

Electronic Cash

Stefan Brands

Brands Technologies, The Netherlands

brands@xs4all.nl

1 Introduction

Soon you may find yourself e-mailing bytes representing your money to service providers across the ocean, using a smart card or a handheld computer to pay at your local grocery, and making backup copies to protect your money against hard-disk crashes. Now that technological advances in chip manufacturing have brought vast computing powers within everyone's reach, traditional cash is about to be replaced for an invention based on modern cryptography: electronic cash.

This chapter provides an overview of the state-of-the-art techniques for designing electronic cash systems, without going into great technical detail. We begin by quickly reviewing today's payment forms and analyzing their inherent shortcomings, to motivate the subsequent description of properties that are generally deemed desirable for an electronic equivalent of traditional money.

1.1 Traditional Cash Payments

Traditional cash is a bearer instrument that can be used spontaneously and instantaneously, to make payments from person to person without the involvement of a bank. It is the preferred method for low and medium value purchases, which make up the bulk of our everyday transactions. Cash payments also offer privacy, because they are not normally traceable by a third party. Together these factors account for the wide acceptability of traditional cash.

Traditional cash also has several shortcomings. Creating cash that is hard to forge, transporting cash from one place to another, protecting cash transport and storage, and replacing worn out coins and bank notes all make traditional cash very costly to handle for banks. Bank notes are easily destroyed and can be forged using sophisticated color copier machines, coins are too heavy to carry around in bulk, and both are easily lost or stolen. Cash comes in fixed denominations and so typically many coins or bank notes need to change hands to pay a single amount. Because coins and bank notes reveal neither the payer's nor the payee's identity, cash is the preferred method of payment for money laundering, bribery and extortion. Cash, moreover, can be passed on many times without the need for settlement by a bank, further conflicting with the desire of governmental organizations to trace criminal money. Another shortcoming of traditional cash, one that has become particularly urgent in recent years, is the inherent requirement for physical proximity of payer and payee; coins and bank notes cannot be used for payments over the phone or the Internet.

1.2 Payments by Instruction

Some of the problems of traditional cash have successfully been addressed over the past few decades with the introduction of checks, debit cards and credit cards. Instead of value itself, the payer transfers to the payee an **instruction**, directing the payer's bank to transfer a specified amount. After reception of the transferred instruction from the payee's bank, the payer's bank moves the value from source to destination account, both of which are specified by the payment instruction. Because the actual value resides at all times within the banks, the risks of theft and loss are largely overcome. Checks moreover can be mailed by post, and credit cards can be used over the phone or even the Internet.

Payments by instruction also bring forth several problems of their own, mainly originating from the problem of ensuring their authenticity. Handwritten signatures are easily forged and magnetic stripes copied, PINs can be learned through fake point-of-sale terminals, and specified amounts can sometimes be modified by the payee. This can be addressed in part by upgrading to cryptographic authentication methods in combination with chip cards. Replacing magnetic stripe cards by chip cards with protected memory renders unauthorized card duplication almost infeasible, and fake terminal attacks can be overcome by using cards with a microprocessor that enable a PIN or a biometric to be entered into the card itself. The microprocessor can furthermore be used to compute message authentication codes or, in case it has sufficient processing power, digital signatures; these are much harder to forge than handwritten signatures, and moreover can be verified rapidly and with one hundred percent accuracy by processors in point-of-sale modules. The use of secret keys for cryptographic authentication also enables message encryption for protection against wire-tappers, and hence can protect against wire-tapping of credit card numbers sent across the Internet.

A problem that cannot be addressed by authentication techniques is the need for **on-line** verification of debit and credit card payments; for the former it is inherent, and in case of the latter it is needed (at least as a rule) to protect against overdrawing of accounts. On-line processing makes handling costs significantly higher than for traditional cash and is expensive for payees. A system which requires on-line payment verification moreover can suffer severely from a network breakdown or overload; this can lead to serious delays, and merchants may even have to refuse all payments for a while. Add to this the fact that payees typically need to satisfy stringent registration criteria (such as having a store front), and need to acquire expensive tamper-resistant modules, and it is easy to understand why the acceptability of credit and debit cards is much lower than of cash payments. Only guaranteed checks can be verified **off-line** and without special equipment, but these are not readily issued by banks, do not allow instantaneous payment, and are time-consuming to obtain, verify, process and redeem.

Another inherent problem is that all payments are effortlessly **traceable** by the bank which performs the actual transfer of value, from source to destination account. This enables

intrusive profiling of spending behavior and, by inference drawing on mathematical techniques such as data mining, all manner of other personal information characteristics. This can lead to junk mail, unjustified or wrong assumptions about personal behavior and other characteristics, and outright discrimination. In the wrong hands, detailed information about spending and inferred habits is a valuable tool for victimizing people, be it by criminals to select their targets, or by political aggressors to track down or lock out opponents. Data protection laws can offer only limited protection against abuse of personal information, and cannot protect against the harmful consequences that may result from payment profiling errors.

1.3 Electronic Cash Properties

By combining the benefits of traditional cash with those of payments by instruction, while at the same time circumventing the shortcomings of each, a list of desirable properties for an electronic equivalent of traditional cash can be composed.

As with traditional cash, electronic cash should have high acceptability, should be cost-effective for low-value purchases, and should be suitable for payment from person to person. This means that payees should not need costly tamper-resistant modules provided by financial institutions, nor should they need to meet stringent registration criteria.

It should also be possible to verify payments off-line. Off-line payment capability is a prerequisite for platform independence and hence for cross-platform portability, a key feature of an open system. While for a platform such as the Internet, on-line verification need not become a bottleneck until a substantial number of users are active, for most other platforms on-line verification is not cost-effective and can lead to unacceptable delays. Systems requiring on-line payment verification go against the philosophy of cash payment, and cannot be considered true electronic cash.

As with payments by instruction, electronic cash should offer storage and transportation convenience, while protecting users against loss, theft and accidental destruction. Physical proximity of payer and payee should not be needed, so that electronic cash payments can be made over the phone or the Internet. Furthermore, all manner of electronic cash handling should be dealt with electronically, to ensure cost-effectiveness, instantaneousness and accuracy; this includes not only creating, issuing, spending, verifying, counting, transporting and revoking of electronic cash, but also fraud detecting and tracing.

The extent to which an electronic cash system offers privacy of payment may well be the decisive factor in its ultimate acceptance in an open consumer market. Three forms of privacy of payment can be distinguished. The first is known as **confidentiality**, and refers to the ability of account holders to hide transaction details from wire-tappers, and to hide information such as purchase descriptions from the bank. Payment confidentiality can be achieved fairly straightforwardly, by encrypting all sensitive data.

The second form of privacy is the ability to hide who is transacting with who, how many

times, and so on. In case of Internet payments and the like, traffic analysis may easily reveal such information, regardless of whether messages are encrypted, unless special measures are taken. We do not consider techniques to hinder traffic analysis in this chapter, because they are platform dependent and not necessarily based on cryptography; for example, in the case of Internet payments one can use IP spoofing or anonymous remailers, and in other circumstances traffic analysis may be inherently difficult.

The third form of privacy is payment **untraceability**. Without the use of proper cryptographic design techniques, every electronic cash payment would cause a unique transaction identifier to end up in the computer files of the bank, and electronic cash would be no better in this respect than payments by instruction. Apart from off-line payment capability, it is the possibility to have payment untraceability that sets electronic cash apart from instruction-based payment forms.

In addition to the objective of untraceability of payments, any acceptable design for an electronic cash system should strike a balance between the perfect two-sided untraceability of traditional cash and the governmental desire to discourage criminal uses, such as money laundering, bribery and extortion.

Achieving many or all of the properties listed above, without giving in on system security, may seem a daunting if not impossible task. However, as discussed in this chapter, with the right choice of cryptographic techniques small miracles can be accomplished. We will successively examine payment authentication techniques, ways to represent electronic cash, techniques for electronic cash transfer, techniques to guarantee security even when tamper-resistance is compromised, security measures for account holders, and techniques to achieve payment privacy. This is followed by a detailed description of an example electronic cash system, designed on the basis of the considerations and insights of the preceding sections. At the end of this chapter a summary of the material presented is provided.

2 Preliminaries

2.1 Modeling Electronic Cash

The flow of electronic cash resembles that of traditional cash, but the minting need not necessarily take place by a central party. When electronic cash is backed by some commodity of generally trusted value, such as gold or traditional cash, different banks can be allowed to mint their own electronic cash. Electronic cash then merely serves as a sophisticated front for exchanging “real” money, and needs to be purchased from a minting bank, like any other good or service.

Each participant in an electronic cash system is represented by at least one computing device. When an account holder of some bank wants to withdraw some electronic cash minted by his bank, his computing device engages in an execution of a **withdrawal protocol** with

a computing device of the bank. At the end of the protocol execution, the computing device of the account holder holds an amount of electronic cash, represented in either one of two forms discussed later. The bank has charged the account holder, by taking the equivalent amount of “traditional” money out of his bank account and moving it into a **float account**; the electronic cash is prepaid.

To spend electronic cash at a payee that accepts electronic cash issued by his own bank, the account holder interfaces his computing device to one of the payee, and the two computing devices perform an execution of a **payment protocol**. As a result, the representation of the electronic cash amount held by the account holder’s device is adjusted, to reflect the new amount. Since the payment is off-line, the payee’s computing device should correspondingly represent the received amount in some form. As we will see later on, the payee’s representation of electronic cash need not necessarily be the same as that for the payer, and in fact it preferably is not.

Ultimately a party that holds electronic cash issued by the issuing bank needs to sell it back to that bank, to prevent losing money. At the very least, redemption is needed before electronic cash expires, but depending on system design it may also be needed because electronic cash received in a payment cannot be used for subsequent payments, or only up to a predetermined number of times. To this end the party interfaces his computing device to one of his own bank, and the two devices perform an execution of a **deposit protocol**. As a result, the account of the party depositing the electronic cash is credited by his bank with the equivalent amount of money. In case the crediting bank is the same as the issuing bank, it simply takes the money out of its float account. If, however, the crediting bank is the same as the issuing bank, it needs to settle with the issuing bank. Because the design of a suitable clearing infrastructure, no matter how important, is only remotely related to the cryptographic design of electronic cash systems, we do not discuss this further here.

For design purposes, an electronic cash system can conveniently be modeled as consisting of a single bank, one or more payers and one or more payees, each holding one or more computing devices. The core of a cryptographic design of an electronic cash system then consists of the protocols for withdrawal, payment and deposit. Depending on the functionality of the cash system, other cryptographic protocols may be needed.

The computing equipment of the bank can be viewed as a single device, so that one can speak of “the” computing device of the bank. Furthermore, a distinction can be made between **paying devices** and **receiving devices**, which is especially convenient when studying the security of value-transferring protocols. Depending on system design, a device can be either a paying device or a receiving device, or both. For example, paying devices serve as receiving devices in the withdrawal protocol.

Regardless of the manner in which electronic cash held by a paying device is represented, it is clear that at least part of the device must be **tamper-resistant**, in order to prevent double-spending of electronic cash. Namely, if an attacker can determine the internal state

of a paying device, then state freezing or copying enables the same electronic cash to be spent over and over again. This fraud can be detected, if at all, only once the forged cash is deposited to the bank.

Since tamper-resistant paying devices are typically smart cards, their microprocessors are preferably of low complexity and small size. Chip size increases with storage space, and the ability to rapidly perform public-key cryptographic multiplications requires a special cryptographic co-processor. Increased chip size and complexity contribute significantly to manufacturing cost and negatively affect chip reliability and durability. In Subsubsection 3.5.4 and Section 4 we will see how paying devices can be used in combination with other computing devices, such as a handheld device when paying in the local mall or a desktop computer when making Internet payments. Such an “interposed” computer of the account holder need not be tamper-resistant, can take over the greater part of the computational and storage burden of the paying device, and can offer many security and convenience advantages to its holder.

2.2 Authentication Techniques

Of utmost importance for the security of any electronic cash system is that forgery of electronic cash be infeasible, regardless of the form in which it is represented. This implies that receiving devices must be able to distinguish paying devices from attackers who try to pass for paying devices. To prove their authenticity, paying devices necessarily need to be equipped by the bank with a secret key. Correspondingly, receiving devices must be able to recognize whether they are communicating with a device holding a secret key installed by the bank. This requires a secure authentication mechanism.

An easy way for a receiving device to verify the authenticity of a paying device is to require the paying device to reveal its secret key. This, however, enables a wire-tapper to learn the secret key of the paying device, and to subsequently pass for it. More generally, a secure authentication protocol should resist a **replay attack**. This is an attack in which a wire-tapped **transcript** of an execution of an authentication protocol is re-used by the attacker in order to pass for a paying device. **Static authentication**, in which the evidence provided to prove authenticity is always the same, does not offer adequate security.

Prevention of replay requires **dynamic authentication**. The key observation is that instead of revealing a secret key, it suffices to prove knowledge of the key, without revealing it. To achieve this seemingly contradictory task, the paying device must perform a computation that can be performed in a reasonable time span, if at all, only when it knows the secret key. The outcome of the computation must be verifiable by the receiving device, so that it can conclude that the outcome must have been generated by a paying device. To prevent a replay attack, the receiving device must ensure that a different computation is performed each time.

Protocols for dynamic authentication are commonly called **challenge-response protocols**, because the task that the receiving device wants to see performed can be regarded as a

challenge to the paying device, and the result of the computation as the **response**. When designing a challenge-response protocol, care must be taken that wire-tappers cannot (feasibly) compute the secret key by analyzing transcripts of protocol executions, or by selecting challenges by themselves in some clever way.

Techniques for secure challenge-response authentication vary widely. An important distinction is between **conventional** dynamic authentication and dynamic authentication based on **public-key cryptography**. The following overview is focussed especially on the applicability to electronic cash design.

2.2.1 Conventional Dynamic Authentication

In case the receiving device already knows the secret key of the paying device, it can verify a response by computing it by itself, and verifying for equality. The result of a computation that is based on a secret key and a message, and that can be verified only by using the same secret key, is called a MAC (Message Authentication Code). Secure MACs are those that resist cryptanalysis even against active attackers, who can request and cryptanalyze arbitrarily many MACs for self-chosen messages. Secure MACs trivially enable the construction of secure challenge-response protocols: to this end the receiving device requests the paying device to compute a MAC for a challenge (and possibly such information as a device ID number). This is also known as symmetric authentication.

The challenge can be chosen either at random from a large domain, possibly by using a pseudo-random number generator, by concatenating a unique ID number of the receiving device to a sequence number (maintained by the receiving device and incremented upon each execution of the authentication protocol), or by using a sufficiently accurate estimate of the receiver's local time and date. In the case of random numbers an attacker who has learned MACs for many messages has negligible probability that a receiving device subsequently requests a MAC for a message seen earlier; in the case of sequence numbers and time/date estimates, challenges are guaranteed to differ with each protocol execution, but they can be anticipated and may conceivably be of greater value for a cryptanalytic attack.

Because receiving devices must know, or at least must be able to generate, the secret keys of the paying devices with which they need to be able to conduct transactions, receiving devices must be tamper-resistant as well. Therefore they must be issued by or on behalf of the bank, which takes care of installing the secret keys into all paying and receiving devices. When multiple banks each issue their own electronic cash, payees need different receiving devices for each bank. (In practice, this can take the form of different tamper-resistant chips plugged into a single standardized receiving module.)

The particular manner in which the secret keys are installed is of great importance to the system security. Three basic approaches are known.

The System-Wide Secret Key. With this method, paying and receiving devices all hold the same random secret key, generated and installed by the bank. A major drawback is that an attacker needs merely be able to extract the secret key of any device, by compromising its tamper-resistance, in order to pass for any paying device.

Diversified Keys. This method is similar to the preceding, with the difference that each paying device holds a unique secret key. Each receiving device must be able to recognize the secret keys of all paying devices. Since storing all the secret keys is inefficient, and adding new keys each time new paying devices are introduced is cumbersome, each receiving device stores a master key, generated at random and installed by the bank. The secret key of a paying device is computed as a function of the master key and a unique ID number of the paying device, and is called a **diversified key**. To ensure that the master key cannot be computed from a device secret key, the function must at least be one-way. In practice often a DES-based one-way function is used.

In case a secret key of a paying device has been extracted by an attacker, and used fraudulently, once the fraud is detected the compromised device can be traced and subsequently blacklisted. A weak point is that the master key is present in any receiving device, and extraction enables an attacker to pass for a paying device with an arbitrary ID number.

Certification of Diversified Keys. A security improvement over the previous method can be achieved by **certification** of paying device ID numbers. To this end the bank stores into each paying device not only an ID number, but also its own digital signature on that ID number (and possibly also information such as an expiration date). In the payment protocol, the paying device transfers its ID number and the digital signature to the receiving device, which establishes the validity of the ID number by applying the public key of the bank to the digital signature. Message authentication then proceeds as with the preceding method.

The advantage over the diversified key method is that an attacker who knows the master key cannot pass for an arbitrary paying device, since he cannot forge signatures of the bank. Hence the attacker can pass only for a paying device for which he has learned the ID number and the digital signature, which conceivably makes it easier for the bank to trace the fraud.

2.2.2 Dynamic Authentication based on Public-Key Cryptography

The vulnerability that is caused by the omnipresence of a master key can be removed by using **asymmetric** instead of symmetric authentication. As before, to prove its authenticity to a receiving device, a paying device computes a response based on its secret key and a challenge; but this time the receiving device verifies the result by applying the corresponding **public** key of the paying device.

As with the method of the system-wide secret key, it is not acceptable that all paying devices use the same secret key, and storing the public keys of paying devices into receiving

devices is problematic for the same reasons as with the diversified key method. Instead, paying devices should provide their public keys to receiving devices during payment. Receiving devices must then be able to verify the validity of public keys, to which end the bank must certify the public keys of all paying devices. MAC certification voids the security benefits of public-key cryptographic authentication, because the bank's authentication key must be known to all receiving devices, so that only digital signatures are acceptable for certification. This also has the advantage that a single receiving device can be used to receive electronic cash issued by different banks.

Two basic public-key cryptographic techniques are known to design challenge-response protocols, **zero-knowledge authentication** and **digital signatures**. Zero-knowledge techniques can be used to design authentication protocols for which active attackers, who are allowed to cryptanalyze many protocol transcripts for self-chosen challenges, provably cannot learn any useful information beyond the public key.

In a secure digital signature scheme, it is infeasible for active attackers to subsequently forge a new (message, signature) pair. Hence a secure method of digital signing can be used effectively as a proof of knowledge of the secret key used for signing, without revealing "useful" information to attackers. A drawback is that the resistance against cryptanalytic attacks is harder to prove than for zero-knowledge authentication. On the other hand, a very important advantage of digital signatures over zero-knowledge authentication, as well as over conventional dynamic authentication, is that a digital signature constitutes an unforgeable transcript of a proof of knowledge of the secret key. It also demonstrates that any other information included in the challenge message, such as an amount, has been agreed to by the signer. A transcript of a protocol execution can therefore be used by a receiving device to demonstrate to the bank that a transaction has taken place; this does not require trust of the verifier in the tamper-resistance of the receiving device, and in fact receiving devices need not be tamper-resistant at all.

Many digital signature schemes, such as the RSA scheme of [Rivest, Shamir and Adleman, 1978] and the Schnorr scheme of [Schnorr, 1991], are based on number-theoretic primitives. One-way functions built from block-ciphers (such as DES) can be processed two to four orders of magnitude more efficiently, and are of significant interest for electronic cash design because they allow implementation on paying devices with low-cost microprocessors. For this reason, we will now review several of these efficient schemes.

Lamport signatures. The first signature method we consider is the Lamport signature scheme [Lamport, 1979]. The public key of a paying device is a set of $2k$ numbers, for an appropriate security parameter k , of the following form:

$$(f(s_{01}), f(s_{11})), (f(s_{02}), f(s_{12})), \dots, (f(s_{0k}), f(s_{1k})), \quad (1)$$

and the secret key consists of the $2k$ preimages s_{ij} . The function $f(\cdot)$ is a one-way function that can be evaluated rapidly; an example is the function that assigns to a 55-bit input x the DES encryption of a fixed message, computed using the 56-bit key that is obtained from concatenating 0 to x . For storage efficiency, the s_{ij} can be generated in a pseudo-random manner from a single secret, for example by applying a one-way function to i , j , and a randomly generated seed that is kept secret by the paying device.

To compute a Lamport signature on a message m of binary length k , the paying device sends to the receiving device a certificate of the bank on the public key. In addition, it releases, for each bit m_i of m , either $(s_{0i}, f(s_{1i}))$ or $(f(s_{0i}), s_{1i})$, depending on whether m_i is zero or one.

With this method, the binary length of a signature is $2k$ times the binary length of preimages of $f(\cdot)$, assuming inputs and outputs have the same length. In a practical implementation this typically exceeds storage requirements for, say, RSA signatures by an order of magnitude. As noted by Lamport, an alternative way to compute signatures is for the paying device to release a distinct subset of preimages for each message: this enables the signing of messages almost twice as long, using the same key set-up. Another method to achieve the same efficiency improvement is due to [Merkle, 1988], who proposed to expand all k -bit messages into codewords, by appending a count of the number of zero-bits; a message can then be signed securely by using a public key consisting of $k + \lceil \log_2 k \rceil$ instead of $2k$ ordered images of secret preimages.

Matrix-based signatures. A further significant improvement in signature storage can be obtained by an improvement of [Merkle, 1985]. In this improved method a secret key for a paying device consists of a matrix of r rows by $2c$ columns, which for explanatory purposes is best thought of as two matrices of r rows by c columns each, referred to as a “message” matrix and a “control” matrix, respectively. A key is generated by filling both matrices in a special way. To fill matrix j , for $j \in \{1, 2\}$, c random numbers, s_{1j}, \dots, s_{cj} are generated, and these are used to fill out the r -th matrix row. The rest of the matrix is filled out row by row by applying the hash function $f(\cdot)$, as follows:

$$\begin{pmatrix} f^{r-1}(s_{1j}) & f^{r-1}(s_{2j}) & \dots & f^{r-1}(s_{cj}) \\ \vdots & \vdots & \vdots & \vdots \\ f(s_{1j}) & f(s_{2j}) & \dots & f(s_{cj}) \\ s_{1j} & s_{2j} & \dots & s_{cj} \end{pmatrix} \quad (2)$$

The top rows of the two matrices are fed into a one-way hash function, referred to as a **compression** function, and the result is the public key for the paying device.

Using this key pair set-up, the paying device can sign any one of up to r^c messages. To sign a message of $\lceil \log_2 r^c \rceil$ bits, the message is first expanded into r -ary representation. Denoting the r -ary representation of m by $m_1 m_2 \dots m_c$, the paying device releases $f^{r-1-m_i}(s_{i1})$ from

the message matrix and $f^{m_i}(s_{i2})$ from the control matrix, for each $i \in \{1, \dots, c\}$. As an example, consider $r = 4$ and $c = 3$. To sign the 6-bit message 010011, with 4-ary expansion 103, the paying device releases the numbers marked by the boxes:

$$\begin{array}{ccc}
 \text{Message matrix} & & \text{Control matrix} \\
 \left(\begin{array}{ccc}
 \overbrace{f^3(s_{11})} & \boxed{f^3(s_{21})} & f^3(s_{31}) \\
 \boxed{f^2(s_{11})} & f^2(s_{21}) & f^2(s_{31}) \\
 f(s_{11}) & f(s_{21}) & f(s_{31}) \\
 s_{11} & s_{21} & \boxed{s_{31}}
 \end{array} \right) & & \left(\begin{array}{ccc}
 \overbrace{f^3(s_{12})} & \overbrace{f^3(s_{22})} & \boxed{f^3(s_{32})} \\
 f^2(s_{12}) & f^2(s_{22}) & f^2(s_{32}) \\
 \boxed{f(s_{12})} & f(s_{22}) & f(s_{32}) \\
 s_{12} & \boxed{s_{22}} & s_{32}
 \end{array} \right) \tag{3}
 \end{array}$$

To verify the signature, the receiving device computes the top row of each of the two matrices, by applying the appropriate number of one-way function applications to the matrix entries provided. By applying the compression function to the two computed top rows, it can then check whether the result matches the public key of the paying device.

It is important to note the need for the control matrix. If only the message matrix were used, then wire-tapping of a device signature for a message would enable signature forgery for any message that has digits in its r -ary expansion that all are no greater than those in the same position in the r -ary expansion of the first message. With the control matrix, in effect an additional “control” message has to be signed, for which each digit is equal to the additive inverse of the message. Releasing values higher up in the message matrix thus requires releasing values moving downward in the control matrix, which requires the ability to invert $f(\cdot)$.

Storage of the secret key of the signer can be compressed using the same technique as for Lamport signatures. In particular, in practice the matrix rows may be generated from a short random seed in pseudo-random manner. A further improvement can be achieved by reducing the number of columns needed for the control matrix; instead of using one “control digit” per message digit, one can get away with logarithmically many control digits (and hence logarithmically many columns for the control matrix) by summing the digits of the message and signing the resulting number in a suitable expansion. Note that the dimensions of the message matrix and the control matrix need not be related; we assumed this merely for explanatory purposes.

The security of the Lamport and the matrix-based signature methods has recently been studied in detail in [Even, Goldreich and Micali, 1996]. A further efficiency improvement can be found in [Vaudenay, 1993], where it is observed that a matrix can be used for a larger message space, by defining a one-to-one relation between messages and subsets of entries in the matrix, such that no subset can be computed from any other and each subset contains exactly one entry in each column. This condition is fulfilled by taking entries according to a Latin square, or more generally by taking all subsets with entries such that the sum of the rows that the entries are in is a constant. A generalization of this technique to directed acyclic graphs is studied in [Bleichenbacher and Maurer, 1994].

Tree authentication. A serious drawback of both the Lamport and the matrix-based signature methods is that it is insecure for the bank to let paying devices sign more than once with respect to the same public key. As an extreme example, consider a wire-tapper who intercepts two signatures made by the same paying device, in either signature scheme, one for the message having all r -ary digits equal to zero and another having all r -ary digits equal to $r - 1$ (with $r = 2$ for the Lamport signature scheme); from then on he would be able to forge signatures of that paying device for any message. Signature schemes in which a public key may be used for signing only a single message are called **one-time** signature schemes.

An extension of one-time signatures, due to [Merkle, 1988], enables paying devices to use a single certified public key for signing many messages, using a one-time signature for each. The method preserves much of the computational efficiency of the one-time signature scheme, and can be implemented in a standard 8-bit smart card microprocessor. To illustrate the technique, we use for concreteness the matrix-based one-time signature method as the basic building block, following Merkle; it will be clear that this technique can also be applied to any other one-time signature method. Viewing the message matrix and the control matrix as a single matrix, the idea is to take multiple matrix instances as leaves in a tree. Each node value in the level just above the leaves is computed by compressing the top rows of the child matrices. All the other node values in the tree are computed by compressing the child node values, and the value of the root node serves as the public key of the paying device.

To compute a digital signature on a message, the paying device uses a matrix in a leaf of the tree that has not been used before. In addition, the paying device must release the compression values of sufficiently many nodes, to enable the receiving device to compute the root value and thus to verify the public key. In case of a binary tree, the paying device to this end releases the compression result for the sibling leaf matrix and, for each of the internal nodes in the path from the leaf used to the root, the value of the sibling node. A binary tree of depth n enables the paying device to securely make 2^n signatures with respect to the same public key.

As with the one-time signature methods, the key storage space for the paying device can be reduced by generating the bottom rows of the leaf matrices all from a single secret. Only the seed and the certificate on the root value then need to be stored by the paying device. With respect to dynamic storage and computational efficiency for the paying device, one-time signatures are best used up from the leftmost to the rightmost leaf in order. As shown in [Merkle, 1988], it is then possible to generate the additional node values for a new one-time signature in an efficient way from the additional node values of the previous one-time signature. According to later investigations, symmetric tree structures are not optimal in this respect; it is better to use a rake-shaped form.

3 Electronic Cash Techniques

We are now prepared for an explanation of the techniques for electronic cash design.

3.1 Representing Electronic Cash

The presentation thus far has been independent of how electronic cash is represented. Two fundamental ways exist for representing electronic cash in computing devices, whether paying or receiving devices. First is to indicate an amount of electronic cash by means of the value of a counter, maintained in a chip register. For example, one hundred electronic dollars spendable up to cent granularity would be represented by a counter value of 10,000. This representation is often referred to as **register-based cash**. Since money can be forged when counters can be bypassed or updated without bank authorization, the security of systems using this representation of electronic cash relies critically on tamper-resistance.

The other way to represent electronic cash is in the form of cryptographic tokens that are digitally signed by the bank. To each token at least a fixed denomination and currency are assigned, and possibly also attributes such as an expiration date and how many times it may be passed on. The tokens are called **electronic coins**, and must be unforgeable and verifiable solely by using the signature public key of the bank. In its simplest form, each coin is a different (message, signature) pair, from now on referred to as the **two-part form** for coins. The denomination and currency, and any other attributes, can be encoded into the message, or can be indicated by the particular key used by the bank to compute its digital signature. The security of electronic coins relies crucially on the secrecy of the bank's secret key for signing.

Each of the two methods for representing electronic cash has its own characteristic implications for security, functionality and efficiency of a cash system. The register-based representation has the advantage over electronic coins that storage space is minimal. In contrast, for each electronic coin, storage space must be allocated. Coin verification moreover inherently requires the ability to verify digital signatures, which is more costly than verifying MACs. Furthermore, the complexity of communication and computation for coins increases with the number of coins needed to form an amount (although for some cryptographic embodiments, techniques are known to trade storage space against computational requirements). Another disadvantage is that when coins of appropriate denomination are not at hand, a payment requires change from the receiving device or a new withdrawal. In contrast, with register-based cash any amount less than the current register value (or a predetermined maximum) can be paid.

Cash held by paying and receiving devices can be either register-based or in the form of electronic coins, and within the same system any of the four possible combinations can make sense. Paying devices may even hold part of their value in the form of electronic coins, and the rest in the form of register-based cash (see Subsection 4.7). The implications of the various

combinations will become clear in the rest of this section, and in particular in Subsection 3.5 we will see that there are valid reasons for preferring coins over register-based cash.

3.2 Transferring Electronic Cash

To transfer electronic cash securely from a paying device to a receiving device, secure authentication by the paying device is required. The techniques for authentication detailed in Subsection 2.2 can all be used in combination with either form of electronic cash representation, but significant differences exist in the manner in which this is accomplished. In any case, the internal cash balance (whether register-based cash or in the form of coins) of a paying device should be decreased before transferring the electronic cash; otherwise the transaction could be interrupted by the holder of the device between the sending of the electronic cash and the internal adjustment (and we do not want to depend on the receiving device to send an acknowledgment).

3.2.1 Transferring Register-Based Cash

When combining an authentication method with a register-based cash representation for paying devices, not only the paying device but also the amount that is transferred must be authenticated. The paying device must decrease its register value to reflect the amount that is transferred, and of course it must have been programmed by the bank to do so only when its current value represents an amount exceeding the payable amount.

In order to authenticate the amount, it can be encoded into the challenge. Because for security the varying part of the challenge must remain intact, the resulting challenge becomes longer. In case a correct response is returned, the receiving device is assured not only that the other device is a paying device, but also that the paying device has decreased its local balance by the specified amount (assuming that its tamper-resistance has not been broken, of course).

The receiving device can store the transferred cash either as register-based cash or in the form of an electronic coin. In the case of zero-knowledge authentication, or authentication by means of a MAC, only the former representation can be used, since the receiving device does not receive information that could only have been created by a paying device or by the bank. In the case of authentication by means of a digital signature computed by the paying device, the digital signature received can serve as an electronic coin, having a denomination specified by the challenge message that has been signed. Alternatively, the use of digital signatures only serves as a more efficient alternative to zero-knowledge authentication (less interaction) and to make an unforgeable transaction log for use by the bank, and the receiving device stores the amount received in the form of register-based cash; a system based on this approach has been proposed in [Even, Goldreich and Yacobi, 1984].

In case the receiver stores the received amount in the form of a coin, either the coin may

be passed on for a subsequent payment, or it can only be deposited. In the latter case one speaks of an **electronic check** payment, because the paying device fills out the amount at payment time and signs it in an unforgeable manner. A check that has not yet been filled out is called a blank check; it has no value when it is issued, and hence need not be prepaid. An important aspect of electronic check payment is that the bank does not need to trust in the tamper-resistance of receiving devices. For security, blank checks should have a maximum spending limit, assigned by the bank at the time of issuing.

Authentication of the paying device can also take place in an indirect manner, by using a **session key**. In the case of conventional authentication, to this end the paying device sends its ID number to the receiving device, which uses the master key to compute the diversified key of the paying device. The receiving device then sends a random challenge to the paying device, and both parties use the secret key of the paying device to compute a MAC for the challenge. However, rather than the paying device sending the MAC to the receiving device, as a response, both parties regard it as a session key. Further messages, such as for the transfer of an amount, are authenticated and/or encrypted using the session key. Since these tasks can be performed only if the session key is known, valid MACs and/or messages decrypting to meaningful messages convince the receiving device indirectly that the other device must be a paying device.

In the case of public-key cryptographic authentication, the two devices can send their certified public keys to one another, and develop a session key by means of Diffie-Hellman key exchange [Diffie and Hellman, 1976] or another method. The session key can be used to compute MACs for subsequent messages and/or to symmetrically encrypt messages. As with session keys derived from MACs, the ability to do any of the above demonstrates indirectly to the receiving device that the other device is a paying device. In contrast to session keys derived using conventional authentication, no master key needs to be held by devices outside the bank.

Alternatively, the following public-key cryptographic technique can be used to form a session key. The bank provides each tamper-resistant paying device with a unique secret key and a corresponding public key. The secret key is the trap-door of a trap-door one-way function $f(\cdot)$, specified by the public key of the device. To transfer cash to a receiving device, the paying device sends to the receiving device its public key and a certificate for it of the bank. The receiving device generates a random challenge x in the domain of $f(\cdot)$, and sends $f(x)$ to the paying device. The paying device uses its trap-door to uncover x , and both devices use x , or a part of it, as the session key. Again, the actual transfer of the amount can be authenticated by means of a MAC or a digital signature. An advantage over the preceding method is that much faster cryptographic embodiments of this technique for forming session keys are known.

In case of conventional dynamic authentication it is sometimes preferable to reverse the roles of devices that hold a secret master key and devices that hold diversified secret keys.

When receiving devices hold diversified keys and paying devices a master key, electronic cash transfer can take place based on the following observation: the ability of a paying device to compute a MAC using the secret key of the receiving device demonstrates to the receiving device that the device it is communicating with knows the master key, and hence that it must be a paying device. This is particularly useful in the withdrawal protocol, because it enables the bank to issue register-based electronic cash to paying devices without having to provide these with its master key.

3.2.2 Transferring Electronic Coins

To issue electronic coins to a paying device of an account holder, the bank computes digital signatures on distinct messages, sends these to the device (possibly encrypted to prevent wire-tapping) and debits the account of its holder by the total value of the coins. To make a payment, the paying device first determines whether coins of appropriate denominations are present. If the amount cannot be made up, either an excess amount must be formed and the receiving device must supply change, or first another withdrawal must be performed.

When coins are in the two-part form, the receiving device must be tamper-resistant and the coins must be encrypted before being transmitted, to protect against coin theft or copying. When using conventional authentication, a session key can be formed by the paying and the receiving device, which is then used by the paying device to encrypt the coins. Before sending out the coins, the paying device erases the coins from memory. The receiving device decrypts, verifies the coins using the public key of the bank, and stores them. Depending on the number of times the coins may be passed on, the receiving device can later on either deposit the coins or use (some of) them to make a payment. As we have seen, the presence of master-keys in devices outside the bank can be avoided by forming session keys using a public-key cryptographic method.

The need for receiving devices to be tamper-resistant can be avoided by defining a coin to be a triple consisting of a secret key, a corresponding public key, and a certificate of the bank on the public key. From now on this is referred to as a coin in the **three-part form**. The secret key of the coin belongs to a paying device, and at least part of it may not be known to anyone else than the device itself (and perhaps the bank). For each triple, the bank charges the account of the device holder for the value of the coin. To spend a coin, the paying device computes a digital signature on a challenge message of the receiving device (zero-knowledge authentication makes the approach pointless), using the secret key of the coin triple, and sends it to the receiving device, together with the public key and the certificate of the coin triple. The receiving device can verify the payment by using the public key of the bank. The received coin cannot be passed on, since the secret key of the coin has not been divulged by the paying device and in a subsequent payment another challenge will have to be signed. Consequently, the receiving device can only deposit the coin. By encoding into the challenge message a unique account identifier, uniquely associated by the bank with the account of

the holder of the receiving device, it is effectively ensured that a wire-tapped coin cannot be deposited to another account. Note that this technique can also be applied to electronic checks.

Payments with electronic coins in the three-part form are very similar to electronic check payments. They differ in that for check payment a register-based cash representation is used and blank checks can be issued free of charge; the amount payable is encoded into the challenge message. In contrast, electronic coins are prepaid at withdrawal time, for their denomination value. Furthermore, each electronic coin is a new triple from the bank, while check payments can be made using a single triple (secret key, public key, certificate), installed by the bank in an initial stage. For greater efficiency, the bank can program paying devices such that they assist in spending coins in the three-part form a predetermined number of times greater than one (a k -spendable coin at withdrawal time then must be prepaid for k times the coin value), but a limit is inevitable in order to determine the value for which the token must be prepaid.

3.3 When Tamper-Resistance is Compromised

Apart from stealing the secret key of the bank or discovering an algorithmic break or breakthrough (such as how to invert functions that are believed to be one-way), there are two ways in which an attacker may be able to forge electronic money. Thus far we have only touched on the possibility that tamper-resistance might be compromised, and investigated value transfer methods under the assumption that secret keys in tamper-resistant devices cannot be physically extracted by attackers. In reality, there is no such thing as tamper-proofness. Experience has shown that organized crime can hire expertise comparable to that in national laboratories, and even hackers nowadays have access to sophisticated tools. Moreover, care must be taken that device secrets cannot be extracted in much simpler ways than by direct read-out; see, for example, recent research results on cryptanalysis in the presence of induced hardware faults, initiated by [Boneh, DeMillo and Lipton, 1997]. When secrets can be extracted from paying or receiving devices, nothing distinguishes counterfeit from cash issued by the bank. What financial damages can this result in?

Tamper-resistance is best viewed as a matter of economics. Roughly speaking, a technology for tamper-resistance offers adequate security for the bank if the required cost for breaking tamper-resistance exceeds the fraudulent profit that can be made as a result. When estimating the expected fraudulent profit that can be made in an electronic cash system, one also needs to take into consideration the economics of large-scale cracking; to crack a single smart card, equipment and expertise running into hundreds of thousands of dollars may be needed, but this is largely a one-time investment. The damage that can be done ultimately depends on the measures incorporated into the system for preventing or discouraging forgery of electronic cash.

3.3.1 Fraud Detection

Of primary importance is the capability of the bank to **detect** whether unauthorized electronic cash is being introduced into the system. Unless special measures are incorporated, nothing prevents the bank from accurately keeping track of the electronic cash held at various moments in time by the paying and receiving devices of each of its account holders. To keep track of the flow of electronic cash between devices, transaction transcripts that reveal at least how much cash has been transferred must regularly be made available to the bank. If not, then the bypassing of registers of compromised paying devices, or the multiple spending of coins, cannot be detected in any other way than through the economic side-effect of hyperinflation. With sufficiently detailed and regular transaction monitoring, the bank can at the very least detect whether more electronic cash is circulating than it has actually issued.

Electronic coins and checks in this respect have an important advantage over other methods: the transaction record is tied in with the received value, into the digitally signed message of the paying device, and hence received coins or checks cannot be deposited without automatically also revealing the corresponding transaction logs. There is also an important distinction between electronic coins and electronic checks. Instead of double-spending a check, an attacker can also make a profit by spending it for the maximum spending limit, while bypassing the register in the paying device. The bank can detect this only by keeping track of the balances of all paying devices. If such a “shadow” balance ever drops below zero, it is clear that forgery has taken place. With electronic coins, the only way for an attacker to make a fraudulent profit is by double-spending withdrawn coins, and so any forgery of electronic cash can alternatively be detected by the bank by maintaining a list of all deposited coins and checking at each deposit for double-spending. This distinction may not be relevant in case checks and coins are fully traceable, because in effect the same intrusive transaction logging takes place. However, it is crucial when privacy measures need to be incorporated, as we will see in Subsection 3.5.

3.3.2 Fraud Tracing

In case a master key is present in devices outside the bank, an attack on the master key enables cash forgery that cannot be traced to a single device, at least not on the basis of transaction logs. With conventional authentication methods, the ability to extract the master key enables an attacker to perform the authentication for any paying device. When cash held by paying devices is in register-based form, the attacker can forge MACs for any legitimate amount, for any paying device.

In order to be able to **trace** devices that have been compromised, rather than merely those that have been passed for, the bank should not deploy system-wide secret keys. This calls for cash transfer based on public-key cryptographic authentication. When coins in the two-part form are transferred in encrypted form, an attacker needs to wire-tap and decrypt

or to physically extract coins from the paying device or the receiving device, in order to double-spend coins. Even though in these cases the bank can determine which devices have been passed for thus far (to this end, transaction dumps should reveal the ID numbers of paying devices), it has no way of tracing the source of the fraud. Namely, when a coin in the two-part form has been double-spent, it is not clear whether the attack has been on the paying device or on any of the receiving devices in the chain of payments with that coin.

An important advantage of coins in the three-part form over coins in the two-part form is that a coin in the three-part form can be double-spent only by physically extracting the coin secret key from the paying device in which the coin is stored; short of a physical attack on the paying device, the secret key of a three-part coin never leaves the paying device. By keeping track of which paying device has stored which coin secret keys, any forgery of money, no matter how little, can always be traced by the bank to the compromised paying device.

3.3.3 Fraud Liability

When counterfeit can be traced to a specific device, it is not necessarily the case that its holder is a criminal. After all, the device might have been compromised by a thief. In order to be able to point the finger to the holder of the device on firm grounds, the bank must require all its account holders to comply with mechanisms for reporting loss, theft or hardware defects. It should also issue personalized paying devices, that may not be swapped. This can easily be achieved by issuing cards with an access control mechanism, such as a PIN or biometric verification; as will be discussed in Subsubsection 3.4.1, this is also desirable for account holders, in view of loss tolerance.

Furthermore, measures to ensure that thieves cannot operate stolen devices can be incorporated. In this respect, however, little added value comes from protecting tamper-resistant devices with PINs, passwords or biometric access mechanisms, because it can be expected that a hardware attacker can bypass these as well. However, a special use of a secret access code can offer added security. To this end the secret key of a paying device is computed as a function of information held by the device and secret access information that is provided by its holder. Each time after having used the secret key to perform a transaction, the device erases it from memory together with the provided access information. This ensures that the secret key is never present for a long time in the memory of the computing device, and is never stored in non-volatile memory. A randomly chosen eight-character password suffices to prevent a successful exhaustive search using currently available technology. In case a successful forgery due to the compromise of tamper-resistance is traced to a specific device, its holder is either a fraud or has been very sloppy with the secret access information.

To be able to non-contestably demonstrate that a forgery has been made using the keys held by a certain paying device, the trace information that becomes available to the bank in case of forgery should be such that not even the bank itself could have computed the trace information, had there been no forgery. For example, this evidence could be the secret key

that the paying device uses to compute digital signatures at payment time (for checks or coins in the three-part form), or the bank's ability to show $k + 1$ signatures with respect to the same secret key of a device that has been programmed to compute only k signatures with the same key. To this end, paying devices must be able to choose their own secret keys, unknown to the bank, which calls for at least some of the paying device's operations to be under the control of its holder. We will defer a discussion of how to achieve this to Subsubsection 3.5.4, since this scenario is also very important to achieve privacy of payments.

3.3.4 Fraud Containment

When forgery of money can be traced to paying devices, the bank can blacklist the devices. To this end, it places the device ID numbers (or ranges thereof) in a list that is regularly distributed to all receiving devices, for example whenever these perform a deposit. Alternatively, the bank can blacklist (one-way hashes of) the secret or public keys used by the compromised devices for making payments. Furthermore, the ability to trace compromised paying devices may suffice to stop the fraudsters from continuing their fraud.

In case tracing of compromised devices is not possible, the bank can stop further fraud only by revoking its own keys. In case of conventional authentication this requires the distribution of a new master key, and in case of public-key cryptographic authentication a new public key for certification must be made available.

By refreshing master keys and certification keys on a regular basis, indicated by expiration dates, containment can be obtained even for frauds that are difficult to detect. For the same reason, it is desirable that tamper-resistant devices can make and/or receive payments only until a built-in expiration date.

3.4 Security for Account Holders

The major concern thus far has been security for the bank against forgery. For account holders, other security aspects are of importance. Firstly, the electronic cash held by a device should not disappear in any other way than by spending it at the approval of its legitimate holder. Secondly, it should not be possible for an attacker to redirect a payment to a party other than that intended by the legitimate holder of the device. Thirdly, whenever the behavior of an honest account holder is disputed, by the bank or another party, the account holder should be able to substantiate his innocence; this is also in the bank's interest, as discussed in Subsubsection 3.3.3.

3.4.1 Preventing Loss

In general, any lost electronic cash that cannot be redeemed by the bank, for instance because proper recovery measures have not been incorporated, results in a profit for the bank. Once it becomes aware of the loss, the bank can take the value out of the float account.

There are various ways for an account holder to lose electronic cash. Firstly, the contents of his device may be garbled because of a device crash or a hardware fault, or his device may get lost or stolen; measures to protect against these events are referred to as loss-tolerance measures. Secondly, a payment transaction can be interrupted with the effect that the paying device has debited the payable amount while the other device has not received it; this must be protected against by fault-tolerance measures. Thirdly, another party may be able to withdraw from his account at the bank. Finally, in case of a receiving device that is not tamper-resistant, an attacker may have the device accept bogus money by modifying its software or by substituting its copy of the public key of the bank.

Loss-Tolerance. Loss-tolerance is of concern for both paying and receiving devices, and requires that a destroyed, lost or stolen amount of electronic cash can be recovered. Recovery from a lost or stolen device requires greater caution by the bank than recovery from a crash. Namely, in the case of a crashed device the bank can scrutinize the device to determine whether there is any chance of its holder having extracted the secret key (or having attempted to do so), while in the case of loss or theft the bank cannot a priori distinguish between a victimized account holder and one faking loss or theft. In view of this, tamper-resistant devices at the very least should have a suitable access control mechanism (PIN, password or biometric), so that another party cannot spend (or deposit to their own account) the cash held by a lost or stolen device without breaking its tamper-resistance. The bank should also require timely reporting of lost or stolen devices, so that device ID numbers, or withdrawn coins or blank checks, can be blacklisted.

The recovery itself can be done either by the victimized account holder himself or in cooperation with the bank. The former is possible only when the account-holder is able to regularly make backup copies of the contents of his device. In systems based on electronic coins in the three part-form or electronic checks, this measure can easily be applied for receiving devices, since these need not be tamper-resistant and received cash can be deposited only to the account specified in the digitally signed challenge message.

Cooperation with the bank is needed whenever a paying device crashes, or a receiving device that holds a secret key of the bank, because parties other than the bank cannot make backup copies of the secrets in these devices. In systems in which the bank can determine how much was present in the device before it crashed, for example by examining transaction logs and perhaps also information of the user about his most recent payments, the bank can issue a new device and adjust its cash representation to reflect the value at the moment of the crash, or credit the account of the account holder for the lost value. A discussion of several loss-tolerance schemes for register-based cash can be found in [Waidner and Pfitzmann, 1990].

In case a device is reported to have been lost or stolen, the bank should delay reimbursement, if necessary until the current version of electronic cash has expired, to make sure that cash can either be reimbursed through recovery or be spent by means of the normal paying

protocol, but not both. Expiration dates can be built in by the bank changing its keys every now and then; this is desirable anyway, for the purpose of containment in the case of theft of the bank's own secret keys.

In Subsection 4.7 we will see that loss-tolerance can be implemented even in case payments are untraceable.

Fault-Tolerance. Fault-tolerance is mainly an implementation issue. To cope with transaction interruption, a device should at all times be prepared to resend the last message it sent out. For security, a device should as a rule resend only exactly the same message that it sent out previously (of course without modifying its internal cash representation a second time). When setting an interruption flag before sending out a message for the first time, upon reset the device can infer from the flag that it needs to do a re-transmittal. Once an acknowledgment is received (this can be implicit in the response message of the other device), or after a time-out, the device can clear its interruption flag. New actions are performed only once the flag has been cleared again. Transaction processing is a specialty in its own right, and the above is obviously an over-simplification to convey the general idea. A thorough account of transaction processing principles and methods can be found in [Gray and Reuter, 1993].

Account Access Control. In order to prevent withdrawal from account by an unauthorized party, the bank should grant account access only to parties that are properly authenticated. To this end, a MAC or a digital signature of the account holder must be required. In the former case, the secret key needed to gain access to an account is known also to the bank, and fraudulent parties (such as bank employees), may be able to gain access to the key. By digitally signing crucial parts of the withdrawal request, the requested amount, and date and time of the request (and perhaps also the most recent account balance), an account holder can always disavow an unauthorized withdrawal. Of course, withdrawal requests must also be protected against replay attacks; for this purpose each message should contain a fresh part (i.e., an account access sequence number, a random challenge, or a time/date estimate), that must be signed along with each request message.

Software Integrity. An attacker who gains access to a receiving device must be prevented from modifying the software in order to have it accept bogus information. In case the receiving device is not tamper-resistant, the attacker may change, say, a function pointer, so that the software calls a verification routine that always accepts. Alternatively, an attacker can attempt to substitute the key used by the receiving device for verifying electronic cash payments (the master key or the public key of the bank). The attacker can even prevent the receiving device from being informed about the error at deposit time, by substituting the error message of the bank by another, since he can also have the receiving device accept arbitrary strings for digital signatures or MACs of the bank. Protection against unauthorized

access and computer viruses can be achieved by using measures such as password protection and secure operating system software.

3.4.2 Preventing Payment Redirection

A more subtle way in which an attacker might attempt to steal electronic cash is to redirect to its own receiving device a payment intended for another party, or to redirect it to the paying device of another party that the attacker intends to pay. This is known as a **man-in-the-middle attack**.

One way to prevent the wrong party from being paid is to explicitly direct an electronic cash payment to the (account of the) intended payee, by making a digital signature at payment time. To this end, three-part coins or blank checks can be used. When an ID number or a certified public key of the intended receiving device is known to the paying device, it can be included in the challenge message that is signed. Alternatively, a random number or a digital pseudonym can be used, as long as it is uniquely associated by the bank with the account of the payee; in this manner the payee can remain anonymous to the payer. As mentioned in Subsubsection 3.2.2, this measure also prevents an attacker from depositing wire-tapped electronic coins or checks to his own account.

For payment platforms in which the paying device and the receiving device need to be in physical proximity, a generally applicable technique known as **distance bounding** can be used to prevent a fraudulent receiving device from passing on a payment to another receiving device. The idea is for the receiving device and the paying device to mutually agree on the (random part of the) challenge message that is to be responded to by the paying device, by each sending a series of random bits to the other, one by one and interleaved, and concatenating the bits to form the challenge. By timing the delay, the receiving device can determine an upper bound on its distance to the other device. What makes this approach practical is that today's electronics can easily handle timings of a few nanoseconds, and light can travel only about 30cm during one nanosecond. Even the timing between two consecutive periods of a 50 MGHz clock allows light to travel only three meters and back. Details and variations of the above technique can be found in [Brands and Chaum, 1994].

3.4.3 Non-Repudiation

Account holders should be able to disavow erroneous or false incrimination of fraudulent behavior. When parties with differing interests authenticate their messages using MACs or zero-knowledge proofs, the transcript of a communication cannot be used by another party to later on demonstrate that the communication took place, since zero-knowledge transcripts can be formed by anyone. Non-repudiation requires the use of digital signatures, since these can only be computed by the party associated with the public key needed for verification (assuming proper protection of the secret key). When crucial parts of the withdrawal, payment

and deposit protocols are digitally signed, all parties can disavow false claims. Of course, as in Subsubsection 3.3.3, this approach works only if each account holder can generate and control his own secret keys, which requires part of (the operations of) his device to be under his control; this is addressed further in Subsubsection 3.5.4.

3.5 Privacy of Payments

Confidentiality of transaction details is easily achieved by means of line encryption and not depositing details such as the purchase description, and the prevention of traffic analysis is a problem largely orthogonal to the design of an electronic cash system. We now turn to what might well be the most complex issue in the design of an electronic cash system: how can we incorporate untraceability of payments without encouraging criminal uses of electronic cash?

3.5.1 Relaxed Monitoring, Anonymous Accounts and Anonymous Devices

Untraceability can to some extent be incorporated into an electronic cash system by the bank relaxing the requirements for transaction log dumping. When tamper-resistant receiving devices only dump transaction logs with aggregated transaction details, and leave out any information that can be correlated to paying devices or payers, payments cannot be traced. The bank still needs transaction logs to reveal payment amounts and, preferably, time and date of each transaction, in order to be able to detect counterfeit. A distinct problem with this measure is that the detection of counterfeit, originating from a successful hardware attack, is possible only when the aggregated amount of forgery exceeds the total amount in the bank's float account, which clearly is unacceptable for an open system. An improvement can be obtained by categorizing paying devices into several groups, and requiring aggregated transaction records to reveal these group IDs, but this causes a trade-off with payment untraceability. Further serious drawbacks of the approach are that fraud cannot be traced by electronic means only, blacklisting of devices is not possible, and payers have to trust receiving devices to not deposit individual transaction records (unless paying devices do not provide any trace information in the first place, which is the key to the approach in Subsubsection 3.5.2). Relaxed monitoring and security for the bank always need to be traded off with each other.

Another approach towards untraceability is anonymous bank accounts. Although the use of anonymous accounts protects the identity of payers, all the transactions conducted by the same account holder are linkable; a single identification of the account holder in any transaction enables the bank to trace all past and future transactions of the account holder. Increasing the degree to which payments are unlinkable to the payer by allowing users to swap their devices with other users conflicts with the desirability of a device access control mechanism (such as PIN verification). Not only can this approach hardly be said to offer privacy of payments, it also conflicts with the tracing capabilities of the bank in case of forgery. Anonymous accounts moreover are not allowed in most countries, because they

interfere with the tax system.

A third straightforward approach is for the bank to issue tamper-resistant paying devices in such a manner that it does not know which account holder receives which device. Again, this approach cannot be said to offer privacy, because all payments made with the same tamper-resistant device are linkable. It is also difficult to run the device distribution process in a manner that randomly distributes the devices over sufficiently many account holders, and ultimately the provider has to be trusted to properly conduct the distribution. Furthermore, by withdrawing cash from a named account the identity of the holder of the device is revealed. Other drawbacks are that the tracing capabilities of the bank are seriously reduced, tracing cannot be conducted electronically on the basis of transaction logs, and it is difficult to assess to which extent the holder of the traced device is responsible for a fraud originating from that device.

3.5.2 Blinding

A much better way to ensure untraceability of payments is by application of special cryptographic techniques in the design of an electronic cash system. These can ensure that the information that the bank learns about its account holders and their devices is uncorrelated to the information that is revealed by devices when making payments.

In a basic cryptographic paradigm due to [Chaum, 1983], a receiver can obtain from a signer a digital signature on a message, in such a manner that the message and the signature remain completely unknown to the signer. More specifically, in each execution of the protocol, the receiver can obtain with uniform probability a single pair (message, signature) from the set of all possible such pairs, and the signer has maximal uncertainty about the particular pair it has issued. The receiver is said to **blind** the execution of the protocol, and the protocol is called a blind signature issuing protocol.

Efficient blind signature issuing protocols are known for a variety of practical digital signature schemes. The most efficient such protocol known to date is for RSA signatures, and is due to [Chaum, 1983]. It enables a receiver to obtain a pair $(m, m^{1/e} \bmod n)$, where n is an RSA modulus, e is an RSA encryption exponent (co-prime to $\varphi(n)$) and m is a message satisfying an appropriate redundancy pattern (alternatively, m is a one-way hash of a message), as follows:

Step 1. The receiver picks at random a message m from the message space, and a random blinding factor r from \mathbb{Z}_n^* . The receiver then sends $m_0 := r^e m \bmod n$ to the signer;

Step 2. The signer sends $s_0 := m_0^{1/e} \bmod n$ to the receiver; and

Step 3. The receiver computes $s := r^{-1} s_0 \bmod n$.

It is easy to see that s is a digital signature of the signer on m , and that the condition for a blind signature issuing protocol is fulfilled.

The basic blinding technique can be used straightforwardly to design an untraceable electronic cash system, as follows. The bank issues electronic coins in the two-part form, (message, signature), by means of a blind signature issuing protocol. To this end, the three steps above are performed in parallel, one execution for each requested coin, and the account holder in Step 1 specifies the desired number of coins and their denominations. The account holder also digitally signs its request message (and a fresh part, for replay prevention) to prove to be the holder of the account. For each denomination, the bank uses a different RSA exponent, and all RSA exponents are co-prime. To make a payment, the paying device selects coins of the appropriate denominations, encrypts them using an authenticated public key of the tamper-resistant receiving device, erases the coins and then transfers the encrypted payment message.

By following the blind signature issuing protocol, using properly generated random numbers, all payments are anonymous and unlinkable. In fact, even with infinite computing power the signer cannot trace payments, if only account holders use “genuinely random” blinding factors. For greater efficiency (and reconstructability), the blinding factors applied by an account holder may all be generated from a single secret key by using a pseudo-random number generator, but then untraceability is only computational. A particular danger of this is that many years from now it may be feasible to retroactively trace payments, by analyzing the archived deposit databases of the bank; the expected progression in sheer computing power and advances in algorithmics make this a realistic scenario. Surveillance and tracking capabilities are the primary tools of political oppressors to resist opposition, and changes in a political climate are not always predictable. For this reason it is preferable for any electronic cash design for national or global use to be independent of the particular manner in which blinding factors are generated.

By regarding the public key of a coin in the three-part form, (secret key, public key, certificate), as the message in a blind signature issuing protocol, it follows that a paying device can withdraw completely blinded coin triples by using the above blinding technique. Alternatively, blinded triples serve as blank checks, in combination with a register-based cash representation in the paying device. Note that the bank does not need to know the secret keys of the triples obtained by the paying device. As we have seen in Subsubsection 3.3.3 this is desirable for the bank, because it is then able to come up with an incontestable proof in case a k -spendable coin or blank check has been used at least $k + 1$ times. Of course, the bank must also prove that the coin or blank check has indeed been withdrawn by the account holder, and not been constructed by itself; this can be accomplished by showing the (digitally signed) request message that the account holder sent in Step 1 of the execution of the issuing protocol in which the coin or blank check was issued.

A general drawback of the basic blinding technique is that the bank can never **trace** forgery: even fraudulent payments are untraceable. For the same reason the bank cannot apply blacklisting to contain further fraud. Moreover, in case of checks substantial amounts

of forged cash may be injected into the system without the bank even being able to detect this. As a consequence, the basic blinding technique is appropriate only to the design of electronic coin systems in which payments are deposited **on-line**; in case of a double-spending attempt, the bank can then simply tell the receiving device to not accept the payment. By having the paying device encrypt its coins for the bank, instead of for the payee's device, neither the paying nor the receiving device needs to be tamper-resistant. On the downside, one of the two major advantages of prepaid electronic cash over instruction-based payment forms, off-line payment verification, has now been sacrificed completely in order to achieve the other, untraceability. We want to achieve **both** properties.

3.5.3 One-Show Blinding

In order to enable the bank to trace double-spent coins, without sacrificing the untraceability of coins that have been spent only once, the paradigm of **one-show blinding** has been introduced in [Chaum, Fiat and Naor, 1988]. This paradigm requires the construction of an issuing protocol and a payment protocol that securely act in concert, in the following manner. The payment protocol must be such that a signature of the paying device on one challenge message does not reveal any information that helps tracing, but any two different signatures, for the same coin, reveal trace information. This trace information must be encoded by the bank into each coin that is issued. To this end, coins are represented in the three-part form, (secret key, public key, certificate), and the issuing protocol must be such that the coin public key and the coin certificate can be fully blinded by the paying device, while the bank must make sure that the coin secret key contains an identifier (note that the tamper-resistance of the paying device may already have been defeated by an attacker before the issuing takes place), at least with substantial probability.

For the payment protocol, a one-time signature scheme can be used, with the property that the computation of two signatures with respect to the same coin public key enables the computation of the coin secret key, or at least of the identifier. To deposit the received coin, the receiving device transfers to the bank the coin public key, the coin certificate, the challenge and the signature of the paying device. The bank verifies the information by checking its own certificate, the signature, and the uniqueness and correct formation of the challenge message. Any double-spending of the same coin (for which the tamper-resistance of the paying device must be compromised) results in the deposit of a second signature with respect to the same coin public key, but on a different challenge message. From any two such different signatures the bank can compute the identifier, and hence trace the compromised paying device. Of course, this paradigm can also be used for electronic checks.

Two fundamentally different approaches are known for designing issuing protocols for the one-show blinding paradigm.

Cut-and-Choose Blinding. The first approach can be inferred from [Chaum, Fiat and Naor, 1988], and is commonly called **cut-and-choose blinding**. To retrieve a coin of the specified three-part form, the paying device and the bank engage in performing in parallel a great many executions of a basic blind signature issuing protocol, with the notable difference that the bank completes its part of the protocol only for one of the protocol runs, which it chooses at random. Moreover it does so only when the paying device can demonstrate for all the other protocol runs that it has properly encoded the required identifier into its messages (whence the name “cut-and-choose”); this is called **opening** of a blinded candidate.

To illustrate this process, consider using the blind RSA signature issuing protocol described in Subsubsection 3.5.2 as the underlying blind signature issuing protocol. The paying device generates independently at random many “candidate” key pairs, (secret key, public key), such that each secret key contains the identifier in a prescribed manner. It blinds each of the public keys, in the same manner as one blinds messages in the basic blind RSA signature issuing protocol, and sends the resulting blinded public keys to the bank. The bank then requests the paying device to “open” all but one of the submitted blinded public keys, by revealing for each blinded public key the secret key and the blinding factor used in its construction. The bank verifies whether the opened candidates have been constructed properly and, if so, discards them and computes its RSA root of the remaining unopened candidate. Upon receiving the signed blinded candidate, the paying device removes the blinding factor, as in Step 3 of the blind RSA signature issuing protocol.

A very serious shortcoming of this straightforward implementation is that the probability of detecting a fraud increases only linearly in the number of protocol runs, and thus a huge amount of data must be exchanged in order to achieve a sufficiently low probability of successful deception. In the scheme actually proposed by Chaum, Fiat and Naor, the security level is **exponentially** related to the number of protocol runs, an important improvement. Their construction, which is very specific and does not seem to allow for generalization, is as follows. The pair (secret key, public key) of a coin triple is a key pair for making a Lamport signature (see Subsubsection 2.2.2), but with a twist. Specifically, the public key is a set of k numbers, for an appropriate security parameter k , of the following form:

$$g(f(s_{01}), f(s_{11})), g(f(s_{02}), f(s_{12})), \dots, g(f(s_{0k}), f(s_{1k})). \quad (4)$$

The functions $f(\cdot)$ and $g(\cdot)$ are collision-intractable, and the $2k$ secrets s_{ij} of the paying device are of the following form:

$$s_{0j} = (I \oplus a_j, b_j) \quad \text{and} \quad s_{1j} = (a_j, c_j), \quad \forall j \in \{1, \dots, k\}, \quad (5)$$

for numbers (a_j, b_j, c_j) chosen at random by the paying device. The number I is an identifier, uniquely associated by the bank with the paying device. The certificate of the bank is an RSA signature on the public key. To retrieve a triple of the specified form, the paying device sends to the bank $2k$ blinded $g(f(s_{0j}), f(s_{1j}))$ terms, and opens only half of these, randomly selected

by the bank. If the verification succeeds, the bank signs **the product** of the remaining k numbers. Upon removing the product of the k corresponding blinding factors, the paying device is left with one coin triple.

Of course, this time the paying device can slip in an erroneously formed $g(\cdot)$ term with probability $1/2$, and more generally n erroneously formed terms with probability close to $1/2^n$. To defeat successful spending of malformed coins, the payment protocol for a coin is designed as follows. The paying device sends the coin public key and the coin certificate to the receiving device, and in addition computes a Lamport signature on a k -bit challenge message, m . Denoting the binary expansion of m by $m_1 \cdots m_k$, the paying device releases for each bit m_i either $(s_{0i}, f(s_{1i}))$ or $(f(s_{0i}), s_{1i})$, depending on whether m_i is zero or one.

It is easy to see that one signature does not help the bank to trace the payment device, assuming that $f(\cdot)$ hides its first argument unconditionally. To guarantee that the probability of untraceable repeated spending is no greater than $1/2^n$, any two challenges must be guaranteed to differ in at least $2n$ bit positions (note that the possibility of payees cooperating with the payer must be taken into account). To this end, challenges can be expanded into code words with minimum distance $2n$. Double-spending the same coin (for which the tamper-resistance of the paying device must be compromised) results in the deposit of a second Lamport signature with respect to the same coin public key, but on a different challenge message. From any two such different signatures the bank can compute the identifier, I . Namely, for any two bit-positions where the two challenge messages differ, the bank knows both s_{0i} and s_{1i} and hence can compute a candidate identifier, by taking the exclusive-or of their first arguments; the majority candidate must then be the identifier sought for.

Note that the linear growth in the challenge length this time brings exponentially growing security, and that a guaranteed minimum distance of n may suffice to achieve the specified security level in case identifiers of account holders are random secrets and are never transmitted in the clear. Namely, in that case an attacker cannot encode valid identifiers of other account holders into his own coins, so that in case of double-spending with overwhelming probability only one of the computed candidate identifiers is valid.

Although ingenious, this improved realization of cut-and-choose blinding is still far from practical. For a practical implementation, k must be taken to be, say, 70, with 40 bits of each challenge containing a varying part, a receiving device ID and possibly an amount (in case of an electronic check), and the other 30 bits being the required hamming distance to get a sufficiently low probability of successful fraud without traceability. With practical choices for the numbers a_j , b_j , c_j and the blinding factors, the resulting signature size exceeds that for an RSA signature by about two orders of magnitude. Not only must all this information be transmitted, it must also be stored by the bank, for a duration at least as long as the validity of its certification key. Furthermore, the size of the data transmitted in the withdrawal of a single untraceable electronic coin or check is several hundred times the size of an RSA signature.

Restrictive Blinding. The second approach to design issuing protocols for the one-show blinding paradigm is called **restrictive blinding**. It avoids the expensive cut-and-choose of the first approach, by inherently restricting the paying device in the manner in which it can blind the secret key for coin or check triples; the bank can then encode the identifier itself into the coin or check, by encoding it into the blinding-invariant part of the secret key.

In [Brands, 1995 (D)] a general technique is described to design efficient restrictive blind issuing protocols based on any so-called Fiat-Shamir type signature scheme (another protocol appears in [Brands, 1994], but its design does not follow a generally applicable technique). An important ingredient to this technique is the use of **secret-key certificates**, instead of public-key certificates, for coin or check triples. Although a secret-key certificate is not a digital signature of the bank on the public key of a triple, it offers the same functionality as a digital signature; see [Brands, 1995 (C)] and [Brands, 1995 (E)] for details. The extra flexibility that comes from this can be exploited in the design of restrictive blind issuing protocols.

Following is an example of this technique, based on the Schnorr digital signature scheme [Schnorr, 1991]. The secret key of the bank is a k -tuple (x_1, \dots, x_k) , where $k - 1$ is the number of “identifiers” that the bank can encode independently into each triple that it issues; $k = 2$ suffices for electronic cash applications, as will be shown below. Its corresponding public key is

$$p, q, \mathcal{H}(\cdot), (g_0, g_1, \dots, g_k), \quad (6)$$

where q and p are primes such that q evenly divides $p - 1$. The secret key (x_1, \dots, x_k) is chosen at random by the bank from $(\mathbb{Z}_q)^k$, g_0 is a random number of order q in \mathbb{Z}_p^* , and $\mathcal{H}(\cdot)$ is a collision-intractable hash function. (Alternatively, all computations can be performed on an elliptic curve of order q over a finite field; this allows working with much smaller numbers than needed in case of \mathbb{Z}_p^* , since the best known algorithms for discrete logarithms in such groups are exponential rather than subexponential.) The rest of the public key is generated deterministically:

$$g_i := g_0^{x_i} \bmod p \quad \forall i \in \{1, \dots, k\}. \quad (7)$$

With the current state of computers and cryptographic knowledge, a 20-byte prime q and a 100-byte prime p should suffice for long-term security.

A certificate of the bank on a public key h of order q in \mathbb{Z}_p^* for a paying device is a pair (r, c) such that

$$c = \mathcal{H}(h, g_0^c h^r \bmod p). \quad (8)$$

Corresponding to the public key h of the paying device is a secret key (y_1, \dots, y_k) in $(\mathbb{Z}_q)^k$, such that

$$h = g_1^{y_1} \cdots g_k^{y_k} \bmod p. \quad (9)$$

The following issuing protocol enables the paying device to obtain a triple (y_1, \dots, y_k) , h , (r, c) from the bank, in such a manner that the public key h and the certificate (r, c) are fully blinded, while the paying device cannot prevent the bank from encoding into the secret key $k - 1$ numbers, (I_2, \dots, I_k) , where $x_1 + \sum_{i=2}^k x_i I_i \neq 0 \pmod q$:

Step 1. The bank generates at random a number $w_0 \in \mathbb{Z}_q$, and sends $a_0 := g_0^{w_0} \pmod p$ to the paying device.

Step 2. The paying device generates at random three numbers $\alpha_1 \in \mathbb{Z}_q^*$, $\alpha_2, \alpha_3 \in \mathbb{Z}_q$, and computes

$$h := (g_1 g_2^{I_2} \cdots g_k^{I_k})^{\alpha_1} \pmod p \quad (10)$$

and

$$c := \mathcal{H}(h, a_0 g_0^{\alpha_2} (g_1 g_2^{I_2} \cdots g_k^{I_k})^{\alpha_3} \pmod p). \quad (11)$$

The paying device then sends $c_0 := c - \alpha_2 \pmod q$ to the bank. Note that almost all of the workload can be precomputed, except for the modular multiplication by a_0 and the evaluation of $\mathcal{H}(\cdot)$.

Step 3. The bank sends $r_0 := (x_1 + x_2 I_2 + \cdots + x_k I_k)^{-1} (w_0 - c_0) \pmod q$ to the paying device.

Step 4. The paying device computes $r := \alpha_1^{-1} (r_0 + \alpha_3) \pmod q$.

It is not hard to show that if r_0 in Step 3 is such that

$$g_0^{c_0} (g_1 g_2^{I_2} \cdots g_k^{I_k})^{r_0} = a_0 \pmod p, \quad (12)$$

which the paying device can verify if it so desires, then:

- the pair (r, c) is a secret-key certificate of the bank on h ;
- the pair $h, (r, c)$ is completely hidden from the “view” of the bank in the execution of the issuing protocol, regardless of the choice of (I_2, \dots, I_k) ; and
- the secret key (y_1, \dots, y_k) known by the paying device for h is such that $y_1^{-1} y_i = I_i \pmod q$, for all $i \in \{2, \dots, k\}$. (Note that $\alpha_1 = 0 \pmod q$ results in an invalid public key.)

It can be proved, under a plausible assumption, that no conspiracy of attackers, each possibly with their own different tuple (I_2, \dots, I_k) , can feasibly retrieve $l + 1$ different triples by performing l executions of the issuing protocol with the bank, for any $l \geq 0$, even if the executions can be arbitrarily interleaved. It can furthermore be proved that if the bank follows the protocol, then no conspiracy of up to $k - 1$ attackers can feasibly retrieve one triple (secret key, public key, certificate) from the bank for which the secret key does not

contain any of the tuples (I_2, \dots, I_k) used by the bank in Step 3 of each of the protocol executions; if this were not the case, then it would be feasible to compute from scratch a pair $h, (r, c)$ such that $c = \mathcal{H}(h, g^c h^r \bmod p)$ without knowing $\log_g h \bmod q$, which is believed to be infeasible. It is believed that this result holds even for conspiracies of k or more attackers, although this conjecture has not been proved.

We now take $k = 2$, for simplicity, and show that the paying device can use the withdrawn triple to make a payment in accordance with the one-show blinding paradigm: two payments with the same triple enable the bank to compute I_2 , while one payment does not reveal any information correlated to the execution of the protocol in which the triple was obtained. As before, the payment can be either an electronic check or an electronic coin payment. With h equal to $g_1^{y_1} g_2^{y_2} \bmod p$, the paying device can compute an Okamoto digital signature [Okamoto, 1992] on a challenge message m , as follows. It generates two random numbers s_1, s_2 from \mathbb{Z}_q , and computes $b := g_1^{s_1} g_2^{s_2} \bmod p$ and $d = \mathcal{H}(m, h, b)$. It then computes $r_1 := y_1 d + s_1 \bmod q$ and $r_2 := y_2 d + s_2 \bmod q$, and sends the signature (d, r_1, r_2) to the receiving device, together with the public key h and the certificate (r, c) . The receiving device accepts the signature (r_1, r_2) if and only if the following holds:

$$d = \mathcal{H}(m, h, g_1^{r_1} g_2^{r_2} h^{-d} \bmod p). \quad (13)$$

In addition, the receiving device must of course verify the certificate of the bank. At a later stage, the receiving device deposits the payment transcript. The bank verifies it in the same manner as described for the receiving device, and checks for double-depositing and double-spending.

It can be proved that the payment is unconditionally untraceable. Suppose now that the paying device is compromised by an attacker, and the attacker spends the same triple a second time, with respect to another challenge message, m^* , but using the same b . The signature for the new challenge message, m^* , is a triple (d^*, r_1^*, r_2^*) such that

$$d^* = \mathcal{H}(m^*, h, g_1^{r_1^*} g_2^{r_2^*} h^{-d^*} \bmod p). \quad (14)$$

It follows from equations 13 and 14, and the collision-intractability of $\mathcal{H}(\cdot)$, that

$$h = g_1^{(r_1 - r_1^*)/(d - d^*)} g_2^{(r_2 - r_2^*)/(d - d^*)} \bmod p. \quad (15)$$

Since $h = 1$ is not accepted by receiving devices (the bank will not redeem it, because it is not a generator), it follows that $\alpha_1 \neq 0 \bmod q$. But then

$$(r_2 - r_2^*)/(r_1 - r_1^*) = y_2/y_1 = \alpha_1 I_2/\alpha_1 = I_2 \bmod q. \quad (16)$$

To ensure that the attacker cannot use another b for the second spending of the same triple, b must be part of the triple. To this end, the bank requires in the payment protocol the use of a number b that has been hashed along by the paying device when computing c in Step 2 of

the withdrawal protocol; this effectively turns the Okamoto signature scheme into a one-time signature scheme. Clearly, (r, c) must then satisfy $c = \mathcal{H}(h, b, g_0^c h^r \bmod p)$.

This technique can actually be used for a much more general paradigm than one-show blinding, because multiple identifiers can be encoded independently by the issuer. These identifiers may serve the role of credential values (attributes), about which all manner of properties can subsequently be demonstrated in the showing protocol. Specifically, returning to the general form $h = \prod_{i=1}^k g_i^{y_i} \bmod p$, [Brands, 1997] shows how one can rapidly demonstrate that the numbers y_1, \dots, y_k satisfy a satisfiable formula from propositional logic, where the atomic propositions are linear relations modulo q , without revealing anything beyond the validity of the formula. This showing technique enables the paying device to rapidly demonstrate formulas such as

$$\text{"}((5y_2 - 3y_3 = 5) \text{ AND } (2y_3 + 3y_5 = 7)) \text{ OR NOT}(y_2 + 4y_6 - 3y_8 = 5)\text{"} \quad (17)$$

It can be ensured that if and only if the number of formulas (not necessarily the same) demonstrated with respect to the same h exceeds a predetermined threshold, then the bank can compute the secret key (y_1, \dots, y_k) of the triple. In the example electronic cash system described in the Section 4, this general showing protocol technique is used as follows: by taking I_2 to be an identifier of the paying device, and letting I_3 denote a coin denomination specifier of the coin triple, the paying device at payment time can prove to the receiving device the denomination of the coin, without revealing anything about I_2 . To this end, it proves the formula

$$\text{"NOT}(y_1 = 0) \text{ AND } (y_3 = I_3 y_1)\text{"} \quad (18)$$

If the coin is double-spent, I_2 can be computed and hence the coin can be traced to the compromised device.

3.5.4 Guaranteeing Your Own Privacy

The one-show blinding paradigm is not sufficient to design an untraceable electronic cash system. In fact, it is only half of the work. Namely, we have thus far assumed that paying devices are tamper-resistant and are issued by or on behalf of the bank; otherwise an attacker can easily forge money. However, if we do not trust the bank with detailed information about the spending habits of its account holders, can we trust the bank to program paying devices so as to properly protect our privacy? This question is particularly relevant when one realizes that the idea of blinding fundamentally relies on the ability to produce random numbers that are unpredictable to the bank, and that today's smart cards can produce only pseudo-random numbers, on the basis of a random seed value. This seed value must be installed by the bank itself or at least under its supervision, because it must be guaranteed to be random and secret; but with the bank knowing all seed values, it can compute all the

pseudo-random numbers that any paying device will ever compute, and thus the whole idea of blinding becomes pointless. Even if the bank were to tell us that it does not know the seed values, or that paying devices produce random numbers by postprocessing bits sampled from an internal source of “true” randomness (such as a noise diode), this cannot be publicly verified.

When transactions are conducted by directly interfacing a tamper-resistant paying device to a receiving device, it cannot be assessed whether the paying device secretly transfers additional information, such as a device ID, which would also make blinding pointless. A solution to this problem is to let all transfer of information between the paying device and receiving devices flow through a computer that is **interposed** and trusted by the payer (a “user-controlled” computer). Its hardware and software may be purchased on the free market, and a knowledgeable user may even engage in manufacturing his own software and/or hardware. A desktop computer serves as a natural interposed computer when making Internet payments, and a handheld device is a natural candidate for paying at the local grocery store. When the paying device is in the form of a PC Card or a smart card, which must be inserted into a slot or a smart card reader, the user-controlled interposed computer can easily check that no data is transmitted in addition to that specified in the protocol description.

Usage of a user-controlled computer with a keyboard and display also offers another important advantage. Namely, the user can enter his password, PIN or biometric using the computer’s keyboard, and can read out the balance and payment information from the computer’s display instead of having to rely on someone else’s device. This makes fake terminal attacks impossible.

A further advantage is that the user-controlled computer can safeguard the user’s secret keys, and can make, store (for the purpose of non-repudiation) and verify all manner of digital signatures. It can also keep its own transaction logs, provide functions for cash-management, and so on. In addition, it may be able to take over part of the workload and/or storage burden of the tamper-resistant paying device, perhaps even to the extent that the tamper-resistant device does not need to perform any heavy number-theoretic operations.

However, without special measures the paying device may still be able to leak out covert information, by using a **subliminal channel** made available through the protocols used for communicating with the outside world (notably the protocols for withdrawal and payment). This leakage is called **outflow**. As a simple example, consider the paying device in the RSA-based one-show blinding system of Chaum, Fiat and Naor, detailed in Subsubsection 3.5.2: in the secret values a_i , b_i and c_i , some of which are revealed during payment, the paying device can easily encode an identifier and a great deal of other covert information, by using an encoding that only the receiving device, or perhaps only the bank, can recognize. Conversely, a receiving device may be able to transmit messages to the paying device through a subliminal channel, for example to instruct it to halt or to supply outflow in case its internal state adheres the supplied inflow information. This is called **inflow**, and can be achieved for instance by

encoding the instruction into the random part of a challenge message.

In [Chaum, 1988], and later in [Chaum and Pedersen, 1993 (B)], a paradigm is proposed in which the interposed computer performs a much more active role than to merely check for the flow of additional information, outside of the standard protocols. Known as the “wallet-with-observer” paradigm, the idea is that proper cryptographic design of the withdrawal and the payment protocol may enable the interposed computer of the payer to ensure that the blinding factors used in the withdrawal protocol are unpredictable to the bank, and that no subliminal channels can exist.

To ensure that withdrawn coin triples are indeed uncorrelated to the view of the bank, [Chaum, 1988] and [Chaum and Pedersen, 1993 (B)] have the interposed computer develop any blinding factors and coin secret keys jointly with the paying device, in such a manner that they are randomized; randomized numbers cannot contain a subliminal channel.

In [Chaum, 1990], an electronic cash system is described in which the interposed computer can prevent outflow and ensure correctness of the blinding process (a summary appears in [Bos and Chaum, 1990]). Although this system suffers from several shortcomings, described shortly, we review it here for educational purposes and because it is the first (and, for a long time, the only) such system to have been proposed. Payments in the system are made using blank checks, as described in Subsubsection 3.2.1. Blank checks are represented by triples (secret key, public key, certificate), which can be withdrawn from the bank by means of a basic blind signature issuing protocol. The (secret key, public key) pair of a triple is used to make a matrix-based signature, as described in Subsubsection 2.2.2, and the certificate is a blind RSA signature of the bank on the public key (as explained in Subsubsection 3.5.2). The following protocol steps take place:

Step 1. The tamper-resistant paying device and the user-controlled interposed computer together generate a mutually random number. The paying device ensures that the interposed computer does not learn it, while the interposed computer ensures that the check secret key, that will be derived from the random number by the paying device, cannot contain outflow. To this end, the paying device sends a commit on a (pseudo) random number, the interposed computer returns a random number, and the paying device combines the two numbers. The paying device does not (yet) open its commit to demonstrate its honest behavior in this process, because the interposed computer is not allowed to learn secret keys of check triples (this would enable multiple use of a blank check, without needing to break the tamper-resistance of the paying device).

Step 2. The paying device uses the mutually random number to compute the bottom row of a matrix, and to subsequently fill out all the entries in the matrix (as described in Subsubsection 2.2.2). The paying device then computes the check public key by compressing the entries in the top row, and provides the public key to the interposed computer. Note that only the paying device knows the secret key (the entries in the matrix).

Step 3. The interposed computer now offers a blinded form of the public key to the bank, in order to obtain a blind RSA signature on it. The paying device may not develop the blinding factor itself, since in Step 6 the bank could then learn the check public key; on the other hand, the interposed computer may not determine the blinding factor itself, because that would enable it to have the bank sign any information, including public keys for which it knows the check secret key itself. Therefore, the interposed computer and the paying device develop the blinding factor in a mutually random manner, using the method of Step 1. The paying device afterwards opens its commit to show that it behaved properly.

Step 4. Although the interposed computer is assured that the blinding factor is really blind, it has no assurance that the paying device in Step 1 actually used its contribution in the formation of the check secret key. This is of importance because in the payment protocol (Step 7) the paying device will reveal matrix entries, and otherwise there could be a great deal of outflow. Hence at this point the interposed computer can request the paying device to open its commit of Step 1; the decision of whether to make the request or not is made by flipping a (not necessarily unbiased) coin. Because with the opening of the commit the interposed computer learns all the information needed to compute the matrix entry and thus the check secret key, the paying device in this case will not further assist the interposed computer, to prevent it from obtaining a blind RSA signature of the bank; instead, it halts the current protocol execution, and a new protocol execution must be started at Step 1.

Step 5. In case the interposed computer has not requested opening of the commit in Step 4, the protocol execution continues. Because the bank must make sure that it blindly signs only check public keys for which the interposed computer does not know the check secret key, the paying device computes a MAC for the blinded public key (using a secret key known also to the bank; this can be a diversified key) and provides it to the interposed computer. (Alternatively, it could compute a digital signature, but this requires greater computing power.)

Step 6. The interposed computer sends the blinded public key and the MAC to the bank, in the withdrawal protocol. The bank uses the secret key of the paying device to verify the MAC and, if it is correct, returns its RSA root of the blinded public key. The interposed computer removes the blinding factor and verifies the result, as described in Subsubsection 3.5.2. The result of the actions up to this point is that the paying device and the interposed computer have together obtained a blank check triple, with the secret key being known only to the paying device.

Step 7. To make a payment with this blank check, the interposed computer passes the challenge message of the receiving device on to the paying device. The paying device reveals to the interposed computer the proper entries of the secret key, as described in Subsubsection 2.2.2. Upon verifying that these are correct (to prevent outflow), the interposed computing device passes them on to the receiving device.

This protocol has many shortcomings. A first shortcoming is that the paying device needs

to perform a public-key cryptographic operation. Namely, in Step 5 it needs to authenticate the blinded public key, and hence must be able to compute the blinded public key or at least verify its correct formation. As shown in [Bos, 1992], probabilistic verification can be used to alleviate the task for the paying device. To this end, the paying device verifies a modularly reduced version of the blinding factor, using a secret prime modulus that has been installed by the bank during initialization of the paying device. This modulus can be chosen fairly small for a practical security level (typically eight bytes suffices), but its secrecy is of utmost importance and hence each paying device is preferably assigned a unique prime. The downside of this method is that it causes a lot of overhead communication and computation between the paying device and the interposed computer.

Another shortcoming is that the MAC of the paying device can contain outflow: any covert message (sufficiently short in comparison to the size of the MAC, for security) could be exclusive-ored into the MAC by the paying device, and extracted by the bank by removing the MAC. This particular problem can be overcome by a minor variation of Step 6, as described in [Chaum, 1990]: instead of having the interposed computer pass on the MAC in Step 6, the bank computes it by itself and returns, say, the bit-wise exclusive-or of its RSA root and an expanded form of the MAC (for example, the expansion can be the symmetric encryption of the blinded public key, using the MAC as the encryption key). Only in case the interposed computer has received the MAC from the paying device can it remove the expanded MAC and extract the check certificate.

Yet another shortcoming is in the manner in which the interposed computer in Step 4 verifies that the secret key, and hence the signature at payment time, cannot contain outflow: verification takes place using a cut-and-choose strategy, much like with the cut-and-choose approach to one-show blinding. Hence the probability that cheating is detected increases only linearly with the average number of requests for commit opening.

Furthermore, the commit function in Step 1 could have a trapdoor, known to the paying device, in case of which it could always perform the opening in Step 5 correctly and still cause outflow. These problems can be overcome by the more drastic change of using a number-theoretic signature scheme to define the (secret key, public key) pair. In fact, this is the only difference of the protocols in [Chaum, 1988] and [Chaum and Pedersen, 1993 (B)] (neither of which deals with the specific problem of electronic cash design) in comparison to the above protocol of [Chaum, 1990]. As an example, consider using the Schnorr signature scheme: in Step 1, the paying device would select a random $x_1 \in \mathbb{Z}_q$ and send $h_1 := g^{x_1} \bmod p$ to the interposed computer; the interposed computer would return a random $x_2 \in \mathbb{Z}_q$, and the paying device would use $h := g^{x_1+x_2} \bmod p$ as the public key. The interposed computer could then check the correctness by verifying that $h = h_1 g^{x_2} \bmod p$, without needing to know the contribution x_1 of the paying device. Of course, the drawback of this approach is the much greater computational burden for the paying device; it now definitely needs to have a cryptographic co-processor in order to complete its computations in reasonable time.

Another problem that cannot be overcome is that the bank can retroactively trace all payments once the paying device is returned to the bank, if only the paying device stores the random numbers developed in Step 1 or, more efficiently, a few bytes of the random challenges received in Step 7; the bank can then match this information against the deposited information. More generally, the bank at issuing time cannot encode additional information, such as a check spending limit or (in case the triple would serve as a coin) a denomination, without the paying device needing to know it: such information can be encoded only by the choice of the encryption exponent and/or the RSA modulus used by the bank, and the paying device for the purpose of Step 5 needs to know both.

Still another problem is that the interposed computer in Step 7 cannot prevent the receiving device from encoding inflow into the fresh part of its challenge message, unless this is generated in a mutually agreed manner.

However, by far the most serious shortcoming and criticism of the above protocol is that withdrawn triples can be spent many times in an untraceable manner in case the paying device is compromised, because check triples are completely blinded. Incorporating the cut-and-choose blinding technique of [Chaum, Fiat and Naor, 1988] into the above protocol would make the workload for withdrawing a check triple unacceptable: the paying device would have to assist in the construction of all blinded candidates, would have to authenticate all of them, and would have to assist in the opening of half of the candidates.

In [Brands, 1994] and [Brands, 1995 (A)] techniques are introduced that are extensions of the restrictive blinding technique and the showing technique described in Subsubsection 3.5.3. These techniques do not suffer from any of the above shortcomings. Specifically, they enable the paying device and the interposed computer to obtain a triple (secret key, public key, certificate), by performing a restrictive blind issuing protocol with the bank, in such a manner that:

- * The public key and the certificate can be blinded efficiently by the interposed computer, because it may determine the blinding factors by itself. In fact, the paying device need not take part in the withdrawal protocol at all, so that its holder can leave it in a safe place;
- * The paying device assists in making a payment by providing a response to a challenge of the interposed computer. The only computational task that the paying device has to perform is to compute responses, and this can be done so rapidly that the paying device need merely have a simple 8-bit smart card microprocessor (in particular, payments can easily be made using several coins instead of a single check);
- * Part of the secret key of each triple is a unique identifier of the paying device, and the paying device does not need to store dynamically any information from the triple: all it ever needs to do is increment sequence numbers and compute responses;
- * Inflow and outflow in the withdrawal and in the payment protocol are actively prevented by the interposed computer, by blinding all communication between the paying device and the

outside world on the flight (instead of using passive prevention by means of cut-and-choose verification). Moreover, this can be done at virtually no computational cost;

* Even in case the paying device stores all the challenges it receives during payments (this is all it can learn during the time it is held by the account holder), and is returned afterwards to the bank, the stored information cannot be used by the bank to retroactively trace payments by matching it against deposit information; the view of the paying device in the payment protocol is statistically uncorrelated to the view of the receiving device, at least for all receiving device views involving the same number of coins for each denomination. Moreover, by allowing the storage space allocated by the paying device to grow linearly in the number of coins that are withdrawn, it can even be ensured that the paying device cannot learn the denominations of the coins that are spent.

In Section 4 an example is described of a practical electronic cash system that is based on these techniques.

3.5.5 One-Sided Versus Two-Sided Untraceability

The one-show blinding and interposed-computer techniques suffice to offer payer untraceability. Even though payers are untraceable, accounts are all named and hence compliance with existing tax regulations is maintained. Since payees are known to the bank, payments are only **one-sided untraceable**, in the sense that payers can trace their own payments (with the help of the bank). Namely, coins and checks are deposited to named accounts, and can be recognized by their payers. Moreover, by disclosing the blinding factors used to withdraw their coins and checks, payers can provide incontestable proofs of payment. One-sided untraceability makes electronic cash unattractive for criminal uses: although a money launderer or a bribed user may readily accept criminal money at one particular moment, he may not be willing to trust the payer to not give evidence against him at a later stage, and victims of extortion have no reason to withhold evidence at all.

However, it is not sufficient for an electronic cash system to support one-sided untraceability. Indeed, one-sided untraceability can always be converted into **two-sided untraceability** (also called payee untraceability), unless special measures are taken. Namely, an account holder can make a payment to a payee by withdrawing electronic coins or checks using blinded candidates that are provided by the payee; the account holder then merely acts as an intermediary between his account and the payee, providing one-time account access to the payee, and cannot learn what coin or check the payee ends up with. To this end, an extortioner may force the account holder to run modified software on his interposed computer, or a willing account holder may provide two-sided untraceability as a service for money laundering purposes. Because proximity of the payer and the payee is not required at any time, the payee can remain anonymous throughout (for example, on the Internet all communication can be through spoofed IP addresses or anonymous remailers). In the case of fully blinded two-part

coins, or three-part coins and checks, the payee can subsequently at his leisure deposit the coins or checks to his own account, by making a payment to himself; nobody, including the payer, can trace the payment.

To counter this strategy, electronic coins are best implemented in the three-part form, (secret key, public key, certificate), with the secret key containing a unique secret identifier known only to the paying device of the account holder. This is easily achieved by using the restrictive blinding technique. In that case the above strategy enables the other party to receive the public key and the certificate of a blinded coin or check, but not the required secret key. Nevertheless, it may still be possible to use this pair, by letting the paying device of the account holder cooperate also when spending the coin or check, using further active blinding to maintain payee untraceability. By designing the cash system so that the paying device responds only when provided with a challenge message that contains a payee (account) identifier, the untraceable payee cannot make a payment to himself without identifying his identifier to the paying device of the account holder (and hence to the interposed computer of the account holder, if present). Therefore, to maintain payee untraceability, the payee can use the electronic cash only to make a payment to another party, in return for goods or services, which makes the whole idea of conversion to two-sided untraceability pointless, because the account holder might as well have paid the other party directly; moreover, the third party may be of help in tracing.

Note that this measure to prevent two-sided untraceability conversion conflicts with measures to ensure that any information stored by the paying device cannot be used by the bank to retroactively trace payments upon return of the paying device; the (paying device of the) account holder learns the account identifier of the payee. More generally, if the views of the paying device and the payee are statistically uncorrelated, then two-sided untraceability conversion can always be accomplished. For a large-scale electronic cash implementation, it may hence be desirable to prevent only inflow and outflow, while ensuring that the views of the paying device and the receiving device are strongly correlated.

An attempt to circumvent this conflict might be to require the account holder at withdrawal time to demonstrate (in zero-knowledge or otherwise) that its blinding factors have been derived from a secret key, which corresponds to a registered public key of the account holder, according to a predetermined method. However, in the known blind and one-show blind signature issuing protocols this can be achieved only by using completely impractical general multi-party computations or inefficient cut-and-choose verification by the bank. Another serious problem, which moreover is inherent, is that privacy of payments can only be computational: with sufficient computing power the bank can compute the blinding factors by itself, for example by computing the secret key of the account holder.

4 An Example Electronic Cash System

In this section we describe an example electronic cash system, to illustrate many of the foregoing design principles and techniques. For the sake of brevity, the **bank** is denoted by \mathcal{B} , the **account holder** by \mathcal{A} , his (tamper-resistant) **paying device** by \mathcal{P} , his interposed **computer** by \mathcal{C} , and the **receiving device** by \mathcal{R} . Receiving devices need not be tamper-resistant. For the sake of clarity of the description, we do not incorporate fault-tolerance measures.

4.1 Bank Set-Up

\mathcal{B} generates a secret key (x_1, x_2, x_3) and a corresponding public key:

$$p, q, \mathcal{H}(\cdot), (g_0, g_1, g_2, g_3). \quad (19)$$

This key pair will be used for the computation of coin certificates in the restrictive blind coin issuing protocol. As in the restrictive blind issuing protocol in Subsubsection 3.5.3, p and q are primes such that q divides $p-1$; $\mathcal{H}(\cdot)$ is a collision-intractable hash function; x_1, x_2, x_3 are three random numbers from \mathbb{Z}_q ; g_0 is an element of order q in \mathbb{Z}_p^* ; and, $g_1 = g_0^{x_1} \bmod p$, $g_2 = g_0^{x_2} \bmod p$ and $g_3 = g_0^{x_3} \bmod p$.

For concreteness, it is assumed that \mathcal{B} issues electronic coins of denomination 2^{index} , for $\text{index} \in \{0, \dots, l\}$, for some limit l and measured in some appropriate unit (for example, dollar cents); alternatively, any other mapping from the set of valid **index** numbers to the set of valid denominations may be used.

4.2 Opening an Account

When \mathcal{A} opens an account, \mathcal{B} provides \mathcal{A} with a tamper-resistant paying device \mathcal{P} . \mathcal{P} holds a randomly chosen secret key $I \in \mathbb{Z}_q$, installed by \mathcal{B} and serving as an identifier of \mathcal{P} ; \mathcal{B} will be able to trace double-spending to the compromised paying device by computing I . The corresponding public key $h := g_2^I \bmod p$ is made available to \mathcal{C} .

\mathcal{P} uses a pseudo-random number generator, $\text{PRGN}\mathcal{P}$, that takes as inputs triples of the form

$$(\text{seed}, \text{index value}, \text{sequence number}). \quad (20)$$

The seed is a random secret of \mathcal{P} , known also to \mathcal{B} ; for efficiency, we take it to be the same as I . The design of $\text{PRGN}\mathcal{P}$ is such that a simple 8-bit smart card micro-processor can evaluate it within a few hundredths of a second; one can build it, for example, from a one-way hash function with pseudo-random properties, such as SHA.

For each coin denomination, \mathcal{P} keeps track of a sequence number, $\text{seqnum}\mathcal{P}(\text{index})$, which it increments each time when it assists in spending a coin of that denomination. The pseudo-random number output by $\text{PRGN}\mathcal{P}$, on input $(I, \text{index}, \text{seqnum}\mathcal{P}(\text{index}))$, is \mathcal{P} 's contribution

to the secret key of the coin of that denomination and sequence number; \mathcal{B} encodes it into the coin at withdrawal time (note that \mathcal{B} can compute the pseudo-random numbers of \mathcal{P}), and it is needed again to spend the coin.

\mathcal{C} for each coin denomination keeps track of its own sequence number, $\text{seqnum}\mathcal{C}(\text{index})$. It increments this number upon each coin withdrawal. \mathcal{C} also keeps track of a copy of each $\text{seqnum}\mathcal{P}(\text{index})$, and in any case can always request the current values from \mathcal{P} , to stay in synch with \mathcal{P} .

Initially, $\text{seqnum}\mathcal{P}(\text{index})$ and $\text{seqnum}\mathcal{C}(\text{index})$ are set to zero, for each coin denomination. At any moment in time, if \mathcal{C} has properly performed the withdrawal protocol, \mathcal{C} and \mathcal{P} together hold $\text{seqnum}\mathcal{C}(\text{index}) - \text{seqnum}\mathcal{P}(\text{index})$ coins of denomination 2^{index} . In practice, one can use four bytes of storage space for each sequence number, three of which serve to store the actual number and one containing an error-correcting code.

\mathcal{C} also generates its own key pair for message signing purposes, and \mathcal{B} registers \mathcal{C} 's message public key with the account.

4.3 Coin Withdrawal Protocol

To withdraw an electronic coin with denomination 2^{index} , \mathcal{C} and \mathcal{B} perform the following withdrawal protocol:

Step 1. \mathcal{C} sends to \mathcal{B} a digitally signed withdrawal request, specifying the account of \mathcal{A} , index and $\text{seqnum}\mathcal{C}(\text{index})$. To prevent replay of withdrawal requests, the withdrawal request also contains, say, an account access counter.

Step 2. If the signature on the withdrawal request is correct, \mathcal{B} generates two random numbers, $w_0, v \in \mathbb{Z}_q$. The number v is computed in a pseudo-random manner, as follows:

$$v := \text{PRGNP}(I, \text{index}, \text{seqnum}\mathcal{C}(\text{index})). \quad (21)$$

\mathcal{B} then computes

$$a_0 := g_0^{w_0} \bmod p \quad \text{and} \quad u := g_2^v \bmod p, \quad (22)$$

and sends (a_0, u) to \mathcal{C} .

Step 3. \mathcal{C} generates a random number $\alpha_1 \in \mathbb{Z}_q^*$ and five random numbers $\alpha_2, \alpha_3, \alpha_4, \alpha_5, \alpha_6 \in \mathbb{Z}_q$. \mathcal{C} then computes

$$h' := (g_1 h g_3^{\text{index}})^{\alpha_1} \bmod p, \quad (23)$$

$$c := \mathcal{H}(h', u (h')^{\alpha_4} g_2^{\alpha_5} h^{\alpha_6} \bmod p, a_0 g_0^{\alpha_2} (g_1 h g_3^{\text{index}})^{\alpha_3} \bmod p) \quad (24)$$

and

$$c_0 := c - \alpha_2 \bmod q. \quad (25)$$

\mathcal{C} then sends c_0 to \mathcal{B} . (Note that \mathcal{C} can perform the bulk of the workload in a precomputation phase, before connecting to \mathcal{B} ; virtually the only on-line computations are the multiplications by a_0 and u .)

Step 4. \mathcal{B} computes

$$r_0 := (x_1 + x_2 I + x_3 \text{index})^{-1}(w_0 - c_0) \bmod q, \quad (26)$$

charges the account of \mathcal{A} by 2^{index} units, and sends r_0 to \mathcal{C} . \mathcal{B} also increases the account access counter for \mathcal{A} by one.

Step 5. \mathcal{C} computes

$$r := \alpha_1^{-1}(r_0 + \alpha_3) \bmod q. \quad (27)$$

\mathcal{C} stores $(\alpha_1, \alpha_4, \alpha_5, \alpha_6)$ and (r, c) onto its “coin stack” of denomination 2^{index} (note that these are all “small” numbers), indexed by $\text{seqnum}\mathcal{C}(\text{index})$, and increments $\text{seqnum}\mathcal{C}(\text{index})$ and its account access counter both by one.

In case \mathcal{A} wants to withdraw many coins, not necessarily having the same denomination, the request in Step 1 should specify all coins requested; Steps 2 through 5, which must be performed once for each coin, can be performed in parallel for all coins. The withdrawal protocol thus consists of four message transmissions, independent of the number of coins requested.

For authentication and non-repudiation, the messages sent in Steps 2 through 4 should also be digitally signed by the sender. The digital signature of \mathcal{C} on its message in Step 3 (and details of the preceding messages) can serve to \mathcal{B} as a proof that \mathcal{C} has requested the withdrawal of the specified amount, and authorizes \mathcal{B} to debit \mathcal{A} ’s account. By requiring \mathcal{B} to be able to resend its response(s) in Step 4 whenever requested to do so, at least until a time-out or until \mathcal{C} requests withdrawal of a new set of coins (this proves that \mathcal{C} has successfully completed the previous withdrawal session), it is ensured that \mathcal{B} obtains \mathcal{C} ’s debit authorization if and only if \mathcal{C} is able to obtain the requested coins (possibly with the help of a judge).

In Step 5, \mathcal{C} may wish to verify that

$$g_0^{c_0} (g_1 h g_3^{\text{index}})^{r_0} = a_0 \bmod p. \quad (28)$$

For efficiency, this verification can normally be omitted. In particular, if r_0 is incorrect then \mathcal{C} will notice this because \mathcal{R} will not accept when the coin is spent; \mathcal{C} can then use the digital signature of \mathcal{B} on its message in Step 5, to demonstrate that r_0 has been formed incorrectly by \mathcal{B} . This requires \mathcal{C} to store the digital signature of \mathcal{B} on its message in Step 4 (and details of the preceding messages) at least until the withdrawn coins have been spent.

Furthermore, all messages may be encrypted, for confidentiality. This can be done using symmetric encryption, using a mutually known random session key; \mathcal{C} can generate the session

key in Step 1, and send it along with the withdrawal request in encrypted form (using a special encryption public key of \mathcal{B}). For efficiency, the RSA signature scheme may be used for message signing, the RSA encryption scheme for session key encryption, and triple-DES for message encryption.

4.4 Coin Payment Protocol

If $\text{seqnum}\mathcal{C}(\text{index})$ exceeds $\text{seqnum}\mathcal{P}(\text{index})$ then \mathcal{C} and \mathcal{P} together hold $\text{seqnum}\mathcal{C}(\text{index}) - \text{seqnum}\mathcal{P}(\text{index})$ coins of denomination 2^{index} . To transfer a coin to \mathcal{R} , \mathcal{C} and \mathcal{R} perform the following payment protocol (with \mathcal{P} necessarily assisting \mathcal{C}):

Step 1. A challenge message m is determined by \mathcal{C} and \mathcal{R} , by concatenating an account identifier of \mathcal{R} , the number index , and a fresh part (such as a random number of \mathcal{R} , or a sufficiently accurate time/date estimate by \mathcal{C} that is approved by \mathcal{R}). \mathcal{C} retrieves $(\alpha_1, \alpha_4, \alpha_5, \alpha_6)$ and (r, c) , indexed by $\text{seqnum}\mathcal{P}(\text{index})$ in its coin stack for coins of denomination 2^{index} . \mathcal{C} then recomputes h' from α_1 (of course, \mathcal{C} could alternatively have stored h' in its coin stack at withdrawal time), and computes

$$d := \mathcal{H}(m, h', (r, c)) \quad \text{and} \quad e := d + \alpha_6 \bmod q. \quad (29)$$

\mathcal{C} then sends e to \mathcal{P} , together with index , to indicate the denomination of the coin it wants to spend.

Step 2. \mathcal{P} computes

$$y := Ie + \text{PRGN}\mathcal{P}(I, \text{index}, \text{seqnum}\mathcal{P}(\text{index})) \bmod q, \quad (30)$$

increments $\text{seqnum}\mathcal{P}(\text{index})$ by one, and sends y to \mathcal{C} .

Step 3. \mathcal{C} computes

$$r_1 := y + \alpha_5 \bmod q, \quad \text{and} \quad r_2 := -\alpha_1^{-1}d + \alpha_4 \bmod q \quad (31)$$

and sends

$$h', (r, c), (d, r_1, r_2) \quad (32)$$

to \mathcal{R} (and possibly also missing details of m).

Step 4. For the challenge message m approved or decided on by \mathcal{R} , \mathcal{R} verifies that

$$c = \mathcal{H}(h', g_1^d g_2^{r_1} g_3^{d \text{index}} (h')^{r_2} \bmod p, g_0^c (h')^r \bmod p) \quad \text{and} \quad d = \mathcal{H}(m, h', (r, c)). \quad (33)$$

If the verification holds, \mathcal{R} accepts the coin payment.

In Step 3, \mathcal{C} may wish to verify that

$$g_2^y h^{-e} = u \bmod p. \quad (34)$$

For efficiency this can normally be omitted. Even if the verification is never performed, if y is incorrect then \mathcal{R} will not accept the payment, while at most one bit of outflow can result (namely, whether the coin is correct or not).

In a typical scenario, this payment protocol is performed simultaneously for multiple coins of suitable denominations, in order to make up the exact amount payable. For increased efficiency, a single m and a single d can be used for all coins. To this end, m must specify `index` for each of the coins (alternatively, and more compactly, the payable amount may be specified), and to compute d the hash function $\mathcal{H}(\cdot)$ must take as inputs m and, for each coin, $(h', (r, c))$ of that coin.

In Step 4, \mathcal{R} can send a digitally signed receipt to \mathcal{C} . For confidentiality, \mathcal{C} and \mathcal{R} may use session key encryption. To this end, \mathcal{C} can generate a random session key in Step 3, publicly encrypt it using a (certified) public key of \mathcal{R} , symmetrically encrypt the payment message using the session key, and send both encryptions over to \mathcal{R} .

4.5 Coin Deposit Protocol

To deposit the received coin at a convenient moment later on, \mathcal{R} performs the following deposit protocol with \mathcal{B} :

Step 1. \mathcal{R} sends to \mathcal{B} the payment transcript:

$$h', (r, c), (d, r_1, r_2), m. \tag{35}$$

Step 2. \mathcal{B} verifies that the purportedly fresh part of the challenge message m has not been used before by \mathcal{R} (note that for this reason the use of an accurate time and date indication is preferred, since \mathcal{B} then merely needs to store the most recent time and date indication of \mathcal{R}); this excludes double-depositing by \mathcal{R} . If this is the case, \mathcal{B} verifies the payment transcript in the same way as specified for \mathcal{R} in Step 4 of the payment protocol, reading `index` from m . If the verification holds, \mathcal{B} credits the account that is indicated by the account identifier in m by 2^{index} units.

Of course, \mathcal{R} can deposit multiple payment transcripts at once, and should preferably sign its deposit request in Step 1. Likewise, \mathcal{B} in Step 2 should return a digitally signed statement, to inform \mathcal{R} of which payment transcripts have been accepted. To hide who is depositing how much, both parties can encrypt their messages using a session key suggested by \mathcal{R} , which \mathcal{R} sends to \mathcal{B} in public-key encrypted form.

4.6 Forgery Detecting and Tracing

At a suitable moment, which might be at the time of deposit by \mathcal{R} but could also be later on, \mathcal{B} checks for double-spending. To this end, \mathcal{B} verifies whether $\mathcal{H}(h')$ already appears in

a “forgery-detect” database. If this is the case, \mathcal{B} subsequently traces the physically compromised paying device, by retrieving from a “forgery-trace” database the “old” pair (d^*, r_1^*) (indexed by the hash of h' that is already in the forgery-detect database) and computing

$$(d - d^*)^{-1}(r_1 - r_1^*) \bmod q; \tag{36}$$

this number is equal to I . In addition, \mathcal{B} can blacklist $\mathcal{H}(h')$.

On the other hand, if the coin has not been double-spent, \mathcal{B} stores $\mathcal{H}(h')$ in the forgery-detect database and the pair (d, r_1) in the forgery-trace database, indexed by $\mathcal{H}(h')$. The forgery-detect database can be maintained on computer hard-disks, while the forgery-trace database can be stored on tape or WORM disks.

4.7 Discussion

There are several noteworthy aspects in the design of this example cash system. Firstly, the withdrawal protocol is the restrictive blind issuing protocol discussed in Subsubsection 3.5.3. Secondly, the interposed computer computes the blinding factors by itself, and in fact the tamper-resistant paying device does not take part in the withdrawal protocol. When assisting in a coin payment, the paying device performs a simple computation that can be performed rapidly by a simple smart card processor. Furthermore it does not store any part of the withdrawn coins; it merely increments coin sequence numbers that occupy preallocated storage space, which can be well below 100 bytes in a practical implementation. Hence the desired low complexity for paying devices is achieved. Thirdly, the receiving device need not be tamper-resistant, and payment transcripts can be deposited only to the account of the payee indicated in the challenge message. Finally, the active blinding by the interposed computer in the payment protocol ensures unconditional protection against inflow and outflow.

Moreover, it can be proved that even if \mathcal{P} stores the challenges e received in all executions of the payment protocol performed by \mathcal{C} , and \mathcal{B} analyzes all information stored by \mathcal{P} upon its return, the payments of \mathcal{C} still cannot be traced, regardless of the strategy followed by \mathcal{B} , \mathcal{P} and all receiving devices, and regardless of their computing power. Of course, this untraceability holds only amongst all payments made with the same number of coins and of the same denominations, because \mathcal{P} does learn the coin denominations involved in a payment. This can be improved on, by using a single list of sequence number instead of one for each denomination, but then the amount of data that must be stored by the paying device grows linearly instead of logarithmically with the number of coins that have been withdrawn. Namely, at withdrawal time it cannot be predicted in which order coin denominations will be addressed at payment time, and in the worst case the paying device must hop almost randomly through the list of sequence numbers to select coins of the appropriate denomination, keeping track of all the sequence numbers used thus far.

In order to keep the size of the forgery-detect and forgery-trace databases manageable, \mathcal{B} should regularly change its own secret key for certifying coins. Once coins have expired for

deposit, \mathcal{B} can erase the databases or archive them. The use of coin expiration dates also helps to contain the financial damage that can be done by an attacker who gets hold of \mathcal{B} 's secret certification key. In fact, for this purpose it is highly desirable that \mathcal{B} can at any instant declare the current coin version invalid, and start issuing coins using a new certification key. To make this work smoothly in practice, coin versions should be given a separate expiration dates for at least the withdrawal and the deposit protocols.

In view of the possibility of a crash, loss or theft of \mathcal{C} and/or \mathcal{P} , \mathcal{C} should generate backup copies of all its coins. \mathcal{C} can make a backup each time when it withdraws new coins, and any coins that have been spent since the last backup can be removed or overwritten. Specifically, for each withdrawn coin that has not yet been spent, a backup entry is kept of the form

$$\text{index}, \alpha_1, b, (r, c). \tag{37}$$

Here, b denotes $u(h')^{\alpha_4} g_2^{\alpha_5} h^{\alpha_6} \bmod p$; this number is computed by \mathcal{C} in Step 3 of the withdrawal protocol, for inclusion in the hash function. (To reduce storage space, $\mathcal{H}(b)$ can be stored instead of b itself; the definition of the coin certificate, (r, c) , must then be modified correspondingly.) In case \mathcal{A} wants to recover, \mathcal{C} reads the backup entries, and sends them to \mathcal{B} . Of course, in case \mathcal{C} was involved in the crash, loss or theft, this requires \mathcal{A} to first obtain a new \mathcal{C} and/or to reinstall \mathcal{C} 's software. For each coin, the provided entry information is sufficient for \mathcal{B} to reconstruct h' and to check (r, c) , and in particular to verify that the information indeed specifies a coin it had issued to \mathcal{A} . \mathcal{B} reimburses \mathcal{A} for all the coins that have not been spent, to which end it matches h' against the forgery-detect database. In case the paying device has been reported lost or stolen, \mathcal{B} should not unconditionally reimburse \mathcal{A} until the expiry date for payment or deposit of the current coin version, to prevent abuse of the recovery procedure for the purpose of double-spending. Note that the recovery method is independent of the manner in which blinding factors are computed, and of whether or not the account holder has a paying device at his disposal at the time of recovery (issuing a new one requires a physically secure channel, while \mathcal{A} may want to suspend his account). Moreover, the backup information cannot be used by a thief to spend the coins, since part of the secret key of each coin triple is not stored. Finally, \mathcal{B} can incontestably prove to \mathcal{A} that a coin offered for recovery appears already in its deposit database by showing the signature (d, r_1, r_2) , which it could not have made by itself. To this end it also needs to store r_2 at deposit time.

For cross-platform portability, the same paying device can be used in combination with different user-controlled computer devices. Of course, this requires a mechanism for the various user-controlled computers to exchange coin information, since coins withdrawn using one computer may need to be spent using another.

In line with our earlier discussions, a minor variation allows the protocols to be used for electronic checks, which can be spent for any amount up to a predetermined maximum. A special value of `index` can be used to denote checks (alternatively, several extra values are

defined, each specifying a different maximum spending limit). The only required change to the protocols is that the challenge message m in the payment protocol must now specify in addition the amount for which the check is filled out, and d must be computed by \mathcal{P} itself in Step 2; correspondingly, there is no use for α_6 anymore, and \mathcal{C} in Step 1 of the payment protocol must provide \mathcal{P} with m , h' and (r, c) , instead of with e . (Alternatively, \mathcal{C} provides m and a hash of $(h', (r, c))$ to \mathcal{P} , and \mathcal{P} computes d by hashing these two numbers. This approach is especially convenient when using a single m and d for multiple coins, because \mathcal{P} can then be given m and a hash of all $(h', (r, c))$, and these two numbers then form the input to $\mathcal{H}(\cdot)$ for computing d ; of course, the verification by \mathcal{R} and \mathcal{B} must be modified correspondingly. Furthermore, parts of m may be hashed, perhaps even with a random salt, to prevent \mathcal{P} from learning or recognizing them.) Before sending out y in Step 2, \mathcal{P} must decrement its register value by the amount for which the check is filled out. Note that now an additional protocol is needed, for withdrawing electronic cash; what we called the withdrawal protocol for coins is now the issuing protocols for blank checks. This can easily be constructed using the conventional authentication techniques described in Subsection 2.2.

The check variation has the drawback that \mathcal{B} can retroactively trace all payments of \mathcal{C} once \mathcal{P} is returned to \mathcal{B} , if only \mathcal{P} stores for each execution of the payment protocol m or a relevant part of it. On the other hand, having \mathcal{P} compute d by itself has the advantage for law enforcement that one-sided untraceability cannot be converted into two-sided untraceability, as discussed in Subsubsection 3.5.5. This adjustment can also be applied to the described coin protocols.

A more serious drawback of the check variation, which as we have seen is unavoidable, is that by physically extracting the secret key I of \mathcal{P} , an attacker can introduce counterfeit without detection by the bank. The following trade-off is believed to be of practical interest: the bank issues coins as well as checks (in both cases preventing conversion to two-sided untraceability), and paying devices use checks only to pay fractional amounts that would otherwise require too many coins. The bank specifies a low spending limit for checks, and monitors the number of checks withdrawn by each of its account holders in order to limit excessive check withdrawal. This ensures that an attacker cannot make a significant profit without being exposed.

5 Summary and Research Issues

As we have seen in this chapter, techniques for designing electronic cash systems vary widely and range from elementary to complex. Assuming that it is infeasible for attackers to compromise tamper-resistance, register-based electronic cash is preferable over electronic coins for reason of efficiency. We have seen that payment authentication must take place on the basis of dynamic authentication, in the form of challenge-response protocols, to prevent replay attacks. Authentication can be performed using symmetric cryptography, whereby MACs are

communicated, or by using public-key cryptography. The former approach is (in general) much more efficient than the latter, but the presence of system-wide secrets in receiving devices is inherent to this approach, and hence tamper-proofness of receiving devices is critical. In the latter approach, it is not of concern to the bank whether receiving devices are tamper-resistant, if only electronic cash is issued in the form of coins or checks in the three-part form. This can be accomplished in one of the following two ways: either paying devices withdraw electronic coins, or they hold register-based cash and digitally sign the payable amount using a blank check. When tamper-resistance can be relied upon, achieving security against forgery is mainly a matter of designing cash transfer protocols that resist cryptanalysis by wire-tappers.

We have also seen that an important aspect of system design is to ensure security for the bank, under the assumption that secrets in tamper-resistant devices can be extracted and registers bypassed. It is here that crucial design decisions must be made. When electronic cash is represented in the form of register-based cash, a forgery can be detected by the bank only by monitoring how much cash ought to be present in each paying device at various moments in time. This requires receiving devices to deposit transaction transcripts that reveal at least a paying device ID and the transferred amount. In case receiving devices are tamper-proof, the bank can trust receiving devices for performing truthful depositing of transaction logs. In case receiving devices are not tamper-resistant, or at least their tamper-resistance is not relied on, accurate depositing of transaction details can be ensured only by issuing electronic cash in the three-part form (coins or blank checks).

Because such monitoring is in conflict with privacy of payments, we have also discussed techniques for incorporating untraceability of payments. The straightforward approaches of relaxed monitoring, anonymous accounts and anonymously issued paying devices were all seen to offer only a very weak form of privacy (all payments are linkable, and trust in the bank or receiving devices is required), and moreover inevitably cause a trade-off with security for the bank. Much better are cryptographic techniques based on the paradigm of blinding. When electronic coins or check triples are withdrawn by means of a basic blind signature protocol, all information can be blinded by the withdrawing party, and forgery can at best be detected by the bank (untraceability is perfect even for attackers who manage to forge electronic cash). An important improvement is one-show blinding, for which two techniques exist: cut-and-choose blinding and restrictive blinding. The former is very expensive, and for practical purposes only the latter is acceptable.

The one-show blinding paradigm is only half of the work that is needed to design an untraceable electronic cash system, because paying devices that are tamper-resistant cannot be guaranteed to compute random blinding factors, or to otherwise follow the protocols; and in case paying devices are not tamper-resistant but user-controlled, there is no prior restraint of double-spending. As we have seen, the other half of the work can be accomplished by having the account holder interpose a computer of his own between his paying device and any

outside devices, to ensure that the paying device cannot leak information that helps the bank to trace his payments. The cash system should then be designed such that the interposed computer can prevent inflow and outflow by blinding on the flight all communication between the paying device and the outside world, and during withdrawal can compute the blinding factors either by itself or randomize them.

By ensuring that paying devices are provided with the challenge information that needs to be signed at payment time, one-sided untraceability cannot be converted into two-sided untraceability, thus making electronic cash unattractive for criminal uses such as money laundering, bribery and extortion. If this property is not deemed relevant it can even be ensured that paying devices cannot learn information that enables the bank to trace payments when analyzing their contents upon return. Finally, we have seen a detailed example of a practical electronic cash system, based on the discussed principles and techniques.

An important area for further research is to rigorously prove the cryptographic security of practical electronic cash systems, by proving all manner of attacks as hard as breaking well-understood cryptographic primitives. For a start in this direction, see [Brands, 1995 (B)].

6 Glossary

Blinding: A paradigm according to which a receiver in an execution of a protocol obtains digitally signed information that remains hidden from the issuer. Also refers to the cryptographic actions by a party interposed between two other parties in a cryptographic protocol, to destroy subliminal channels.

Cut-and-choose blinding: A technique for designing an issuing protocol for the one-show blinding paradigm. A great many basic blind issuing protocols are performed in parallel, and the signer completes the blind signature issuing protocol only for some of these, after having verified for the rest that the blinded candidates contain an identifier.

Diversified key: A secret key that is computed by hashing at least a master secret key and an identifier, using a one-way hash function.

Dynamic authentication: A method for a device to prove its authenticity, in such a manner that replay attacks have negligible or zero probability of success.

Electronic check: A method for transferring electronic cash. A blank check is a triple (secret key, public key, certificate), issued by the bank to a paying device that holds cash in register-based form. At payment time the check is filled out by the paying device for any amount up to a predetermined maximum amount, by signing the amount and subtracting it from its internal cash balance. Blank checks are not prepaid, because they do not carry value.

Electronic coin: A method for representing electronic cash. A coin is a prepaid public-key cryptographic token, digitally signed by the bank, to which a fixed value and currency are assigned at issuing time. Coins can be either in the two-part form, (message, signature), or

in the three-part form, (secret key, public key, certificate).

Inflow: Covert information that is communicated by an outside device to a paying device, through a subliminal channel in a system protocol.

One-show blinding: A cryptographic paradigm, requiring the design of a signature issuing protocol and a corresponding signature showing protocol, such that showing an obtained signature once is untraceable, while showing it twice allows a built-in identifier to be computed.

Outflow: Covert information that is communicated by a paying device to an outside device, through a subliminal channel in a system protocol.

Register-based cash: A method for representing electronic cash. The amount of cash held by a tamper-resistant device is indicated by the value of a counter, maintained in a register in a chip.

Replay attack: An attack whereby information revealed by a party when proving its authenticity is re-used to pass a subsequent authenticity test.

Restrictive blinding: A technique for designing an issuing protocol for the one-show blinding paradigm. The issuer can issue a triple (secret key, public key, certificate) in such a manner that the receiver can blind the public key and the certificate, but not a non-trivial blinding-invariant part of the secret key; in this part one or more identifiers can be encoded.

Static authentication: A method whereby one device proves its authenticity to another by always revealing the same predetermined secret, such as a password or an identifier. Not secure against replay attacks.

Subliminal channel: A channel present in a cryptographic protocol, by means of which a party can secretly communicate information to another party, in a manner unrecognizable by interposed parties.

Untraceability: In an untraceable electronic cash system, payments cannot be traced to the payer by examining information revealed through system protocols, and payments by the same payer are unlinkable. Untraceable cash systems can be designed using cryptographic blinding techniques.

7 References

- Bos, J. Verification of RSA Computations on a Small Computer. 1992. *Practical Privacy*. Ph.D. thesis, ISBN 90-6196-405-9.
- Bos, J. and Chaum, D. 1990. SmartCash: A Practical Electronic Payment System. *Centrum voor Wiskunde en Informatica REPORT*. No. CS-R9035.
- Bleichenbacher, D. and Maurer, U. 1994. Directed Acyclic Graphs, One-way Functions and Digital Signatures. *Advances in Cryptology – CRYPTO '94*. Yvo G. Desmedt, editor. Lecture Notes in Computer Science, Volume 839, Springer-Verlag, pp. 75–82.
- Boneh, D., DeMillo, R. and Lipton, R. 1997. On the Importance of Checking Computations for Faults. *Advances in Cryptology – EUROCRYPT '97*. Walter Fumy, editor. Lecture

- Notes in Computer Science, Volume 1233, Springer-Verlag, pp. 37–51.
- Brands, S. 1994. Untraceable off-line cash in wallets with observers. *Advances in Cryptology – CRYPTO '93*. Douglas R. Stinson, editor. Lecture Notes in Computer Science, Volume 773, Springer-Verlag, pp. 302–318.
- Brands, S. 1995 (A). Electronic Cash on the Internet. *Proceedings of the ISOC Symposium on Network and Distributed System Security*. San Diego, California, February 16-17, pp. 64–84.
- Brands, S. 1995 (B). Off-Line Electronic Cash Based on Secret-Key Certificates. *Proceedings of the Second International Symposium of Latin American Theoretical Informatics*. R. Baeza-Yates, E. Goles, P. V. Goblete, editors. Lecture Notes in Computer Science, Volume 911, Springer-Verlag, pp. 131–166.
- Brands, S. 1995 (C). Secret-Key Certificates. *Centrum voor Wiskunde en Informatica REPORT*. Report CS-R9510.
- Brands, S. 1995 (D). Restrictive Blinding of Secret-Key Certificates. *Advances in Cryptology – EUROCRYPT '95*. Louis C. Guillou and Jean-Jacques Quisquater, editors. Lecture Notes in Computer Science, Volume 921, Springer-Verlag, pp. 231–247.
- Brands, S. 1995 (E). Secret-Key Certificates (Continued). *Centrum voor Wiskunde en Informatica REPORT*. Report CS-R9555.
- Brands, S. 1997. Rapid Demonstration of Linear Relations Connected by Boolean Operators. *Advances in Cryptology – EUROCRYPT '97*. Walter Fumy, editor. Lecture Notes in Computer Science, Volume 1233, Springer-Verlag, pp. 318–333.
- Brands, S. and Chaum, D. 1994. Distance Bounding. *Advances in Cryptology – EUROCRYPT '93*. Tor Helleseth, editor. Lecture Notes in Computer Science, Volume 765, Springer-Verlag, pp. 344–359.
- Brickell, E., Gemmell, P. and Kravitz, D. 1995. Trustee-Based Tracing Extensions to Anonymous Cash and the Making of Anonymous Change. *Proceedings of the 6th Annual Symposium on Discrete Algorithms*. pp. 457–466.
- Camenisch, J., Maurer, U. and Stadler, M. 1996. Digital Payment Systems with Passive Anonymity-Revoking Trustees. *Proceedings of Computer Security - ESORICS '96*, Lecture Notes in Computer Science, Volume 1146, Springer-Verlag, pp. 31-43.
- Chaum, D. 1983. Blind Signatures for Untraceable Payments. *Advances in Cryptology – CRYPTO '82*. R. L. Rivest and A. Sherman and D. Chaum, editors. Volume 0, Plenum Press, pp. 199–203.
- Chaum, D. 1988. Card-Computer Moderated Systems. Patent no. US 4,926,480.
- Chaum, D. 1990. Optionally moderated transaction systems. Patent no. EP 0 439 847 A1.
- Chaum, D., Fiat, A. and Naor, M. 1988. Untraceable Electronic Cash. *Advances in Cryptology – CRYPTO '88*. S. Goldwasser, editor. Lecture Notes in Computer Science, Volume 403, Springer-Verlag, pp. 319–327.
- Chaum, D. and Pedersen, T. 1993 (A). Transferred Cash Grows in Size. *Advances in Cryptol-*

- ogy – *Proceedings of EUROCRYPT '92*. R.A. Rueppel, editor. Lecture Notes in Computer Science, Volume 658, Springer-Verlag, pp. 390–407.
- Chaum, D. and Pedersen, T. 1993 (B). Wallet Databases with Observers. *Advances in Cryptology – CRYPTO '92*. Ernest F. Brickell, editor. Lecture Notes in Computer Science, Volume 740, Springer-Verlag, pp. 89–105.
- Diffie, W. and Hellman, M. 1976. New Directions in Cryptography. *IEEE Transactions on Information Theory*. Volume IT-22, pp. 644–654.
- Eng, T. and Okamoto, T. 1994. Single-Term Divisible Electronic Coins. *Advances in Cryptology – Proceedings of EUROCRYPT '94*. Alfredo De Santis, editor. Lecture Notes in Computer Science, Volume 950, Springer-Verlag, pp. 306–319.
- Even, S., Goldreich, O. and Micali, S. 1996. On-Line/Off-Line Digital Signatures. *Journ. of Cryptology*, Volume 9, No. 1, pp. 35–67.
- Even, S., Goldreich, O. and Yacobi, Y. 1984. Electronic Wallet. *Advances in Cryptology – Proceedings of CRYPTO '83*, D. Chaum, editor. Volume 0, Plenum Press, pp. 383–386.
- Froomkin, M. 1996. Flood Control on the Information Ocean: Living With Anonymity, Digital Cash and Distributed Databases. *15 Pitt. J. Law & Commerce*, No. 395.
- Gray, J. and Reuter, A. 1993. *Transaction Processing: Concepts and Techniques*. Morgan Kaufman Publishers, Inc., San Francisco, ISBN 1-55860-190-2.
- IEEE Spectrum. February 1997. *Technology and the Electronic Economy*. (Special issue on electronic money.)
- Jones, D. July 1997. *Mondex: A House of Smart-Cards?; With e-cash, privacy is illusory and security is questionable*. The Convergence.
- Lamport, L. 1979. Constructing Digital Signatures from a One Way Function. *SRI International Report*. No. CSL-98.
- Law, L., Sabett, S. and Solinas, J. 1996. How to Make a Mint: the Cryptography of Anonymous Electronic Cash. National Security Agency, Office of Information Security Research and Technology, Cryptology Division, June 14.
- Merkle, R. 1985. Matrix Digital Signature For Use with the Data Encryption Algorithm. *IBM Tech. Discl. Bulletin*, Volume 28, No. 2, pp. 603–604.
- Merkle, R. 1988. A Digital Signature Based on a Conventional Encryption Function. *Advances in Cryptology – CRYPTO '87*. Carl Pomerance, editor. Lecture Notes in Computer Science, Volume 293, Springer-Verlag, pp. 369–378.
- Okamoto, T. 1992. Provably Secure and Practical Identification Schemes and Corresponding Signature Schemes. *Advances in Cryptology – CRYPTO '92*. Ernest F. Brickell, editor. Lecture Notes in Computer Science, Volume 740, Springer-Verlag, pp. 31–53.
- Pedersen, T. 1995. Electronic Payments of Small Amounts. *Aarhus University Technical Report*, DAIMI PB-495, Denmark.
- Rivest, R., Shamir, A. and Adleman, L. 1978. A Method for Obtaining Digital Signatures and Public-Key Cryptosystems. *Comm. ACM*, Volume 21, pp. 120–126.

- Schnorr, C. 1991. Efficient Signature Generation by Smart Cards. *Journ. of Cryptology*, Volume 4, No. 3, pp. 161–174.
- Vaudenay, S. 1993. One-Time Identification with Low Memory. *Proceedings of EUROCODE '92*. Udine, Italy, CISM Courses and lectures, No. 339, Springer-Verlag, pp. 31–53.
- Waidner, M. and Pfitzmann, B. 1990. Loss-Tolerance for Electronic Wallets. *20th International Symposium on Fault-Tolerant Computing*, Newcastle upon Tyne, pp. 140–147.

8 Further Information

8.1 Literature

Public literature on the design of electronic cash systems is scarce and dispersed. Since 1995 several books have been published about electronic commerce, but these are all non-technical and focus on credit-card systems. The main source for articles on the cryptographic design of electronic cash systems are the annual EUROCRYPT and CRYPTO conferences. Interesting technical information may also be found in proceedings of smart-card conferences, and by entering relevant keywords into a search engine for the World-Wide-Web. Patents are another useful source of information on techniques related to electronic cash systems.

In this chapter we have not discussed all manner of functionality extensions and variations that have been proposed for untraceable electronic cash. To name a few: off-line transferability for coins in the three-part form is discussed in [Chaum and Pedersen, 1993 (A)]; a technique for tick payments for blank checks is introduced in [Pedersen, 1995]; escrow functionality for electronic coin systems is introduced in [Brickell, Gemmell and Kravitz, 1995], and improved by [Camenisch, Maurer and Stadler, 1996]; and divisibility of electronic coins is addressed, amongst others, by [Eng and Okamoto, 1994]. An account of legal aspects of untraceable electronic cash can be found in [Fromkin, 1996], and a compact overview of cryptographic techniques for untraceable cash in [Law, Sabett and Solinas, 1996]. The february issue of [IEEE Spectrum, 1997] is devoted entirely to technological aspects of electronic money.

8.2 Electronic Cash Today

Most prepaid electronic purse initiatives are based on symmetric authentication and encryption. Many of these are modeled after the Proton system, a non-anonymous register-based cash system developed by Banksys, the association of Belgium banks. One of first electronic purse projects of significance has been a Danish initiative allied to Visa International, called Danmønt. Other national purse scheme initiatives are underway in the Netherlands (Chipknip, by Interpay), Switzerland (Telekurs bank consortium) and Portugal (Multibanco Electronic Purse). A related technology called Visa Cash has been developed by Visa International, and has been piloted at the 1996 Summer Olympics.

International standardization efforts for electronic purse schemes are the CEN Intersector Electronic Purse and the EMV 3.0 specifications.

Another development is the Mondex prepaid system of Mondex International Limited, which enables repeated off-line transferability of value from person to person. All devices are tamper-resistant, and offer a keyboard and display. The version currently in operation uses conventional cryptographic authentication. In September 1995, a Fair Trading Act complaint was filed against Mondex for falsely advertising their system as anonymous, and the claim was upheld after a nine month investigation by the Fair Trading Office; see [Jones, 1997] and the references provided therein.

Citibank is developing a cash system called the Electronic Monetary System (EMS), with functionality similar to that of Mondex. In particular, cash can repeatedly be transferred from person to person, without the involvement of a central party, and all devices (Money Modules) are tamper-resistant and have a keypad a display. Instead of using a register-based cash representation, cash appears to be stored and transferred in the form of two-part coins. Each note carries a complete electronic audit trail, and (invisibly to the user) electronic notes are submitted to the bank for validation and control whenever a withdrawal or deposit is made. Security is based on the ability of the bank to trace all transactions to devices.

CAFE is a European Commission sponsored project, involving thirteen leading European parties from academic research and industries, that has tested an off-line electronic cash system in the Commission headquarters in Brussels. The CAFE system is based on the one-show blinding paradigm and the wallet-with-observer paradigm, and more specifically on the public-key cryptographic techniques of Brands for withdrawing, paying and depositing. Transactions are conducted using a tamper-resistant smart card inserted into a handheld computer with a keyboard and display, and limited off-line transferability is offered.

Four financial institutions (the Mark Twain bank, the Merita bank, the Deutsche bank and the Advance bank) and the Sweden Post offer (or are about to) a system for untraceable electronic cash payments over the Internet. This system is a software-only on-line payment system based on Chaum's basic blind RSA signatures, developed by Amsterdam-based Digi-Cash. Because two-sided untraceability is not prevented, this system is open to criminal uses such as extortion, bribery and money laundering.

Recently, CyberCash has launched a payment system called CyberCoin, also for on-line payments over the Internet. This system has been designed to have functionality similar to the DigiCash system, but lacks provisions for anonymity. According to public statements from CyberCash, value resides at all times within the bank and the system actually is an instruction-based system.