Making Electronic Refunds Safer

Rafael Hirschfeld

Laboratory For Computer Science
Massachusetts Institute of Technology
Cambridge, MA 02139

Abstract. We show how to break an electronic cash protocol due to van Antwerpen (a refinement of the system proposed by Chaum, Fiat, and Naor), and give an alternative protocol that fixes the problem.

1 Introduction

There has been much recent interest in electronic money—ways to perform monetary transactions by computer, telephone, fax machine, etc. Most proposed electronic money schemes rely on cryptography for their security, in particular, on digital signatures [7].

In response to privacy concerns, electronic money systems in which payments are untraceable without the cooperation of the payer have been developed. This untraceable electronic money is called **electronic cash**. The cryptographic mechanism used to provide untraceability is that of blind signatures [3].

Several different forms of electronic cash have been proposed. In addition to electronic coins [4], which have a fixed value, there are electronic checks [4] [2], which can be used for any amount up to a maximum value and then returned for a refund of the unused portion, and divisible electronic cash [5], which can be broken into smaller pieces that can be spent separately.

We concentrate on a recent electronic check scheme [1] that is based upon earlier check schemes, but with great improvement in efficiency. We show how a weakness in the refund mechanism of the earlier systems becomes a fatal flaw in the newer system, and how that flaw can be exploited to cheat undetectably. We propose a revised protocol to correct the flaw.

2 Transactions

We call the participants in an electronic cash system the **bank**, the **user**, and the **shop**. The bank is the issuer of the electronic cash. The user obtains electronic cash from the bank in a **withdrawal** transaction, spends it at a shop in a **payment** transaction, and the shop then redeems it at the bank in a **deposit** transaction. In addition, for electronic check systems, the unused portion of a check is returned by the user to the bank in a **refund** transaction.

We distinguish the user and the shop only to emphasize their roles in a payment; in fact, users can act as shops and shops can act as users. The user and the shop are also called the **payer** and the **payee**, respectively.

3 Checks

Untraceable electronic checks were introduced by Chaum, Fiat and Naor [4], and refined by den Boer, Chaum, van Heyst, Mjølsnes, and Steenbeek [2], with a significant improvement in efficiency. Checks are made up of three kinds of elements: challenge terms, denomination terms, and refund terms.

Challenge terms are used to prevent double-spending of a check. Each challenge term c_i contains a random a_i chosen by the user, as well as $a_i \oplus u$, where u is the user's identity (bank account number). During payment, each challenge term is opened to reveal either a_i or $a_i \oplus u$, depending on the corresponding bit of a challenge chosen by the shop. If a check is spent with two different challenges, then (for some i) both a_i and $a_i \oplus u$ will be revealed, from which u can be obtained.

Denomination terms are used to represent the value of the check. Each denomination term d_i contains a unique coin number b_i, and corresponds to a different power-of-two denomination. The denomination terms are either ordered or are signed with different roots to indicate which denomination they represent. During payment, only those terms corresponding to denominations used are actually opened. In order to keep the user from mixing terms from different checks (which would expose the protocol to a simple attack), the challenge and denomination terms are tied to a check number. The shop (and the bank) can easily verify that all the terms presented for payment (or deposit) belong to the same check.

Refund terms are used to obtain refunds for unspent denomination terms in a check. Unlike the other terms, they are not tied to the check number. If they were, the bank could link deposits to their corresponding refunds, defeating the untraceability of the scheme. Instead, the refund terms r_i contain the same coin numbers b_i as the denomination terms. The bank keeps track of spent and refunded coin numbers to ensure that a denomination is not both spent and refunded.

The coin numbers and identity numbers are built into the terms by the user. Because the terms are blinded for untraceability, the bank doesn't actually see these numbers at the time of withdrawal. To ensure that the terms are correctly formed, the protocol uses the cut-and-choose methodology introduced by Rabin [6]: the bank asks for more candidate terms than it actually needs, chooses a random subset of them and asks the user to demonstrate that they are well-formed, and uses the remaining unopened candidates only if all of the opened candidates were legitimate. We will assume that the bank asks for twice as many candidates as it needs, so that the probability of the user getting caught attempting to slip in a single bogus term is 1/2.

This is the source of the weakness in these check systems. With probability 1/2, the user can slip a bogus candidate past the bank. If there are enough challenge terms, a single bogus challenge term is unlikely to be of much use to a cheater, because the probability is still overwhelming that two challenges will differ at some other position. A user who tries to slip enough bad challenge terms into a check to cheat effectively will with high probability be caught.

For the other terms, though, this presents a problem. If the user can slip in a denomination term/refund term pair for which the coin numbers b_i don't match, then that denomination can be both spent and refunded, undetectably.

To address this problem, Chaum, Fiat, and Naor suggest penalizing detected cheating attempts so that the net expected effect favors the bank. This solution is somewhat unsatisfying. At the time of the detection—withdrawal—the user has not yet cheated (by spending and refunding the same term). When a bad term is detected, the user may claim it is due to a data error, or may even refuse to open a bad term, claiming that the data has been lost. Since data errors and losses do occur, it would be difficult for the bank to fully justify imposing the penalty.

Another possibility is to require more than one pair of terms for each denomination; Chaum, Fiat, and Naor suggest using two, as an alternative to imposing penalties upon detection. This would lower the probability of cheating because in order to escape detection, both pairs would have to be bad. The more pairs make up a denomination, the lower the chance of successfully cheating, but the bigger and costlier to handle the check becomes.

As we will see, this troublesome problem proves to be fatal in the latest incarnation of the system.

4 "Improved" Electronic Cash

A new protocol by van Antwerpen [1] further refines this electronic cash system. This is a sophisticated protocol with order-of-magnitude improvements in both the size of the checks (and consequently the amount of storage required for them) and the amount of communication required, at the expense of some extra computation that can be done in the background. For efficiency reasons, several checks are grouped into a single pack; during withdrawal the user obtains an entire pack from the bank, but then spends them one at a time.

For a complete description of the protocol, the reader is referred to van Antwerpen's paper, but we will give a brief synopsis. A pack of k checks is made up of $2k$ *pseudochecks* and $2k$ *pseudo-refund-parts*, each of which is a single RSA-sized number. The terms that make up each pseudocheck (pseudo-refund-part) are multiplied together in such a way that they can later be separated. The pseudochecks contain the denomination and challenge terms, and the pseudo-refund-parts contain the refund terms. The terms that make up an *actual* check (refund part) are distributed among the pseudochecks (pseudo-refund-parts) by permutations chosen by the bank. The refund terms in the pseudo-refund-parts are also permuted by the user so that they won't line up with the corresponding denomination terms in the pseudochecks. This is to prevent the bank from gleaning information that might be used to link by by amount, *i.e.*, to link a deposit to a corresponding refund by checking that they are for complementary amounts.

In addition, the cut-and-choose process has been modified. The bank still chooses a random subset of the terms, and the user still provides opening information for those terms, but the bank, rather than verifying that they are correct (which would be difficult because of the way the pseudochecks are formed), instead multiplies the resulting pseudocheck by a *protection factor* that renders it useless if the user lied about any of the opened terms. Veugen [8] has proved that the security of this technique is equal to that of the original, in the sense that if the user attempts to cheat, the probability that she will be unable to use the resulting checks is the same as the probability that she would be caught by the original cut-and-choose process.

But this is precisely where the new protocol is flawed. Although there is still no problem with the challenge terms, recall that the solution (of imposing penalties) to the problem with the refund terms relied upon detection by the bank of cheating attempts. With this new mechanism, cheating attempts are *never* detected.

Let b_i be the coin numbers in the denomination terms and b'_i be the coin numbers in the refund terms. The user could make $b_i \neq b'_i$ for *all* i. Then when asked to open some terms, the user provides the b_i, so that she will be able to remove the protection factors. The opened terms are divided out of the check, but the user will be able to both spend *and* refund all of the denominations that actually make up the check, undetectably. Needless to say, this is not a desirable property of an electronic cash system.

5 Attempted Fixes

There are several ways that one might attempt to patch the protocol. One possibility is to put protection factors on the pseudo-refund-parts in addition to the ones on the pseudochecks. That way if the user were asked to open the ith terms, where $b_i \neq b'_i$, then if she produced b_i she wouldn't be able to refund the term, and if she produced b'_i she wouldn't be able to spend it. So in the previous scenario (with $b_i \neq b'_i$ for all i), she would still be able to spend all of the denomination terms, but not be able to refund any of the refund terms, and so would gain nothing.

But this isn't good enough. The user need not make all of the denomination/refund pairs bad. If only some are bad, it is possible that the bank won't select any of them during the cut-and-choose. The fewer bad terms there are, the more likely that they will all slip by. Whenever the bank picks a bad term to be opened, the user won't gain anything, because she'll only be able to spend the check and not refund any of it. But she won't lose anything either. Since the attempts are undetectable, she can just keep trying until she succeeds.

To work around this, we could add an extra refund term that has no corresponding denomination term, so that the user must refund it or else lose money. But this isn't quite good enough either, because the user could reveal the coin numbers b'_i from the refund terms instead of the coin numbers b_i from the denomination terms, and when "caught," she would just refund the whole check and not spend any of it. To prevent this, we would need also to add an extra denomination term that has no corresponding refund term, forcing the user to spend as well as to refund. This works after a fashion, provided that the values of these extra terms are large enough to ensure negative expectation from cheating attempts, but it is cumbersome and makes the checks inconvenient to use. We will present a much cleaner solution.

6 A Modified Protocol

Our problems stem from the difficulty of ensuring that two coin numbers that are provided separately are in fact the same. The solution is actually quite simple—just provide a single coin number and use it for both terms! With the earlier systems, it is difficult to see how to do this without compromising the unlinkability of the terms, but in the newer system it is fairly straightforward.

In van Antwerpen's system, candidates provided by the user to the bank during withdrawal look like

$$X = R^{PQ} \prod_i F(a_i)^{P_i Q} \prod_i G(b_i)^{PQ_i} \tag{1}$$

$$Y = T^r S^Q \prod_i G(b_i)^{Q_i} \tag{2}$$

where X is a pseudocheck candidate, Y is a pseudo-refund-part candidate, R, S, and T are blinding factors, the P's and Q's are exponents related to the prime roots used for signatures, the F's are challenge terms with random numbers a_i (as well as $a_i \oplus u$) incorporated, and the G's are denomination or refund terms, with the random coin numbers b_i incorporated. There are multiple X and Y for a given check pack; not shown here is a collection of permutations θ_i chosen by the user, the ith refund term of the jth pseudo-refund-part corresponds to the ith denomination term of not the j'th pseudocheck, but rather the $\theta_i(j)$th pseudocheck. These permutations are to prevent the bank from linking by amount, as previously mentioned.

For now we will ignore these permutations as well as the blinding factors. Note that the basic form of X and Y are

$$X = \prod_i F(a_i)^{P_i Q} \prod_i G(b_i)^{PQ_i} \tag{3}$$

$$Y = \prod_i G(b_i)^{Q_i} \tag{4}$$

The b_i are given in both X and Y, and it is difficult to ensure that they in fact match. But instead of giving X and Y to the bank, the user can instead give

$$X_1 = \prod_i F(a_i)^{P_i} \tag{5}$$

$$X_2 = \prod_i G(b_i)^{Q_i} \tag{6}$$

and now the bank can compute

$$X = X_1^Q X_2^P \tag{7}$$
$$Y = X_2 \tag{8}$$

and *voila!*, the denomination and refund terms are now *guaranteed* to contain the same coin numbers.

This is the primary protocol change. Except for the refund transaction, the rest of the protocol is the same as before.

We ignored the blinding factors and the user permutations. The blinding factors are not a problem; we could just put blinding factors as before on X_1 and X_2. The resulting X would have a different blinding factor from the original protocol because it would be the product of the two blinding factors, but that has no detrimental effects on either the privacy or the security of the system.

The permutations are another matter. Because the denomination and refund terms are now provided together as a single unit, there is no way to put them in separate places as before. This may not be as serious as it seems, because we need not refund the pseudo-refund-parts in the same order in which we withdrew them, so linking by amount can be made very difficult. Still, it would be preferable for the new protocol to leak no more information than the old. In the next section we will show how to recapture the full degree of unlinkability by slightly altering the structure of the blinding factors.

7 Regaining Lost Privacy

Notice that the pseudo-refund-part candidate has *two* blinding factors, T^r and S^Q. The blinding factor T^r is used to blind during withdrawal (r is the prime root used to sign the pseudo-refund-part, so the user can divide by T afterwards). The blinding factor S^Q is used during refund; terms that have been spent are moved into this blinding factor so that they are not revealed when refunding the unspent terms.

If the user could separate out and refund each unspent term individually, she could rearrange them into any order she wanted, getting the same privacy effect as from the permutations θ_i. Although she could do this just by moving unspent terms along with spent terms into the blinding factor, she would have to do it once for each unspent term in the pseudo-refund-part, and the resulting blinding factors would be correlated because they would contain some of the same factors. The bank could use this fact to link terms that come from the same pseudo-refund-part, undoing the privacy gain we thought we had achieved.

If we change the refund blinding factor, however, to be of the form S^{Qr}, then we can make this work. After the bank takes the rth root, the user is left with S^Q rather than $S^{Q/r}$, which means that she can change S. She couldn't do this before because she couldn't extract rth roots. By changing the S for each separated unspent refund term, the user can make them unlinkable from each other.

By making this change to the blinding factors we can recapture all of the privacy lost by eliminating the permutations θ_i. The rest of the protocol is also simplified by the elimination of the permutations. The cost is in efficiency: each refund term from the same pseudo-refund-part must be separated and refunded separately. But the blowup is only by the average number of unspent denominations per pseudocheck, and can be traded off against a slight increase in traceability if desired.

Acknowledgements

I would like to thank David Chaum for his guidance and support for this work, and Thijs Veugen for his assistance in working out the method for regaining privacy.

References

1. C. J. van Antwerpen, "Electronic cash," master's thesis, Eindhoven University of Technology (1990).
2. B. den Boer, D. Chaum, E. van Heyst, S. Mjølsnes, and A. Steenbeek, "Efficient offline electronic checks," Proceedings of Eurocrypt '89, 294–301.
3. D. Chaum, "Blind signatures for untraceable payments," Proceedings of Crypto '82, 199–203.
4. D. Chaum, A. Fiat, and M. Naor, "Untraceable electronic cash," Proceedings of Crypto '88, 319–327.
5. T. Okamoto and K. Ohta, "Universal electronic cash," Proceedings of Crypto '91.
6. M. O. Rabin, "Digitalized signatures," Foundations of Secure Computation, Academic Press, NY (1978).
7. R. Rivest, A. Shamir, and L. Adleman, "A method for obtaining digital signatures and public-key cryptosystems," CACM 21, 2 (February 1978).
8. T. Veugen, "Some mathematical and computational aspects of electronic cash," master's thesis, Eindhoven University of Technology (1991).