure with a technique to reduce queueing in the serial version (which amounts to letting the bank demand that a challenge be retured in a pre-determined amount of time after having sent out its random number) ensures that the additional storage per token need be no more than a few bytes.

7 Optimizations

A number of practical optimizations for the embodiment of Section 4 will now be discussed.

Firstly, it suffices for $\mathcal{C}_i$ to perform the verifications $g_0^r h^{-c} = a$ and $(h_i g_2)^r z_i^{-c} = b$, when post-processing the certificate issuing protocol, at randomly chosen occasions: if these verifications do not hold, then the service provider will not accept in the payment protocol. A scenario in which service providers cooperate with $\mathcal{B}$ to gather tracing information, by also accepting incorrectly formed certificates, will definitely not be worthwhile to $\mathcal{B}$: it can be proven that $\mathcal{B}$ cannot send an r in certificate issuing protocol such that $\mathcal{T}_i$ in the corresponding payment protocol can leak more than one bit of subliminal information, while it enables users to earn money by transferring bogus information to the service providers.

Secondly, the number d in the payment and deposit protocols can be computed, by all parties involved, as $d := \mathcal{H}(h_i', \mathcal{H}(a_i'), \mathtt{spec})$. The verification relation $g_1^{r_1'} g_2^{r_2} (h_i')^{-d} = a_i'$ must correspondingly be replaced by $\mathcal{H}(g_1^{r_1'} g_2^{r_2} (h_i')^{-d}) = \mathcal{H}(a_i')$. Since the size of the outputs of $\mathcal{H}(\cdot)$ will typically be about one third or one fourth of the size of a_i', the size of the transmitted data is reduced.

Likewise, the number z_i' can be hashed before being hashed by $\mathcal{C}_i$ in Step 2 of the on-line part of the certificate issuing protocol. Correspondingly, the certificate is verified by $\mathcal{S}_j$ in the payment protocol, and $\mathcal{B}$ in the deposit protocol, according to $c' = \mathcal{H}(h_i', \mathcal{H}(a_i'), \mathcal{H}(z_i'), g_0^{r'} h^{-c'}, (h_i')^{r'} (z_i')^{-c'})$.

A further reduction can be attained by merging the verification relations $g_1^{r_1'} g_2^{r_2} (h_i')^{-d} = a_i'$ and $c' = \mathcal{H}(h_i', a_i', \mathcal{H}(z_i'), g_0^{r'} h^{-c'}, (h_i')^{r'} (z_i')^{-c'})$ into

$$c' = \mathcal{H}(h_i', g_1^{r_1'} g_2^{r_2} (h_i')^{-d}, \mathcal{H}(z_i'), g_0^{r'} h^{-c'}, (h_i')^{r'} (z_i')^{-c'}).$$

In that case, a_i' need not be transferred in the payment and deposit protocols, and need not be stored by $\mathcal{C}_i$ when the certified public key is downloaded. In the deposit protocol, $\mathcal{B}$ then uses comparison to h_i', instead of to (h_i', a_i'). Of course, d can no longer be computed as $\mathcal{H}(h_i', a_i', \mathtt{spec})$, and so another suitable form must be chosen. Caution must be taken here, to prevent the security from being compromised. A suitable choice for d is $\mathcal{H}(h_i', \mathcal{H}(z_i'), c', r', \mathtt{spec})$. To illustrate what can go wrong for another choice of d, consider using

$d = \mathcal{H}(h'_i, \texttt{spec})$. This would allow a cooperating user and service provider (they can be the same party) to make a profit, by working backwards: generate (r'_1, r_2) and h'_i at random, compute $a'_i = g_1^{r'_1} g_2^{r_2} (h'_i)^{-d}$, and then download a certificate on public key (h'_i, a'_i). The resulting payment transcript $h'_i, (\mathcal{H}(z_i)', c', r'), (r'_1, r_2)$ will be accepted $\mathcal{B}$. Had we taken the approach to let users pre-pay the maximum amount for each certified public key, then this attack would not have brought any gain. In the chosen approach, however, where only the increase of the counter must be paid for, the attackers gain the amount that is signed by the forged signature.

Thirdly, it suffices for $\mathcal{B}$ to store at deposit time the information $\texttt{hash}(h'_i), d, (r'_1, r_2)$, and perform an occasional random check against the information in the deposit-database. The collision-freeness of $\mathcal{H}(\cdot)$ ensures that it is infeasible for service providers to find two different payment specifications that are mapped by $\mathcal{H}(h'_i, a'_i, \cdot)$ to the same outcome (otherwise, they could deposit the two corresponding payment transcripts, and claim that the fact that the corresponding numbers d are identical is a mere coincidence). The function $\texttt{hash}(\cdot)$ serves merely to reduce the storage space that must be reserved for h'_i by $\mathcal{B}$. It need not be one-way; for example, it may simply map h'_i to its ten most significant bytes.

Fourthly, all parties can simulate their random numbers by using a pseudo-random number. For $\mathcal{B}$, this measure hardly has worthwhile advantages in terms of storage and computational requirements, whereas it may reduce the security. Hence, it will be assumed that $\mathcal{B}$ uses advanced methods for generating its random numbers, that depend at least in part on physical sources of randomness.

For $\mathcal{C}_i$ and $\mathcal{T}_i$, though, it definitely is worthwhile to use deterministic pseudo-random generators. Apart from the fact that physical means will be hard to implement, in particularly for the tamper-resistant device, it can significantly reduce the computational and storage requirements for $\mathcal{C}_i$ and $\mathcal{T}_i$. The preferred pseudo-random generators iteratively generate numbers by iteratively performing some suitable recurrence. This allows one to keep track of the produced numbers by keeping track of the iteration sequence number: to be able to regenerate any one of 256 of such pseudo-random numbers, it suffices to store just a single byte.

The random numbers of $\mathcal{C}_i$ only serve to guarantee the privacy of $\mathcal{U}_i$, and so they may be simulated by using, say, a linear congruential generator, or the exclusive-or of several thereof. The fact that each user can use its own home-brewed pseudo-random generator ensures that even unsophisticated methods will suffice to guarantee privacy when many users participate in the payment system. Properties due the linearity in the pseudo-random generator may be used by $\mathcal{C}_i$ to compute forms such as $a_i^{\alpha_1} g_1^{\alpha_2} g_2^{\alpha_3}$ much more efficiently than by straightforward simultaneous repeated squaring.

The advantage of $\mathcal{C}_i$ using a pseudo-random generator is that it can choose to regenerate numbers at payment time, instead of storing them at certificate issuing time. The storage of h'_i and z'_i can be circumvented in this way. Although this improvement will hardly be worthwhile for Personal Computers, given the hard disk capacities that are available at affordable prices, it may certainly be worthwhile when $\mathcal{C}_i$ is, say, a palmtop computer or mobile telephone that can be interfaced to the Internet by infra-red communication to a local server.

Since the randomness of $\mathcal{T}_i$ serves to prevent $\mathcal{U}_i$ from finding out its secret key, and hence prevents $\mathcal{U}_i$ from using certified public keys multiple times without $\mathcal{T}_i$'s assistance, $\mathcal{T}_i$ must use a more sophisticated pseudo-random number generator. Nevertheless, I believe that a block-cipher based one-way function, or cascaded linear shift registers, are certainly sufficient to maintain the security in practice. Note that $\mathcal{B}$ may in addition keep the description of the pseudo-random generator of the tamper-resistant devices secret, and can even vary the implemented method randomly over tamper-resistant devices.

The important benefit of the use of pseudo-random numbers by $\mathcal{T}_i$ is that Step 1 of the pre-processing for the certificate issuing protocol can be performed entirely by $\mathcal{B}$. This moves the computational burden of computing $g_1^{w_i}$, which is by far the most costly action performed by $\mathcal{T}_i$ in the protocols, from $\mathcal{T}_i$ to $\mathcal{B}$.

The specific changes that must be made to the protocols are as follows. In the j-th execution of the certificate issuing protocol with $\mathcal{U}_i$, $\mathcal{B}$ (instead of $\mathcal{T}_i$) computes a_i by raising g_1 to the power $w_i^{(j)}$, where $w_i^{(j)}$ is the number produced by the pseudo-random number generator of $\mathcal{T}_i$ in the j-th iteration. $\mathcal{B}$ then transfers a_i to $\mathcal{C}_i$. Correspondingly, in its j-th execution of the payment protocol, $\mathcal{T}_i$ uses the number $w_i^{(j)}$ produced in the j-th iteration of its pseudo-random number generator, to compute $r_{i1} = dx_i + w_i^{(j)} \bmod q$. $\mathcal{B}$ can easily synchronize j with $\mathcal{T}_i$ by maintaining a counter. Alternatively, $\mathcal{C}_i$ can inform $\mathcal{B}$ of the current value of this counter at the start of the certificate issuing protocol. Note that $\mathcal{T}_i$ no longer needs to store g_1 and code to perform computations in G_q.

8 Performance evaluation

The performance evaluation that is presented in this section assumes that all the optimization techniques described in the previous section are applied to the embodiment of Section 4.

For explicitness, it will be assumed that G_q is the subgroup of $\mathbb{Z}_p^*$ for some prime p such that $q|(p-1)$ (although it is good to realize that an elliptic curve implementation can significantly reduce the storage required for the public keys of $\mathcal{B}$). Using the parameter lengths proposed by Schnorr [18] for his signature scheme, it will be assumed that $k = 72$, $|q| = 140$, and $|p| = 512$, where $|\cdot|$ denotes binary length. For greater security, one may want to increase these values.

As far as computational speedups is concerned, the performance estimate is fairly unsophisticated: only the computation of products such as $g_0^a g_1^b$ using straightforward simultaneous repeated squaring is considered.

The computational effort of generating the random numbers will be neglected: the time complexity of performing an iteration of the pseudo-random generators that is used by $\mathcal{C}_i$ and $\mathcal{T}_i$ is negligible in comparison to the complexity of a modular multiplication in G_q. Likewise, the computational effort of computing images of $\mathcal{H}(\cdot)$ and $f(\cdot)$ is neglected.

The following list shows the estimated performance figures for each of the four types of computing devices:

* To perform an execution of the certificate issuing protocol, $\mathcal{B}$ must perform about 420 modular multiplications of two 64 byte numbers. Virtually the entire effort is off-line and can hence be pre-processed.

To verify a payment transcript, $\mathcal{B}$ must perform about 750 modular multiplications of two 64 byte numbers. To facilitate efficiency at deposit time (service providers may deposit thousands of transcripts at once), $\mathcal{B}$ should preferably use cryptographic co-processors.

Storage of $(\mathtt{hash}(h_i'), d, r_1', r_2)$ for a deposited payment transcript requires only 54 bytes.

The search for collisions of $\mathtt{hash}(h_i')$ in the deposit database can be done very fast for even for a huge database, if only the database is sorted.

* The computational effort of $\mathcal{S}_j$ in the payment protocol is about 750 modular multiplications of two 64 byte numbers. On a Personal Computer with a 80486 micro-processor running at 66 MHz, a C-implementation will perform this task in about one second. To store the data of one payment for deposit later on, 160 bytes must be stored (assuming for explicitness that `spec` is 17 bytes). In other words, a popular World-Wide-Web server that processes 131,000 downloading requests per day, and deposits only at the end of each day, would need a 20 Megabyte hard disk for storing payment transcripts.

* $\mathcal{T}_i$ must permanently store q, x_{i1}, the description of a pseudo-random number generator, and an initial seed for the pseudo-random number generator. Furthermore, it must maintain a counter. Note that the storage requirements for $\mathcal{T}_i$ are completely independent of the number of certified public keys that is withdrawn by $\mathcal{C}_i$.

The total computational effort of $\mathcal{T}_i$ in the certificate issuing protocol is zero, since Step 1 of the pre-processing protocol is performed by $\mathcal{B}$.

The total computational effort of $\mathcal{T}_i$ in the payment protocol is equal to the cost of performing one modular multiplication of a 9-byte number and a 17.5-byte number. If $\mathcal{T}_i$ is a PCMCIA card with an ordinary 8-bit micro-processor, this task can be performed within a few milli-seconds. Due to the low storage and transmission requirements, 256 bytes of RAM and, say, 2 Kbyte EEPROM, on the PCMCIA card chip will be sufficient for a highly practical implementation.

* $\mathcal{C}_i$ must permanently store x_{i2}, (g_0, g_1, g_2, h), (p, q), h_{i1}, h_i, and the description of $\mathcal{H}(\cdot)$. $\mathcal{C}_i$ also needs a software program to perform its role in the protocols.

In order to withdraw a certificate, the most demanding action required of $\mathcal{C}_i$, $\mathcal{C}_i$ must perform about 1400 multiplications modulo a 64 byte number. The greater part of this effort is done off-line; the on-line computations are about 210 multiplications modulo a 64 byte number. On a Personal Computer with a 80486 microprocessor running at 66 MHz, a C-implementation will perform the on-line computation in about one third of a second.

Assuming that $\mathcal{C}_i$ regenerates $\alpha_1, \alpha_2, \alpha_3$, z_i' and h_i' when pre-processing for a payment, the dynamic storage for a certified key, consisting of (c', r'), is merely 26.5 bytes, plus about one byte to keep track of the state of the pseudo-random number generator (this allows regeneration of the number h_i' for about 85 certificates without needing to update the seed).

In total, $\mathcal{U}_i$ needs to send only 143 bytes in order to pay.

For slower computers, the on-line computation for the certificate issuing protocol may take many seconds. In that case, the following approach can be taken. To download, say, 1000 certified public keys, $\mathcal{B}$ and $\mathcal{C}_i$ perform the certificate issuing protocol in parallel. After having received the 1000 initial pairs (a, b) from $\mathcal{B}$, $\mathcal{C}_i$ disconnects. It then computes 1000 appropriate challenges c, which may take, say, an hour. Once finished, it logs in again, sends the 1000 challenges, and receives the matching 1000 responses. Of course, this requires $\mathcal{B}$ to store for each (a, b) the secret number w for this period of time (or regenerate them from a pseudo-random number generator). Note that this is an important advantage of interactive protocols that may be performed in parallel over those that may not.

9 Extensions

In this section, it is shown how to incorporate currency conversion and accommodate multiple banks.

Currency conversion. To transfer an amount that is specified in a different currency than the currency maintained by the tamper-resistant device, the following technique can be used. Before subtracting the amount from the counter, the tamper-resistant device multiplies it by an appropriate conversion rate. There are several ways for the bank to ensure use of valid conversion rates; the interested reader is referred to [5].

Multiple banks. Different Internet banks can coexist. Each bank can issue its own electronic cash tokens, by using its own signature scheme. One way to embody this is to let each bank use its own secret key x. When only a limited number of banks is participating, the public keys of each bank can be stored locally at the service providers. If there are a great many banks, the public-key certificate technique can be used to enable service providers to verify the validity of electronic cash tokens of banks that are not known to them. Hereto, a master-organization must issue a public-key certificate on the public key of each Internet bank, which can then be transferred along with the other payment data to the service provider.

To settle between multiple banks, a clearing center must be used. The design of an appropriate clearing mechanism can be copied from existing paper-conducted transaction mechanisms, and hence falls outside of the scope of this paper.

10 Trade-offs.

There are two methods for the participants to increase efficiency, by trading off with privacy of payments.

Degeneration of randomness. The overwhelming part of the computational effort that is performed by the equipment of the user consists of the blinding operations in the certificate issuing protocol. If $\mathcal{U}_i$ wants perfect untraceability and unlinkability of a payment, he must perform about 1400 modular multiplications of two 64-byte numbers. The use of random numbers, $(\alpha_1, \alpha_2, \alpha_3, \alpha_4, \alpha_5)$, from a smaller domain than $\mathbb{Z}_q$ results in a reduction of this computational effort, but introduces correlation between the obtained certificate and the identity of $\mathcal{U}_i$. As will be clear, users can trade off between efficiency and privacy of payments. Since each user can in principle select his own method of random number generation, in practice such trade-offs need not necessarily result in a significant loss of privacy.

A particularly interesting trade-off is the following. $\mathcal{C}_i$ can set the tuple $(\alpha_1, \alpha_2, \alpha_3, \alpha_4, \alpha_5)$ to some initial random value, and use this tuple to download a great many certificates. As a result, all these certificates are linkable, but not traceable, while the computational requirements drop to 420 modular multiplications per certificate. In effect, this emulates the process of the bank issuing a tamper-resistant device, whose payments are all linkable, in an anonymous way to the account holder. The smaller the domain is from which α_1 is generated, the more bits of tracing information are revealed at payment time; this can be compared to the situation in which the anonymous distribution of a bunch of devices over users is constrained to, say, those users that live in the immediate environment of a local distribution office. At any time, $\mathcal{U}_i$ can switch the choice of tuple, thereby emulating the process of swapping his tamper-resistant device with another user.

Multi-spendable certificates. The bank can allow, or demand, certified keys to be used a great many times. By *requiring* k messages to be signed with respect to one certified public key, the deposit database of the bank is significantly reduced in size. To discourage users from still using each certified public key only once, the bank can let users pay for certificates (instead of for updating their counters). Specifically, to download one k-spendable certified public key, the account balance of the user is decreased by k times the maximum spendable amount. The counter in the tamper-resistant device then serves to accumulate the unspent parts, so that the user can deposit the accumulated value at the bank later on. All the payments that are made with respect to the same certified public key are linkable, but not traceable.

To implement this measure in the embodiment of Section 4, a public key that may be used up to k times can be taken to be of the form $(h_i', a_{i1}', \ldots, a_{ik}')$ (as in Section 7, each of these a_{ij}' values can be hashed). Each a_{ij}' is generated by $\mathcal{U}_i$ in cooperation with $\mathcal{T}_i$, in the same way as a_i' is generated in the described preprocessing for the certificate issuing protocol. To sign an amount with respect to this public key for the j-th time, the signature (r_1', r_2) is computed with respect to (h_i', a_{ij}'). The necessary modifications for the deposit protocol will be obvious. It is easy to show that k signatures with respect to the same certified public key are perfectly untraceable, while $k+1$ signatures with respect to the same public key reveal $x_{i1} + x_{i2} \bmod q$ (pigeonhole principle).

Discussion. The first method only decreases the computation complexity of the users, while the second also decreases the computation and storage complexity for the bank. On the other hand, the first measure allows users to determine the degree of privacy of their payments by themselves, which is not the case with the second measure. It seems therefore best in practice to apply a suitable mix of both measures. For example, the bank can issue different types of certified keys, and specify for each type a different upper bound k. Note that k should not be taken to large, because the size of the transferred payment data in the payment protocol grows accordingly.

11 Conclusion

The Internet payment system that has been proposed in this paper may seem to be less attractive than many other proposals, because it requires tamper-resistant hardware for the users. In the longer run, though, when the use of smart cards and the like for electronic payments has become commonplace, the advantages will significantly outweigh this objection. What will remain are the advantages: click-and-pay ability to make instantaneous off-line payments, the ability to cost-effectively serve tens of millions of participants, the ability to guarantee one's own privacy, multi-party security, and portability of tamper-resistant devices to other payment platforms.

References

[1] Birch, D., "Downloading Software, Uploading Money–Business on the Infobahn," April 1994, presented in June 1994 at the Technology Appraisal's conference "Internet and the Enterprise" in London.

[2] Bos, J., Chaum, D., "SmartCash: a practical electronic payment system," Centrum voor Wiskunde en Informatica, Report CS-R9035, August 1990.

[3] Brands, S., "Untraceable off-Line cash in wallets with observers," Advances in Cryptology—Proceedings of Crypto '93, Lecture Notes in Computer Science, no. 773, Springer-Verlag, pp. 302–318. An extended pre-print has appeared as: "An efficient off-line electronic cash system based on the representation problem," Centrum voor Wiskunde en Informatica, Report CS-R9323, March 1993. Anonymous ftp: ftp.cwi.nl: /pub/CWIreports/AA/CS-R9323.ps.Z.

[4] Brands, S., manuscript (1993), submitted in parts for publication.

[5] Brands, S., "Off-line cash transfer by smart cards," Centrum voor Wiskunde en Informatica, Report CS-R9455, September 1994. Anonymous ftp: ftp.cwi.nl: /pub/CWIreports/AA/CS-R9455.ps.Z. Also in: Proceedings of the First Smart Card Research and Advanced Application Conference (October 1994), France, pp. 101–117.

[6] Chaum, D., "Blind signatures for untraceable payments", Advances in Cryptology—Proceedings of Crypto '82, Lecture Notes in Computer Science, Springer-Verlag, pp. 199–203.

[7] Chaum, D., "Achieving electronic privacy", Scientific American, Aug. 1992, pp. 96–101.

[8] "World's first electronic cash payment over computer networks," DigiCash B.V. press release, May 27, 1994.

[9] Chaum, D., Fiat, A., Naor, M., "Untraceable electronic cash," Advances in Cryptology—Proceedings of Crypto '88, Lecture Notes in Computer Science, no. 403, Springer-Verlag, pp. 319–327.

[10] Chaum, D., Pedersen, T., "Wallet databases with observers," Advances in Cryptology—Proceedings of Crypto '92, Lecture Notes in Computer Science, no. 740, Springer-Verlag, pp. 89–105.

[11] Dukach, S., "SNNP: A simple network payment protocol," Proceedings of Computer Security Applications Conference, November 1992, pp. 173–179.

[12] Even, S., Goldreich, O., Yacobi, Y., "Electronic wallet," Advances in Cryptology—Proceedings of Crypto '83, Lecture Notes in Computer Science, Springer-Verlag, pp. 383–386.

[13] Stein, L., Stefferud, E., Borenstein, N., Rose, M., "The Green Commerce Model," October 1994. [about the system of First Virtual.]

[14] Low, S., Maxemchuk, N., Paul, S., "Anonymous credit cards," submitted to Globecom '94.

[15] Medvinsky, G., Neumann, B., "NetCash: a design for practical electronic currency on the Internet," Proceedings of the First ACM Conference on Computers and Communications Security, November 1993.

[16] Okamoto, T., "Provably Secure and Practical Identification Schemes and Corresponding Signature Schemes," Advances in Cryptology—Crypto '92, Lecture Notes in Computer Science, no. 740, Springer-Verlag (1993), pp. 31–53.

[17] Okamoto, T., Ohta, K., "Disposable zero-knowledge authentications and their applications to untraceable electronic cash," Advances in Cryptology—Crypto '89, Lecture Notes in Computer Science, Springer-Verlag, pp. 481–496.

[18] Schnorr, C., "Efficient Signature Generation by Smart Cards," Journal of Cryptology, Vol. 4, No. 3, (1991), pp. 161–174.

[19] "UK banks introduce Mondex, the cashless cash card," Newsbytes News Network, January 6, 1993.

[20] "System planned for shopping in the Internet," Wall Street Journal, September 13, 1994, pp. B1. [about CyberCash, Inc.]