How to Break Another "Provably Secure" Payment System

Birgit Pfitzmann, Matthias Schunter

Universität Hildesheim
Institut für Informatik
Marienburger Platz 22
D-31141 Hildesheim, Germany

✆ ++49-5121-883-{738, 788}
Fax ++49-5121-883-732
{pfitzb, schunter}@informatik.uni-hildesheim.de

Michael Waidner*

Institut für Rechnerentwurf und Fehlertoleranz, Universität Karlsruhe
Zirkel 2
D-76128 Karlsruhe, Germany

✆ (Zurich) ++41-1-724-8220
Fax ++41-1-710-3608
wmi@zurich.ibm.com

* on leave to IBM Zurich Research Laboratory, Rüschlikon.

Abstract: At Eurocrypt '94, Stefano D'Amiano and Giovanni Di Crescenzo presented a protocol for untraceable electronic cash based on non-interactive zero-knowledge proofs of knowledge with preprocessing. It was supposed to be provably secure given this and a few other general cryptographic tools.

We show that this protocol nevertheless does not provide any untraceability and has some further weaknesses. We also break another "provably secure" system proposed by Di Crescenzo at CIAC 94.

This is the second case of problems with "provably secure" payment systems. Moreover, yet another system with this name tacitly solves a much weaker problem than the seminal paper by Chaum, Fiat, and Naor and most other "practical" papers in this field (de Santis and Persiano, STACS 92). We therefore identify some principal problems with definitions and proofs of such schemes, and sketch better ways to handle them.

1 Introduction

Untraceable electronic cash seems to be a field where schemes called "provably secure" are easier to break than those constructed ad-hoc — this is the second such case after [PfWa_92] broke and repaired a system from [Damg_90].[1]

In the cryptologic community, electronic cash means digital payment systems, nowadays offline, that do not fully rely on the tamper-resistance of devices, and that provide some privacy to their users. It is sometimes said that electronic cash is just like paper cash, only more secure and transferable over networks. We do *not* say so, and have experienced that it is a very confusing thing to say — people associate quite a lot of different properties with cash, and electronic cash fulfils some of them, but others not, and has some new unexpected properties.[2] Moreover, electronic cash systems differ a lot among themselves.

1 To avoid misunderstandings from the start: Any possible sideswipes are certainly not directed at proving security in cryptography. Quite the contrary, we would like to advocate that at least all the systems called provably secure are indeed proved.

For instance, before [ChFN_90], the term electronic cash was even used for online payment systems, since no offline ones (at least in the model where one does not fully rely on the tamper-resistance of devices) were known and it was widely believed that the known systems were as close as one could get to simulating paper cash.

Hence it should always be said *what* electronic cash one is talking about. The current style of expressing this is, in our opinion, one reason for the problems with "provably secure" electronic cash systems. Hence, in Section 2, we just give a brief sketch of the definition against which the systems we break are *claimed* to be secure. We already mention that this is a very weak definition. In Section 3, we describe the system from [AmCr_94]. In Section 4, we show that it does not even fulfil the weak definition. Actually, we think that any similar use of non-interactive zero-knowledge proofs with preprocessing will have such a problem. In Section 5, we describe and break the system from [Cres_94]. This time, we think the claimed restriction of the round complexity cannot be achieved at all, except under a very unreasonable additional assumption.

In Section 6, we give some recommendations. Mostly, these are hints as to what can be done to avoid "provably secure" systems being broken in the future, and to make clearer what properties individual systems are even *trying* to achieve. In addition, we explain some more properties that electronic cash can and should have in practice and that are usually omitted in theoretical papers like the ones considered here or [SaPe_92, FrYu_93].

Some History

It is not necessary to know much about the history of electronic cash for the purpose of this paper. For completeness, however, note that electronic payment systems that *do* fully rely on tamper-resistance usually follow the description in [EvGY_83], whereas those that *only* rely on cryptology for security started with [Chau_83, Chau_85] and similar papers; [Chau_89] is the most comprehensive one. The latter were all on-line systems, i.e., they need on-line verification of payments. The first off-line electronic cash system was presented in [ChFN_90]. At the same time, [Damg_90] presented the first electronic cash system claimed to be provably secure under a well-known cryptologic assumption. This was still an on-line system, whereas the corrected version in [PfWa_92] also has an extension to off-line payments, combining ideas of [Damg_90, ChFN_90]. The newest efficient, not provably secure systems are [Bran_94, Ferg_94]. Further provably secure systems were presented in [FrYu_93, SaPe_92]. The latter solves a weaker problem than the other papers mentioned and relies on non-interactive zero-knowledge proofs in the shared random-string model. The system in [AmCr_94] can be seen as an adaptation of that scheme to the other model for non-interactive zero-knowledge proofs, the model with preprocessing, and to multiple transfers of the "same coin" and divisibility of "coins" into smaller pieces.

---

[2] Just two examples: You cannot simply undo a payment by giving a "coin" back — you can solve some disputes about whether a payment was made by asking the payer to pay the same "coin" to the payee again.

2 The Weak Definition

2.1 Definition Sketch

The electronic payment systems in [AmCr_94, Cres_94], like most others in cryptology since [ChFN_90], are so-called coin systems. Their most important parts are three protocols:

- **Withdrawal**, where a user obtains information called "electronic coin" from a bank in exchange for real money (usually deducted from the user's bank account). In all existing constructions, an electronic coin somehow contains a string called "coin number" and a signature by the bank on the coin number with a special key, such that any signature with this key is interpreted as "the signed string is a coin of value so-and-so". (Thus there are different keys for different denominations.)
- **Payment**, where one user passes an "electronic coin" to another user.
- **Deposit**, where a user exchanges an "electronic coin" for real money again at a bank.

In addition, there is a protocol for establishing a bank account and a start-up protocol for the bank. In real life, several further protocols are needed, see Section 6.

The requirements made in [AmCr_94] can be stated as follows. (This description is more informal, but more correct than the original one. Some general problems with requirements at this level of abstraction are mentioned in Section 6.)

- **Unforgeability**: With k withdrawals, it should be infeasible for the users to compute k+1 different coins.
- **Untraceability**: After observing k withdrawals and k deposits, the bank should not be able to find out which deposit corresponds to which withdrawal, and thus who paid to whom. (Hence the "electronic coin" must change its outlook between the withdrawal and the deposit — this is where interesting cryptology, such as privacy homomorphisms, comes in.)
- **No double-spending**: If a user spends the same coin twice, she will be identified as soon as both payees deposit their copies at the bank.
- **No framing**: Nobody can wrongly be blamed as a double-spender.

Extensions to the basic scheme in [AmCr_94] are supposed to offer the following two properties:

- **Transferability**: "Coins" received in a payment cannot only be deposited at the bank, but also forwarded to other users.
- **Divisibility**: "Coins" can be divided into smaller pieces that can be spent individually.

2.2 Weakness of the Definition of Untraceability

The degree of untraceability required in [AmCr_94] and sketched above is very low: There is no untraceability against *payees*, since the adversary is assumed to have observed withdrawals and deposits only. Usually, the adversary against untraceability is assumed to have observed payments, too; see [ChFN_90, FrYu_93]. This deviation from the usual definition cannot be just a typo in [AmCr_94], since the protocol in [AmCr_94] for identifying double-spenders *depends* on the ability of each payee to identify the corresponding payer.

The lower level of untraceability is unacceptable for several reasons:

1. Protocols providing unconditional untraceability in the usual sense, i.e., against all coalitions of bank *and* payees, exist already, e.g., [ChFN_90, FrYu_93, Bran_94, Ferg_94].
2. In practice, untraceability by the payee is often just as important as untraceability by the bank. For instance, people will not want a public transport company to be able to record all their bus rides any more than a bank (In particular, since this company would know the times and places, whereas the bank alone would not.)
3. The same low level of untraceability can be achieved much more easily than described in [AmCr_94]. We sketch this in Section 4.5, after the original protocol.

In the following sections, however, we show that the systems do not even provide this weak level of untraceability.

3 The System Presented at Eurocrypt 94

Initialization

For initialization, the bank in [AmCr_94] publishes some system parameters and public keys, especially a public key of an arbitrary signature scheme and a string commitment scheme.

Each user publishes a public key of a signature scheme, too. Additionally, each user performs the preprocessing phase of a non-interactive zero-knowledge proof (NIZKP) system of knowledge with preprocessing, with the user as the prover and the bank as the verifier.

It is important (although not stated in [AmCr_94]) that the bank's result, β, of this preprocessing must be published, too: If a payee receives a coin, he needs this information for verifying that the coin is valid. More precisely, he needs β as an additional input to the protocol called V_2.

Withdrawal

If a payer wants to withdraw a coin, she chooses a coin identifier, c, randomly, computes a commitment, *com*, of c, and sends it to the bank. The bank sends a signature on *com* back to the payer.

Based on the information used to compute *com*, the signature received from the bank, and the information generated in the preprocessing phase of the NIZKP system, the payer computes a non-interactive zero-knowledge proof, *cp*, that will convince a payee that the payer knows:

- a commitment on c and

- a valid signature by the bank on this commitment.

Payment

The payer forwards c and cp to the payee. The payee verifies the proof cp, using the result, β, of the bank during the preprocessing phase with the payer.[3]

Additionally, the payer signs the pair (ID_{Payee}, c) and forwards this to the payee. This signature is indeed needed by the payee: If c is spent twice, the payee is asked to identify the party that paid c to him.

The payee can reuse the coin or deposit it. Deposit is simply a payment to the bank. (Note that all payments after the first one do not involve *computing* a proof of knowledge, but passing the received one on, since only the first payer has the secret inputs needed to compute such a proof.)

Double-Spending Detection

If the bank receives the same coin in two deposits (i.e., the strings c of the two coins are equal, the strings cp may or may not be), say from users A and B, it determines all previous owners of c, based on the signatures forwarded in payments. In [AmCr_94], this is done by broadcasting a request to all users; we do it more practically below. At the end, either

1. an owner X is detected who does not cooperate, i.e., does not reveal the source of c,
2. an owner X is detected who has paid the coin twice, or
3. two withdrawals of the same c are detected.

In the first two cases, X is assumed to be an attacker. This is clearly not necessarily true in Case 1: X could simply be unreachable or have lost the necessary information due to a technical defect. I.e., this approach works in a technical sense, but would often yield incorrect results in practice.

Case 2 poses technical problems, see Section 4.3.

Case 3 is not considered in [AmCr_94], but it is no problem: In each of the withdrawals, the appropriate sum was deducted from an account, hence there is no fraud at all.

It should be mentioned that the schemes for double-spending detection based on secret sharing [FrYu_93, Bran_94, Ferg_94] can be applied to *transferable* schemes, too [Antw_90, ChPe_93], i.e., feasible solutions to this problem exist already.

4 Weaknesses of this System

4.1 The Main Untraceability Flaw

Unfortunately, the protocol in [AmCr_94] does not provide any untraceability, not even

[3] The protocol in [AmCr_94] is ambiguous about whether c must be passed or only cp, but it must: There is no guarantee that one cannot construct two different NIZKPs on the same c, hence c is needed for the uniqueness of coins. Thus, in [AmCr_94 p.156], T should be the statement that a commitment and signature on the given coin identifier c are known, and c should be an input to V_2, too.

the weak version required there:

During the deposit of a coin (c, cp), the bank must verify that cp is a valid NIZKP. This verification depends on the preprocessing phase of the user X who originally withdrew the coin; i.e., just like the payee in a payment, the bank must know which string β to use in the verification protocol V_2. It therefore knows the identity of the payer during the preprocessing with whom this particular β was generated, i.e., it has traced this payer.

This is a general problem with NIZKP with preprocessing: At least as long as preprocessing is a 2-party protocol, any reference to it, i.e., any proof, identifies the two parties. Thus at least the real untraceability against payee and bank seems impossible to achieve by straightforward use of this primitive. If preprocessing were performed anonymously before opening an account, all payments of one payer could still be linked, which would lead to identification after a while. The only remedy would be to execute anonymous preprocessing once for each coin to be withdrawn, where all preprocessings have to be unlinkable to the withdrawals. This would be very inconvenient for the users, and the immediate distribution of all the resulting strings β to all payees would contradict the offline properties of the cash system.

4.2 An Untraceability Flaw in Double-Spending Detection

Even without the flaw due to individual preprocessing, the user's privacy is not really protected: Until the double-spender is detected, the bank cannot *prove* that a coin was double-spent, i.e., the users have to believe this. Thus, a dishonest bank could always claim that c was deposited twice in order to get the payees to disclose the payers.

The general rule with untraceable protocols should be that a proof of wrong-doing must be shown before any sensitive information is asked to be disclosed, if at all.

4.3 Finding the Wrong Double-Spender

First, it is wrong to call any user a double-spender who has spent the same coin twice: She may have received the coin twice, e.g., from an attacker at different times. In this case, the honest user is likely to be found before the attacker and punished as the double-spender.

A first refinement, which avoids this problem and the broadcast channel, works as follows: The bank recursively asks each owner X of c for the identity of the user who paid c to X, starting with $X = A, B$. At the end, Case 1 and 3 are as above, and Case 2 is modified to

2. an owner X is detected who has paid the coin twice and cannot show that she also received it twice.

However, there remains a problem, since the signatures are only on the coin and the identity of the payee. Consider the graph of payments with one coin in Figure 1.

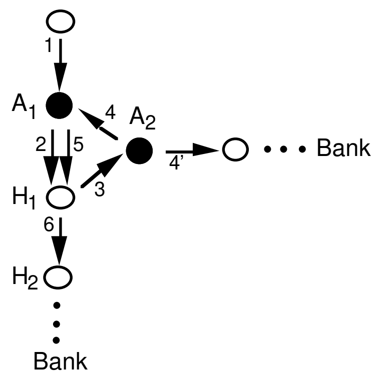

Fig. 1: Payment graph with one coin for an attack on the double-spending detection.

A_1 and A_2 are attackers, the others are honest users. A_2 is the double-spender. However, when the bank traces the coin back, A_2 and H_2 both show they got the coin from H_1. H_1 has two completely identical signatures from A_1, so she cannot prove she got the coin twice. Thus H_1 is punished, and A_1 never has to reveal that he also got the coin from A_2, which leaves A_2 without risk of detection.

To prevent this attack, passing one coin twice to the same payee must yield different signatures, and the payee must be able to verify this. This can be done interactively with a random challenge or a transaction number from the payee. Non-interactively, one can use time-stamps if synchronized clocks are available. Otherwise, the payer uses transaction numbers and the payee must compare a newly received coin with all previously received ones. This cannot introduce a new untraceability problem, since this scheme has no untraceability in this respect anyway.

4.4 No Divisibility Together with Transferability

The divisibility scheme [AmCr_94, Sect. 5] may be correct or not — at least for transferable coins it does not make much sense: Divisibility is based on a predefined tree structure in each coin, and all important complexity parameters (communication during withdrawal and payment and storage space for a coin) are linear in the number of nodes in this tree. Thus, this scheme has no real advantage over using coins of the smallest denomination, such as centimes, only.

To see this complexity, note that a completely separate signature by the bank on a value d_l at each node, l, of the tree must be transferred or stored, respectively, in all these situations. (If the coin is not transferable, only the signature on the root of the transferred part of the tree has to be transferred.)

Moreover, the divisibility scheme *requires* bit-by-bit commitments, whereas much more efficient string commitment schemes exist [Naor_91]. Hence, independent of developments in the computational complexity of the general cryptographic tools, it will remain inefficient.

4.5 Achieving Weak Untraceability More Easily

As announced in Section 2.2, we show that the weak untraceability required in [AmCr_94] can be achieved more easily than with that protocol.

Since the payee knows the payer anyway, no NIZKP is needed for proving the validity of a coin to a payee. Instead, the payer can forward all information obtained during withdrawal (i.e., the chosen coin identifier, c, the string, r, used to commit to c, and the bank's signature on the commitment) to the payee, who can easily and noninteractively verify it. During deposit, the payee proves knowledge of a signature on a commitment to the coin identifier, using any appropriate zero-knowledge proof scheme (not necessarily noninteractive now!). This would also remove the privacy flaw exhibited above.

5 The System Presented at CIAC 94

There is no space to describe the system from [Cres_94] completely. However, the problems can be understood without many details. We first describe a general problem with the requirements and assumptions, and then show that even those requirements are not fulfilled.

5.1 Extreme Non-Interactiveness and the Consequences

The main purpose of [Cres_94] is to make all transactions completely non-interactive — even a withdrawal consists of just one message, from the user to the bank (a broadcast message to everybody, so that they can later recognize the coin). Now, however, the bank cannot announce if it rejects the withdrawal since no money is left on the user's bank account. Hence either the bank has to accept all withdrawal wishes, or payees will accept the coin but not be able to deposit it. Although this does not contradict the requirements made, it must clearly be seen as an error.

Here we more or less agree with [SaPe_92] who claimed that their 2-message withdrawal was optimal. The only alternative seems to be that everybody mirrors the bank account of everybody else, and all transactions are carried out via a broadcast channel, so that everybody can decide locally if a withdrawal is possible or not. This is not too far from the protocol in [Cres_94], where broadcast channels are also used extensively, but completely unrealistic in practice.

Thus for any further consideration of that system, one has to give up this notion of non-interactiveness and assume that the bank broadcasts if it accepts or rejects a withdrawal.

5.2 No Untraceability

Very obviously, there is no untraceability at all in the scheme, since each coin is identified by a string $(g_1, \ldots, g_n)$ that was broadcast during withdrawal and is transferred in clear in each transaction with this coin.

5.3 Cryptographic Security Problems

For spending a coin, a kind of signature scheme is used, where a tuple of numbers (w_1, …, w_n) is signed modulo a Blum integer x_u by computing the square roots r_i of exactly those values w_i that are quadratic residues. It is said that such a signature can easily be checked by a recipient. However, deciding quadratic residuosity modulo Blum integers is assumed to be hard. As an extreme consequence, someone can forge a signature by claiming that none of the w_i's was a quadratic residue.

Moreover, as is well-known from Rabin's scheme, such a scheme can be broken if a chosen-message attack is possible. This is not the case for the original payer in [Cres_94]. However, the payee also has to use this scheme when he deposits the coin. Now w_i is constructed as $r_i d_i g_i$ for certain fixed values r_i and d_i, but with g_i chosen last and by the payer. The payer is restricted in this choice, but this is only verified by the bank. Thus a collusion of a bank and one payer can perform the chosen-message attack and compute the secret key of a payee.

Furthermore, the system starts with the bank choosing several random strings, which should be obtained from the source for shared random strings instead.

6 Recommendations

We do not try to repair the schemes considered above — in both cases we argued that major changes would be needed. We do not doubt that general cryptologic tools are powerful enough to construct many kinds of electronic payment systems. The question that interests us more is how one can avoid that further payment systems called "provably secure" with several, partly obvious, errors are published at refereed conferences. Considering this and experiences that many people both inside and outside cryptology have no idea what exactly all these papers about cryptologic electronic payment systems are trying to achieve, we think that the part of cryptology that deals with large protocols has rather a poor standard of definitions for an exact (and security-related) science. (Of course, the proof sketches in [AmCr_94, Cres_94] were far too short, too.) We see three types of problems with the current definitions of payment systems:

1. There is no correct definition yet.[4]
2. The definitions are incomplete, i.e., properties that would be needed in practice and that can be achieved, are not mentioned and thus often forgotten in constructions. For instance, if one does not trust the bank with respect to framing, one should also require that the bank cannot falsely claim that all the money was already withdrawn from an account, and that a payee can prove that he deposited money if it later fails to turn up on his statement of account. Moreover, it may be required that the bank can only deposit money to the particular payee that the payer intended to pay it to, to say nothing about receipts for payments.
3. Far fewer names for properties are in use than different properties, and thus things are often considered equal that are not. This was seen above in the use of "untraceability" for a far weaker property than in other papers without ever saying so. Actually, the same weak definition is already used in [SaPe_92]. Similarly, it is hard to find in

[OkOh_91] and not at all in [OkOh_92] that untraceability there is weak in a different sense: all payments by one customer can be linked, whereas usually, they are all independent.

We do not claim to have a complete definition at present (actually, if we claimed to have one that fitted on two pages you should be very suspicious), but we can give some hints as to what such a definition should look like.

a) Short statements of cryptographic properties (formal or informal) should always come with an explicit *trust model*, i.e., *for* whom a property is guaranteed, and which *other* participants have to be trusted to guarantee this. For instance, untraceability is usually *for* payers (but some schemes by Chaum offer untraceability for payees instead), and nobody else is trusted, but [AmCr_94] requires trust in all payees.

b) A top-level specification is needed where the *properties* required can be expressed independently of the trust model or even the details of the protocols to be proved, but still in a way that can later be combined with a trust model in a formally precise way.[5]
   - For complete protocols, this can mean a specification by an "ideal" protocol performed by a centralized trusted host, together with a definition what it means for a "real", decentralized protocol with untrusted participants to simulate the ideal protocol. This would extend the similar and still unfinished definitions for multi-party function evaluation (cf. [Yao_82, Beav_91, MiRo_91] for the oldest and newest versions).
   - For individual properties, at least integrity properties, it can mean a specification in any usual formal specification language, if one defines two things: (1) Once and for all, what it means for such a specification to be fulfilled in a cryptologic sense, that is, under a trust model (i.e., with certain participants deviating from their protocols and colluding), and often with error probabilities or in a complexity theo-

---

[4] To justify this briefly, we just look at what is both in public and our own opinion the best one, in [FrYu_93], although we know that we are now being pedantic and that it is good that such a definition sketch exists at all.

First, it is only semi-formal (no probability spaces defined), and most active attacks are not mentioned.

Secondly, it is really not quite correct: The property called "unforgeable" here and "unexpandable" in [FrYu_93] is unfulfillable in any similar form: It says that from the views of customers of n withdrawal protocols, it should be infeasible to compute $n + 1$ distinct coins that will lead to successful purchase protocols. Both the algorithms that compute and spend the "coins" must be adversaries here; hence the term "coin" cannot be defined as "correct coin", but any tape content that will lead to a successful purchase. Now, for any secure system, one can construct a stupid adversary that extends coins by random bits at the end, but still spends them according to their original part. Thus it can formally spend many "distinct" coins. To express what was really meant, this property and that of double-spending detection must be defined in one piece, roughly saying that more successful purchases than withdrawals lead to identification.

Thirdly, some standard requirements are missing, e.g., "no framing" from above, or that the bank should not be forced to carry out a withdrawal if the user has no real money. (The latter can trivially be fulfilled with the protocols in [FrYu_93], but we saw that the protocol in [Cres_94] does *not* fulfil it.)

[5] Just to prevent misunderstanding: We do not mean "logics of authentication", which people outside cryptology use as top-level specifications for lack of such things from inside cryptology, although it seems clear that monotonic logics cannot possibly fulfil what most people expect of them, and although the formal semantics of those logics makes assumptions about cryptologic primitives that the definitions of these primitives do not justify.

retic sense only. (2) What aspects of the protocols considered are visible to the outside world ("interface behaviour"), i.e., are mentioned in the specification. E.g., the problem with the notion of "coin" when held by an adversary (Footnote 4) disappears if one specifies in terms of inputs that start transactions, e.g., "withdraw, *amount*", and outputs denoting the result of such transactions (such as "deposit accepted") alone (cf. [Pfit_93] for such a definition of the (much simpler) properties of signature schemes).

c) Properties that deal with provability can only be defined by reference to a verification protocol. For instance:
   - "No framing" needs a protocol **prove_framing** between a bank, a third party, and possibly the accused payer. The requirement of the bank is roughly that it can convince any honest third party that someone was a double-spender if double-spending occurred; the requirement of each payer P is that no honest third party will be convinced of P double-spending if P follows her protocols.[6]
   - The fact that the bank cannot deduct more real money from a bank account than the electronic money the payer requested and received must be defined with a protocol **audit_withdrawals** between the bank, a third party, and the payer. In constructions, the bank may show signed withdrawal requests to the third party (which must be produced and stored in the withdrawals), and if the payer claims she did not receive the corresponding coins, it must basically be possible to send them again.

Acknowledgment

We thank *Stig Mjølsnes* for a discussion at Eurocrypt 94 where we already agreed that a cash system requiring occasional broadcast and cooperation of all honest users was not what one would expect, *Joachim Biskup* for helpful comments on this paper, and *Mihir Bellare* and *Jens Krüger* for interesting discussions about requirements on payment systems. Furthermore, we would like to thank the anonymous reviewer who pointed out that some attacks on the system (or, according to Giovanni di Crescenzo, the weakness of the definition mentioned in Section 2.2) have been described in a letter by Jacques Traore. Unfortunately, we have not been able to contact him or to obtain a copy of his letter.

References

AmCr_94 Stefano D'Amiano, Giovanni Di Crescenzo: Methodology for Digital-Money Based on General Cryptographic Tools; Pre-proceeding of Eurocrypt '94, May 9-12, 1994, University of Perugia, Italy 151-162.

Antw_90 Hans van Antwerpen: Electronic Cash; Centre for Mathematics and Computer Science (CWI), Amsterdam, October 11, 1990.

Beav_91 Donald Beaver: Secure Multiparty Protocols and Zero Knowledge Proof Systems Tolerating a Faulty Minority; Journal of Cryptology 4/2 (1991) 75-122.

Bran_94 Stefan Brands: Electronic Cash Systems Based on The Representation Problem In Groups Of

[6] Moreover, one may have the funny requirement of a partly dishonest payer that she is not falsely convicted of more double-spending than she actually carried out.

Prime Order; Crypto ’93, LNCS 773, Springer-Verlag, Berlin 1994, 302-318.

Chau_83 David Chaum: Blind Signature System; Crypto ’83, Plenum Press, New York 1984, 153.

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

Chau_89 David Chaum: Privacy Protected Payments – Unconditional Payer and/or Payee Untraceability; SMART CARD 2000: The Future of IC Cards, Proceedings of the IFIP WG 11.6 International Conference; Laxenburg (Austria), 19.-20. 10. 1987, North-Holland, Amsterdam 1989, 69-93.

ChFN_90 David Chaum, Amos Fiat, Moni Naor: Untraceable Electronic Cash; Crypto ’88, LNCS 403, Springer-Verlag, Berlin 1990, 319-327.

ChPe_93 David Chaum, Torben P. Pedersen: Transferred Cash Grows in Size; Eurocrypt ’92, LNCS 658, Springer-Verlag, Berlin 1993, 390-407.

Cres_94 Giovanni Di Crescenzo: A Non-Interactive Electronic Cash System; Proc. Italian Conference on Algorithms and Complexity, CIAC 94, LNCS 778, Springer-Verlag, Heidelberg 1994, 109-124.

Damg_90 Ivan Bjerre Damgård: Payment Systems and Credential Mechanisms with Provable Security Against Abuse by Individuals; Crypto ’88, LNCS 403, Springer-Verlag, Berlin 1990, 328-335.

EvGY_83 Shimon Even, Oded Goldreich, Yacov Yacobi: Electronic wallet; Crypto ’83, Plenum Press, New York 1984, 383-386.

Ferg_94 Niels Ferguson: Single Term Off-Line Coins; Eurocrypt ’93, LNCS 765, Springer-Verlag, Berlin 1994, 318-328.

FrYu_93 Matthew Franklin, Moti Yung: Secure and Efficient Off-Line Digital Money; 20th International Colloquium on Automata, Languages and Programming (ICALP), LNCS 700, Springer-Verlag, Heidelberg 1993, 265-276.

MiRo_91 Silvio Micali, Phillip Rogaway: Secure Computation (Chapters 1-3); Laboratory for Computer Science, MIT, Cambridge, MA 02139, USA; distributed at Crypto ’91.

Naor_91 Moni Naor: Bit Commitment Using Pseudorandomness; Journal of Cryptology 4/2 (1991) 151-158.

OkOh_91 Tatsuaki Okamoto, Kazuo Ohta: How to Utilize the Randomness of Zero-Knowledge Proofs; Crypto ’90, LNCS 537, Springer-Verlag, Berlin 1991, 456-475.

OkOh_92 Tatsuaki Okamoto, Kazuo Ohta: Universal Electronic Cash; Crypto ’91, LNCS 576, Springer Verlag, Berlin 1992, 324-337.

Pfit_93 Birgit Pfitzmann: Sorting Out Signature Schemes; 1st ACM Conference on Computer and Communications Security, 3.-5.11.1993, Fairfax, acm press 1993, 74-85.

PfWa_92 Birgit Pfitzmann, Michael Waidner: How to Break and Repair a “Provably Secure” Untraceable Payment System; Crypto ’91, LNCS 576, Springer Verlag, Berlin 1992, 338-350.

SaPe_92 Alfredo de Santis, Guiseppe Persiano: Communication Efficient Zero-Knowledge Proofs of Knowledge (With Applications to Electronic Cash); STACS 92, 9th Annual Symposium on Theoretical Aspects of Computer Science, LNCS 577, Springer-Verlag, Heidelberg 1992, 449-460.

Yao_82 Andrew C. Yao: Protocols for Secure Computations; 23rd Symposium on Foundations of Computer Science (FOCS) 1982, IEEE Computer Society, 1982, 160-164.