RZ 2823 (#90126) 05/06/96
Computer Science/Mathematics 29 pages

Research Report

Properties of Payment Systems: General Definition Sketch and Classification

Birgit Pfitzmann

Institut für Informatik
Universität Hildesheim
Samelsonplatz 1
D-31141 Hildesheim, Germany
email: pfitzb@informatik.uni-hildesheim.de

Michael Waidner

IBM Research Division
Zürich Research Laboratory
Säumerstrasse 4
CH-8803 Rüschlikon, Switzerland
email: wmi@zurich.ibm.com

Research Division
Almaden • T.J. Watson • Tokyo • Zurich

Properties of Payment Systems
— General Definition Sketch and Classification —

Birgit Pfitzmann, Michael Waidner

May 6, 1996

Abstract

We present a systematic treatment of the properties of digital payment systems. By properties, we mean that we abstract from internal details and only consider what kind of service the system offers its users, which may be people or other processes. In particular, the integrity properties are meant as a sketch of a general formal definition of payment systems, which can be filled in with moderate difficulty given previous papers; and there is currently no other comprehensive definition (even of only the integrity and at this degree of detail) of payment systems in the literature.

1 Introduction

Digital payment systems are increasingly important, both for payments on networks, such as the Internet, and for payments in shops.

This paper gives a systematic treatment of the properties of such payment systems. By systematic, we mean that we take a scientific approach at these systems; this should allow us to derive a general formal definition of digital payment systems from the properties we consider as vital. By properties, we mean that we abstract from internal details of the payment systems and only specify what kind of service the systems offer at their user interface.

For surveys that are less systematic, but include internal mechanisms of payment systems and many references, see [JaWa1 95, Sing 95]. More information on systems proposed in practice can be found at

<http://www.zurich.ibm.com/Technology/Security/sirene/outsideworld/ecommerce.html>.

An overview of previous definitions of payment systems and other definitions related to ours is deferred to Section 4, where more terminology is available.

2 Model and Terminology

It is tempting to start an overview on electronic payment systems by saying that these systems are almost like real cash or other well-known means of payment, and to start with terms like electronic money and electronic cash (actually we did so in [JaWa 95, AbAW 96]). However, there are many systems with many different properties, where these terms are given many different meanings. Furthermore, all these systems have some properties different from non-electronic ones, in particular from cash. Finally, people from the banking world are unhappy if computer scientists use words like money, which have specific legal meanings, too freely.

IBM Research Report RZ 2823 (#90126) 05/06/96

We will therefore not assume any of these words a priori. Instead, we consider payment systems as distributed systems and take a normal computer science approach to them.

2.1 General Model: Payment Systems as Distributed Systems

For general architectural aspects, we use ISO-OSI terminology [Lini 83, Tane 88]. A distributed *system* consists primarily of logical *entities* that interact with each other and with the *environment* of the system. For the interaction, they use an underlying communication system. Entities in the environment are called *users*. Logical points where users can interact with the system are called *access points*. We can regard a payment system as one layer in a layered system. Underneath is the communication system. Above, there may be human users or, e.g., an electronic commerce application that includes not only payments, but also electronic offering, ordering, and delivery. The *service* of a distributed system is the range of its possible interface behaviors. An interface behavior is a sequence of inputs and outputs at the access points. One can distinguish entity types and actual entities (in other terminologies, programs and processes, or object classes and objects), but in the following, this is usually irrelevant or clear from the context.

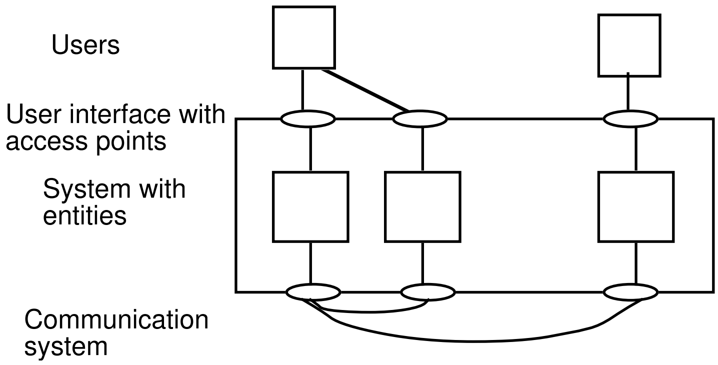

Figure 1 Model of a distributed system

When describing and classifying payment systems, one should distinguish the following aspects:

1. *Abstract service specification*, i.e., a description of the service independent of anything internal to the system.
2. *Concrete service specification*, i.e., syntactic details of the representation of the interface events, such as text, graphic, and menu guidance for inputs.
3. *Structure*, i.e., what entities there are and how they have to be connected.
4. *Protocol specification*, i.e., rules that describe how the entities have to react on messages (both from the environment and internal), without fixing the internal behavior of the entities.
5. *Entity implementation*, i.e., rules for the internal behavior of the entities.
6. *Physical devices* in which the logical entities are implemented.
7. *Usage*, i.e., what the users, in particular human ones, do with the system.

We will mainly treat the abstract service specification and the structure. This is natural for a systematic treatment of a class of distributed systems. However, particularly for payment systems, we cannot completely neglect the remaining aspects.

The concrete service specification is not treated in the following, but it will have to be standardized. Where the users are application programs, the obvious reason is that one application may have to work with several payment systems. Where the users are human, one wants to make them responsible for their inputs to the system, which presupposes that they are aware of the consequences of their inputs. If one wants to guarantee a priori that this is the case, or at least that courts will judge that it is, one needs widely known and well-evaluated user interfaces, just as it is currently well-known what the consequences are of writing a hand-written signature on an empty or filled-in cheque.

Physical devices are usually not considered at all in the field of distributed systems, because services and protocols are independent of them. However, with payment systems in practice, people often do talk about devices, e.g., they speak of a smartcard-based system. Primarily, this is a only because most systems are proprietary and come with specific devices. However, we really have to consider some aspects of trust in devices, see Section 2.4.

Usage is in theory obviously not a part of a system. On the one hand, though, one will have to educate users about "proper usage." On the other hand, one has to keep in mind that what is theoretically proper usage may be handled quite differently in practice, and that this really is no matter of the system and should be discussed separately. The main issue here is *risk management*. For instance, in a system where users electronically reserve money on their bank accounts for particular purposes, "proper usage" is that the bank allows them to reserve as much money as they have, or down to their credit limit. But the bank may also decide to allow some overreservation based on statistics that not all reservations are actually used. Thus, technically, the system will have interface events that may be called "*reservation_request*", "*reservation_confirm*" etc., but whether they are used for reserving real money or credits or nothing at all has to be regulated by outside contracts.

2.2 Access Points and Entities

The types of entities present in a payment system usually correspond to human users in certain roles. Even though this is usage, we will give names to the roles that are assumed. More technically, this means that the system offers different types of access points, one for each role.

- *Buyer*. A normal user who wants to spend money electronically. Other words that some authors use with the same meaning are *payer*, *customer*, or *client*.
- *Merchant*. A user who receives money electronically. Other words are *payee* and *shop*.
- *Buyer's bank*. All payment systems need a party where a buyer has a normal bank account or via whom the buyer converts other external forms of money, e.g., cash, into the electronic payment system. Other terms that are sometimes used with the same meaning are *issuing bank*, *electronic money issuer*, or simply *issuer*. Words such as *billing server* and *payment server* are also used similarly, but sometimes also for the merchant's bank.
- *Merchant's bank*. Another term that is sometimes used with the same meaning is *acquirer*. In real life, this role (and also the previous one) may be split into several legal entities, e.g., a bank might subcontract acquirers to collect something from the merchants for it. Such

hierarchies are usually not mentioned in electronic payment system, because each bank might have a different procedure, and the implementation is usually straightforward.

The following system entities correspond to these roles. Technically, it is a structural decision that there is essentially one entity per access point; in practice, this is quite clear by the assumed physical distribution of the system, i.e., one device per person.

- *Buyer's entity*. Other words are (*electronic*) *purse* or *wallet*, but they are often reserved to "cash-like" electronic payment systems, and sometimes used for devices. Electronic wallet may also mean, more reasonably, a multi-functional device that implements most functions of a normal wallet. Furthermore, the word is used for a specific entity in a specific model regarding trust in devices [ChPe1 93].
- *Merchant's entity*. Other words are (*electronic*) *till*, used similar to "electronic purse," and *point-of-sale terminal.* The latter is mostly used for devices, sometimes with an implication of tamper-resistance, and often reserved to payment systems used in shops.
- *Entity of the buyer's bank*. Internally, such an entity is likely to be distributed again.
- *Entity of the merchant's bank.*

Three more standard roles and entities are needed in almost all payment systems, although they are not always mentioned explicitly:

- *Courts*. Parties that may be called upon to settle a dispute. Another name is *arbiter*.
- *Clearing centers*. Parties mediating the transfer of real money from the buyer's bank to the merchant's bank corresponding to a payment.
- *Certification authorities*. Parties that perform registration or initialization functions. In real life, they may be central banks or credit-card companies for banks. For buyers and merchants, the best solution is to use notary publics. Technically, certification authorities certify keys.

Specific payment systems may have specific further roles, e.g., anonymity servers, or roles for parties that mostly interact with the standard parties outside the system, such as tax or law enforcement authorities. Moreover, there may be more types of entities than roles, in particular, so-called *observers* [ChPe1 93], which correspond to a security module trusted by one party, such as a bank, in a device, such as a mobile phone, that is held by another party.

2.3 Interface Events

A payment system allows certain types of interface events. Usually, they are grouped into transactions, such as "a payment" or "a withdrawal of money from an account." Transactions may be more intuitive than interface events, but the individual interface events are more similar between different payment systems, and thus a better starting point for a general systematic treatment.

2.3.1 Money Transfer Interface Events

The idea of a payment system is that money is paid or, more generally, transferred from one person to another. By money, we mean the real money external to the system. To make this clear, we will sometimes speak of *external money*. In the assumed usage of an electronic payment system, the external money is almost always represented by bank accounts that can be used for many other purposes besides the electronic payment system. A transfer of money needs the consent of the buyer, but often also the consent of the merchant and the buyer's bank. Actually, we will see that the buyer's bank is always needed at some point to ensure that the buyer has enough external money to transfer. However, the proper usage of the system is that whereas buyer and merchant are free in their decisions about a transfer of money, the buyer's bank *should* consent to payments if the buyer has the money. Thus, at the buyer's bank, the access point to the electronic payment system will usually not be handled by a human, but by a program that administrates the bank account. Thus payment systems have three standard input interface events:

- *pay*. This is an input by a buyer that initiates or consents to a payment. The parameters include an amount and the merchant. For payments in shops, the merchant is often designated in the trivial form of "the merchant with whose device my device is now connected." Furthermore, a subject parameter is often needed to link the payment to its purpose.
- *receive*. This is the corresponding input by the merchant. It also includes the amount and the subject if there is one and, in some form, the buyer. One may allow payments without an input *receive*, but as soon as payments comprise receipts or other proofs of payment that may be linked to obligations, it is better to permit merchants to refuse.
- *allow*. Here the buyer's bank guarantees that a certain amount of external money is there and can be used for electronic payments. *allow* always has an amount as one parameter, and it may also specify a number of payments, individual maxima, or specific merchants. The exact parameters of *allow* and at what time it should be input is arguably the most important distinction between payment systems.

Systems where *pay* is generalized as much as *allow*, i.e., the payer may authorize more than one payment, or without completely specifying the amount or the merchant, will be called general *money transfer systems*.

The most important output events are the following:

- *deduct*. This signals to the buyer's bank that, as an assumed usage, it should deduct money from the buyer's account. The amount must be one parameter or known from the context at this time. *deduct* is usually in the same transaction as *allow*, i.e., if a certain amount is now authorized to be spent electronically, it cannot be spent in other ways.
- *add*. This signals to the merchant's bank that, as an assumed usage, it should add money to the merchant's account.
- *paid* and *received*. They signal to the buyer and the merchant, respectively, that a payment has worked.

There are usually more informational outputs that show to one participant what other participants have done so far.

If *allow* and thus *deduct* is for more than one payment, we need an additional input

- *withdraw* for the consent of the payer; it does not specify a merchant, unless explicitly stated differently. An output *withdrawn* signals to the buyer that the withdrawal has worked.

Similarly, in some systems, the output *add* may be delayed until some time after one or more payments; then there is an additional input

- *deposit* from the buyer that starts a corresponding transaction. An output *deposited* signals to the merchant that the withdrawal has worked.

2.3.2 Other Interface Events

- *Disputes*. It is important to regard some transactions for settling disputes as parts of the payment system, even though they would only be a small part of the procedure in a real court. For instance, the security of a withdrawal is only a meaningful notion in relation to the procedure for settling a dispute where a client claims that a withdrawal listed on her statement of account is wrong. There are interface inputs for starting disputes, and the main output is at the court's access point and indicates, as an assumed usage, how the court should decide (compare the disputes of signature schemes in [Pfit4 93]). Furthermore, there may be outputs in normal transactions that indicate the need for a dispute.
- *Initialization*. These are interface events at the system start and for the addition of new buyers and merchants. Usually, they involve certification authorities.
- *Information*. Apart from informational outputs following any types of inputs, most systems allow specific commands that call up information, such as how much money is currently available for electronic payments according to the previous commands *allow* (this is sometimes called checking the electronic balance) or looking up one's previous transactions.

Some payment systems have specific further functionality, see Section 3.1.

2.4 Devices

As mentioned in Section 2.1, a payment system can in principle be implemented in devices of various forms. In Section 3, we will only use a few abstract properties of them. Here we describe these properties and to what real devices they correspond.

- *Computing power*. We reserve the name "device" for physical objects that can be used to implement entities on, i.e., devices must have some computing power. Thus magnetic stripe cards and pure memory cards are not devices in our sense, but special storage media.
- *Trust*. We will often simply say "a person *X* trusts a device *Y*". In real life, trust in a device can never be completely well-founded, and it is even subjective. It requires measures in all steps of the design, implementation, personalization, and operation of both hardware and software, which we cannot go into here; see [PPSW 96]. Note that these measures should ultimately be verifiable by whoever is supposed to trust the device. Moreover, an important aspect of the usage of systems is that the trustworthiness of devices will and must be questioned in disputes, and this may be far more time-consuming than the digital dispute transactions.
- *Tamper-resistance* is a particular aspect of trust, where a person trusts a device even when it is in the hands of another person who may try to manipulate it. There is no precise unclassified literature on tamper-resistance; see [Krus 91, Pate 91] for at least some details. However, it seems to be generally believed that the tamper-resistance of small devices can

be broken by attackers with access to a suitable laboratory and willing to spend a certain amount of time on each device. Devices with an internal battery can be protected much better.

- *Secure access points*. We will often simply say "a person *X* inputs a command into her device *Y*." This presupposes a secure input channel. Thus *Y* must have a user interface of its own, at least a small display and buttons for saying yes or no, unless *X* has another device it trusts. We cannot go into details of the mutual identification between the user and the device. For an overview of how a device can identify its owner, see [DaPr 89, Chapter 7]; currently, PINs and passwords are still the only viable measures on fairly cheap devices. Note that system entities, in contrast, can recognize each other by cryptologic means. That people must also identify their device is often neglected; otherwise they could be tricked to input a secrets, such as a password, into a device that stores it for an attacker who later uses it with the stolen genuine device. Such fake-device attacks are known from automatic teller machines, see [Ande3 94].

Currently, the most well-known types of buyer's devices are smartcards for payments in shops, and normal personal computers for payments over networks. A problem with smartcards is that they have no user interface, i.e., they cannot implement secure access points. A problem with personal computers is the trust in the operating system, i.e., the separation from malicious other applications, is not well funded.

In the future, all types of mobile computers may be more important. The payment application would then be one out of many. See [PPSW 96] for more details.

3 Properties

We now describe important properties of payment systems, both properties common to all systems and properties that distinguish different classes. We only consider service and structure from the list in Section 2.1, but we subdivide the service properties into functional properties, which describe what should happen in the correct case, fraud protection, and privacy.

3.1 Functional Service Properties

3.1.1 Money Transfer Transactions

One of the most important distinctions between payment systems is how the money transfer interface events are grouped into transactions. By *transaction*, we mean that one has the usual computer-scientific goals: atomicity, isolation, and permanence. The following list is not exhaustive in the theoretical sense, but covers the existing or proposed systems.

- *Pay-now* systems have just one money transfer transaction, which we call a (pay-now) *payment*. It is started by inputs *pay*, *receive*, and *allow* from the three respective participants. The parameters of the three inputs should be consistent, i.e., the amount should be the same and the merchant intended by the buyer should be the actual merchant. Similar preconditions hold for all transactions, and we will not mention them in the following. The payment yields all the four outputs *deduct*, *add*, *paid*, and *received*.

Real-world examples are point-of-sale payments with Eurocheque cards in Europe, or credit card transactions with on-line authorization.

- *Pre-paid* systems. Other names are *stored-value* and *cash-like*, but at least the latter can also have other meanings. These systems are distinguished by a specific *withdrawal* transaction. It has an input *withdraw* that does not specify the merchant, and leads to the outputs *deduct* and *withdrawn*. The payment transaction has the remaining two inputs, *pay* and *receive*. We now have to subdivide:
  - In *deposit-now* systems, the payment transaction already produces all the remaining outputs *add*, *paid*, and *received*. Real-world examples are phone cards if one distinguishes between the phone as the merchant and the phone company as the merchant's bank.
  - In *deposit-later* systems, the payment transaction only produces the outputs *paid* and *received*, but not *add*. There is a special transaction *deposit* for that, where the merchant inputs *deposit* and the output *add* occurs, and *deposited* for the merchant. Similar to a withdrawal, a deposit may be for more than one payment. Real-world examples are cash and traveler cheques.

The following models may seem stranger from a theoretical point of view:

- *Reservation* systems are like *pay-now* systems, except that they have a deposit, i.e., the output *add* does not occur in the payment, but in a separate transaction as in deposit-later systems. This is strange in the digital world, except for certain privacy models, because one needs an on-line connection to the buyer's bank during payment, to take the input *allow* into account, so why defer the deposit. However, credit card payments with hand-written signatures and on-line authorization are of this type.
- *Provisional payment* systems have a payment transaction that is not definite. In all variants, it has the input *receive* and does not produce any of the outputs *deduct*, *add*, *paid*, and *received*. Instead, there will be preliminary informational outputs. Later, there is a deposit that may fail for the merchant in either of two ways:
  - With *post-allowance*. The buyer already inputs *pay* in the provisional payment, but the buyer's bank only inputs *allow* in the deposit. A real-world example is an unguaranteed paper cheque system.
  - With *revocation*. The buyer takes part in the deposit and may revoke the payment; we therefore call the buyer's input in the provisional payment *provisional-pay* only, and her input in the deposit *pay* or *revoke*. The input *allow* can occur before or during the payment or in the deposit. Revocation is typically allowed in systems with technically insecure payments, where it is not clear whether the claimed payer really took part.

  It is up to the merchant's risk management whether he delivers goods for the provisional payment. In the case with *post-allowance*, the assumed usage is that it is considered fraud by the buyer to initiate a payment that will not be allowed.

We have not used the words *credit* and *debit* systems above, because their most usual semantics only concerns the assumed usage of a system: In a debit system, the bank allows payments only if the buyer has money, whereas in a credit system, the buyer has a certain credit limit. Alternatively, "debit" is sometimes used as a synonym of "prepaid", and "credit" for systems where *deduct* occurs later than the payment in some sense, e.g., reservation and post-allowance systems. This latter usage is also sometimes called *pay-later*.

Finally, here are some classes of more general money transfer systems:

- *Periodical payment order*. The buyer inputs *pay* for several completely specified payments to one merchant at once. The individual payments are later initialized by the buyer's bank with *allow* (usually without *receive*), and the bank would call the buyer back outside the system if it cannot enter *allow* for other reasons. In real life, periodical payment orders are, e.g., used to pay rent regularly.
- *Preauthorized debit*. The payer inputs *pay* for several payments to a specific merchant, but for unspecified amounts. Later, the merchant initiates individual *collect* transactions with the inputs *receive* and *allow* and all four outputs. Revocation is usually allowed, but often by an extra transaction outside the electronic system. Standard usage is in conjunction with contracts that specify what amounts the merchant may reasonably collect. In real life, preauthorized debit systems are, e.g., used to pay phone bills.
- *Rated payments* also leave the amount or number of payments open in *pay*. In contrast to preauthorized debit, however, *allow* for a certain maximum amount is input in *pay* or in a withdrawal before. Rated payments are usually without revocation. For security, one then needs receipts so that buyers can complain externally if merchants collect too much money. If the maximum amount input in *pay* is very small, one may give up this security for efficiency; this is a typical case of so-called *micropayment systems*. Real-life examples are paying in phone booths with large pieces of cash or phone cards.
- *Anticipated payment systems* are like prepaid deposit-later systems except that the buyer has to specify the merchant during withdrawal already. A real-life example are guaranteed cheques for a specific recipient.

3.1.2 Details of Money Transfer Transactions

There are some differences in the parameters that can or have to be input in the standard money transfer input events, and similarly in the outputs. Here are the most important ones:

- *Limits*. The amounts that can be paid may be limited by the buyer or by the bank both above, usually for risk limitation, and below, e.g., because a particular systems is uneconomic for small amounts. Similarly, the number of payments or the amounts transferred in other transactions may be limited.

  There may also be different limits on different actual ways of spending. In particular, a buyer should be able to *unlock* a certain amount for spending without identification, i.e., typically without PIN input [Deth 94]. This is important to avoid PIN input in places where the buyer is likely to be observed. Unlocking may or many not be restricted to one payment, and relocking should be possible.
- *Designated merchant*. We wrote that the payer designates the merchant to whom a payment should go. However, this may be specific and enforced to different degrees. For instance, sometimes the designated merchant can change this choice, and sometimes even the bank can.
- *Expiration* is primarily a property of prepaid systems: the influence of *allow* may be for limited time only. In simpler words, stored "electronic money" may become invalid for the

normal transaction *payment* after some time. If a system has this property, there should be other ways to free the money again, similar to exchanging old bank notes at banks. In most non-prepaid systems, only the initialization expires, i.e., technically the user's keys.

- *Divisibility* or *fungibility* is also primarily a property of prepaid systems. We speak of a fully divisible system if an amount reserved for electronic payments by *allow* can be spent in any combination of payments, within the above-mentioned limits on the amount per payment. Any additional constraints are said to restrict divisibility. The following two constraints are important, because they occur for technical reasons related to security in some systems.
  - *Number of payments restricted*. Technically, this occurs in systems where the buyer's entity has to obtain digital objects similar to cheques from the bank in advance, and fill one out in each payment, e.g., in [BoCh 90].
  - *Coin-like divisibility*. This means that money is handled electronically in individual pieces of certain denominations, just as real cash and bank notes, e.g., in [Chau 89] and most other cryptographic payment systems. A number of improvements on coin-like divisibility are possible; see, e.g., [Okam 95].

3.1.3 Extensions to Money Transfer Transactions

The following extensions, which exist in some payment systems, add new parameters to the well-known interface events, or offer new interface events and transactions.

- *Receipts* are means for a buyer to prove to third parties that a payment was made. For real-life use, all systems should provide receipts, but few actually do. Receipts only change one standard money transfer interface event slightly: the output *paid* for the buyer has an additional Boolean parameter that indicates whether the buyer's entity can in fact prove that a payment was made. In most cases, this means that inside the system, a specific receipt message arrived. The main addition is a transaction *received_dispute* for disputes about payments. Moreover, there is usually a transaction for error handling if no receipt arrived, e.g., to obtain a replacement for the receipt from the bank. See, e.g., [BüPf 89, PWP 87].
- *Transferability* is a property of prepaid deposit-later systems, or more generally of systems where the output *add* does not occur in payments, and roughly what distinguishes real cash from traveler cheques: In a system with transferability, the merchant can immediately reuse received money in the role of a buyer, whereas in a non-transferable system, it must first be deposited and then freed for spending with *allow*. For transferability in cryptographic systems, see, e.g., [ChPe 93].
- *Multi-currency* is an important issue at least in European systems. Simple solutions just allow different currencies to be distinguished, but all conversions are outside the electronic system. More complicated systems have explicit conversion transactions or even conversions integrated into payments.
- *Cancellation* means that a transaction, usually a payment, can be undone. It is not provided in most systems, and not essential, because one can transfer money back by other means. In a purely digital system, in contrast to one strongly relying on tamper-resistance, one cannot simply reset all entities to their internal state before the transaction,

because malicious participants can again set theirs to the state after the transaction. However, one can achieve that the undone transaction has no effect on the future interface events.

- *Loss tolerance* is an addition that is mainly interesting for prepaid systems, or more generally those with a withdrawal. In simple prepaid systems, a buyer who loses her device also loses money, because the amounts for which a withdrawal has already taken place, but no payment yet, are usually not accessible again without that device. Loss tolerance means that there are specific transactions for recovering money that has been lost in this way, see [WaPf 90, PfWa 95]. Of course, loss tolerance measures have to maintain security against fraud even if a buyer falsely claims to have lost her device and tries to continue using it.

More transactions are certainly conceivable, e.g., one for paying withdrawn money back into the account, without the detour of a payment to oneself or the bank.

3.1.4 Functional Properties Regarding Other Transactions

- *Openness and universality* are rather vague terms that mean that everybody can take part in a payment system and use it for any purpose. To a large extent, this is a question of usage, not of the digital system. However, one would at least require that the system allows several banks, that buyers and merchants can enter or leave at any time, and that the buyer's and the merchant's bank need not be the same. These properties have influence, e.g., on the initialization transactions needed. Moreover, for later sections, they imply that one should assume as little trust as possible.
- *Clearing*. For simplicity, we have not shown interface events for clearing above, because the whole clearing might be done digitally and automatically. However, in the near future, banks would use the existing clearing infrastructure, and thus the electronic payment system would produce outputs, say, *clearing_add* if they have to transfer money and *clearing_deduct* if they are entitled to receive it, for the banks to use as inputs to that clearing system. The main distinction in the service is then whether these outputs concern individual payments, or whether they are accumulated.
- *Disputes* should not occur at all unless in cases of fraud. Hence we consider them in Section 3.2.
- *Compatibility* would mean the existence of transactions for conversions with other payment systems. In particular, compatibility between older and newer electronic payment systems would be desired, but newer systems nowadays typically offer higher security, which might be compromised by linkage to the older system. Thus conversions are usually made outside the systems via bank accounts.
- *Accounting*, *taxing*, *law enforcement*, *etc*. This category contains all additional transactions that allow third parties to verify previous transactions, unless for the purpose of settling disputes between the standard parties. One has to distinguish what information the third parties are to obtain, and whose consent or cooperation is needed. For details, see Section 3.3., because one has to find a balance with privacy.

3.2 Security Against Fraud

The basic goal of security against fraud is that nobody wants to lose money, except, of course, by transferring it deliberately. Thus the basic security distinction between different payment systems is for which parties this goal is in fact satisfied and under what assumptions about the goodwill of other parties and the correctness of devices.

3.2.1 Framework

Security is usually roughly categorized into *integrity*, *availability*, and *confidentiality*. Integrity and availability concern the service that the system provides in the usual sense of distributed systems, whereas confidentiality concerns information or knowledge that parties gain about the behavior of others. Each such requirements has to be fulfilled in the interest of certain parties even if other parties are dishonest and their devices manipulated. This is called a threat, or, conversely, a trust model.

In this section, 3.2, we consider integrity and availability. They can be differentiated just like safety and liveness [AlSc 85]. This is not a usual definition, but it seems to be the only way to formalize these notions. If we make a general availability requirement that each entity locally finishes each transaction somehow in a certain period of time, so that it becomes usable again for other transactions, and if all the interface events of the systems considered are grouped into transactions, the remaining requirements are technically integrity requirements. Readers familiar with formal methods for reactive systems may, e.g., imagine our following requirements to be expressed in a temporal logic with variables and unrestricted quantification.

For the actual requirements, we restrict ourselves to a system with the money transfer events from Section 3.1.1 and disputes, and we postpone considering interruptions until Section 3.2.6. Additions that transfer money, e.g., loss tolerance measures, clearing details, and transferability have to be treated similarly. Moreover, requirements on the non-money transfer outputs would have to be formulated, too.

3.2.2 Multi-Party Security

Ideally, we want any payment system to offer multi-party security. This means that

- security requirements of all parties are considered, and
- the security of each party relies as little as possible on assumptions about the behavior of other parties and the security of the other parties' devices.

We cannot avoid that people have to trust their own devices, including the operating system and the payment system software, which is enough of a problem; see Section 2.4.

Nobody doubts that banks should not have to rely on the well-behavior of their clients. In an open system, it should also be clear that buyers should not have to rely on the honesty of merchants or vice versa. However, we will discuss why it is also desirable that buyers and merchants should not be forced rely on the honesty of "the bank".

First, even if the question were just about "the bank", why should they? Of course, many people will trust banks, but certainly some do not, and in a free society nobody should be forced to trust one particular organization. Choosing between different banks is no help, because ordinary people cannot evaluate their trustworthiness. Moreover, conventional payment systems offer multi-party security. For instance, all cash withdrawals and money transfer in a paper-based system need the client's hand-written signature. It would be a step backwards if electronic payment systems gave this security up.

Secondly, it is also a disadvantage for the banks if the clients' security depends on them, because that means that there is no technical way to distinguish client fraud from bank insider fraud. Thus real courts may also sometimes decide in favor of a client even if the bank and all its insiders *are* honest.

Thirdly, of course, the issue is *not* primarily about "the bank," but about individual bank insiders. With the current size and complexity of payment systems, even the best efforts of a bank at preventing insider fraud by organizational measures are far from excluding it. In particular, the bank cannot prevent fraud by designers, operators, and maintenance personnel of soft- and hardware. Current specification and verification techniques would probably not even be sufficient to tackle systems of this complexity if this were tried, in particular where absence of hidden functionality has to be verified. What sort of simple flaws in organizational security can actually happen, and how bad the consequences can be if client fraud and bank insider fraud cannot be distinguished, can be found in [Ande3 94].

For an example of a payment system that was designed explicitly for providing multi-party security see [BGHH 95].

3.2.3 Effectiveness Requirements

The first class of requirements say how transactions should end in the correct case, i.e., if all participants in that particular transaction are honest and make consistent inputs. Consistent means, e.g., that buyer and merchant input the same amount in a payment, and that the amount does not exceed general system limits.

- **Payments including *allow* and withdrawals.** Here the requirement is simply that if all parties make the inputs sketched in Section 3.1.1, all parties obtain the outputs sketched in Section 3.1.1.
- **Payments without *allow*** have to succeed if the buyer and merchant make consistent inputs, and if, at least, the buyer has enough *electronic money*: In this type of system, we can formally define the amount of electronic money a buyer has as the sum of the amount parameters of previous outputs *withdrawn* minus the sum of the amount parameters of previous outputs *paid*. In systems with restricted divisibility (see Section 3.1.2), there are additional preconditions.
- **Deposits** have to succeed if the merchant has enough *received electronic money*. This can be defined in systems with deposits as the sum of the amount parameters of previous outputs *received* minus the sum of the amount parameters of previous outputs *deposited*.

Note that the definitions of *electronic money* and *received electronic money* are not bound to any internal representation of that money in the respective entities. Instead, they denote how much money the users *believe* they have in the electronic payment system just by regarding their local inputs and outputs. The effectiveness requirements say that the users can in fact pay with the money they believe to have, or deposit it, respectively.

Furthermore, note that we cannot require that the bank inputs *allow* if the buyer has enough external money, because this is not a requirement on the electronic payment system, as long as bank accounts can be used for other purposes, too. The only contribution of the electronic payment system to this real-life requirement is that all account changes due to the electronic payment system are made disputable, see Section 3.2.7.

3.2.4 Local One-Transaction Negative Requirements

Our next class of requirements are local for one participant and say that certain outputs should not occur unless the corresponding inputs, as sketched in Section 3.1.1, were made. This means no *deduct* without *allow* for the buyer's bank, no *paid* without *pay* and no *withdrawn* without *withdraw* for the buyer, and no *received* without *receive* and no *deposited* without *deposit* for the merchant.

The first requirement means that people cannot exceed their credit limit, and the second one means that no money can be stolen from payers, even though the requirements may look a little silly in this local form. As an example, recall that effectiveness of payments was defined under the local condition that the buyer, by previous outputs such as *paid*, believes that she has enough money. Now we have required that unless she wanted to pay, she believes that she still has enough money. Together, such simple requirements therefore yield that unless she paid the money before, she can still pay it now.

3.2.5 Remaining Negative Requirements

Finally, there are requirements that certain outputs should not occur under conditions that involve more than one transaction or more than one access point.

- On *add*: The banks require that the clients altogether cannot increase their money by abusing the electronic payment system. Thus at any time, the *float* should be positive, which is defined as the sum of the amount parameters of all previous outputs *deduct* minus the sum of the amount parameters of all previous outputs *add*. This is sometimes called *strong integrity for banks*; a weaker fall-back version is explained in Section 3.2.7.
- On *deduct*: If a buyer and her bank are honest, they require that *deduct*, which decreases the buyer's external money, only occurs if the buyer made the corresponding inputs. Thus at any time, the sum of the amount parameters of all previous outputs *deduct* should be less or equal to the sum of the amount parameters of all previous inputs *pay* or *withdraw* in systems without and with withdrawal, respectively.

3.2.6 Completeness of the Standard Requirements

It should never be taken for granted that one has captured all real-world requirements in a specification. Hence we will at least look at our requirements on money-transfer interface events from a second point of view and show that they seem complete from that point of view, too.

We presuppose that requirements on a digital system are requirements on its outputs, given preconditions on previous events, mainly inputs. In particular, as we think we have captured all availability requirements in one sweep, we have only integrity requirements left, and we think that we can capture all those by considering each type of output individually and formulating conditions when it should and should not occur.

The columns of the following table show the outputs according to the access points where they occur. In the rows, we list all potentially interested parties, and in the inner part of table, we indicate whether this party has in fact interest in the occurrence or non-occurrence of that input. The symbols ⬙ are a, here imprecise, abbreviation meaning that the following input should have occurred before.

Output for whom?	Buyer		Buyer's bank	Merchant		Merchant's bank
Who requires?	*paid*	*withdrawn*	*deduct*	*received*	*deposited*	*add*
Buyer	*pay*	*withdraw*	Not without *pay* or *withdraw*, resp.	Should occur after *pay*	—	—
Buyer's bank	—	—	*allow*	—	—	—
Merchant	—	—	—	*receive*	*deposit*	Should occur after *deposit*
Merchant's bank	—	—	—	—	—	covered by *deduct*

Table 1 Outputs and requirements on them

3.2.7 Interruption Tolerance

We initially said that transactions should be atomic. However, this cannot be achieved in the strict sense because we cannot assume that any lines are completely reliable, nor that the participants are honest. There is always at least one step where one participant has already received an output and another one has not. However, it should be guaranteed that nobody can draw advantage from interrupting transactions and that honest participants do not suffer permanent loss.

- *Interruption detection.* If any correct entity makes the final output of a transaction, all other entities either make theirs, too, or output that they detected an interruption. An optional additional requirement is that such entities are not blocked until the interruption is handled.
- *Resumption.* In case only the communication was interrupted, there should be a transaction type *resume_X* for each type *X* of money transfer transaction with the same participants as *X* had.
- *Interruption disputes.* If the interruption was by a malicious participant or the communication is permanently cut off, transactions are needed where a consistent state is reached without one of the original parties, usually by calling upon another party.

  Interruptions between buyer and merchant in payments can often be handled with the help of the bank. Usually, the buyer is missing a receipt, and the bank gives a replacement for that if, according to the bank's records, the merchant has received the money.

  Interruptions between a bank and one of its clients are, by the transaction design, generally such that the client is at a disadvantage, because banks are less likely to disappear when they have reached an inconsistent state. Hence, in a multi-party secure system, transactions are needed to dispute an interruption that occurred after *deduct*, but before the corresponding transaction was finished, and those that occurred before *add*, but after all the corresponding information was sent to the bank. Prepaid deposit-later systems need an additional dispute for buyers whom neither merchant nor bank send the due receipt.

Interruptions are usually not mentioned in papers about payment systems, but a real system certainly needs to handle them; e.g., they needed an important part of the actual design effort of the CAFE protocols [BBCM 94].

3.2.8 Disputes

Apart from disputes about interruptions, there are disputes about previous completed transactions.

Disputes whether an output occurred. If, by the assumed usage, an output at one party, P_{out}, allows or obliges this party to do something with an impact on another party, it must be disputable whether this output occurred. For instance, an output *deduct* at the bank allows the bank to deduct money from an account, and thus there must be a way of handling disputes where a buyer claims she did not permit the bank to deduct this money. This yields the following disputes:

- *dispute_deduct*, as just explained.
- *dispute_received*, where a buyer claims that the merchant obtained an output *received* and, e.g., should therefore have delivered goods.
- *dispute_add*, where a merchant claims that his bank obtained an output *add* and was therefore obliged to add money to his account.

The arbiter's output is *yes* or *no*, indicating whether the disputed output is now assumed to have occurred. More precisely, the arbiter decides whether this output would have occurred if P_{out}'s entity was correct. Furthermore, the decision need not absolutely be about what happened in the past, as long as a state is restored that fulfills the requirements of all honest parties.

Just like interruption tolerance, disputes are usually not mentioned in papers about payment systems, although, as we said, several security properties only make sense in their relation to disputes, and storage and verification of data needed in disputes must be an normal part of any implementation.

Weak integrity. Not all systems fulfill the strong integrity for banks, i.e., the requirement on *add*. In particular, many systems no longer fulfill it under adverse conditions, e.g., if the tamper-resistance of a device has been broken. As a fall-back, one may require *weak integrity*, which means that one can carry out a dispute and have someone convicted as responsible, in the hope of recovering the money. We call this an *overspending dispute*. The requirement is that if *float* becomes negative, the bank can carry out overspending disputes such that certain clients are convicted for amounts that cover the missing money. Correspondingly, honest buyers require that they are never wrongly convicted for overspending. One should also require that dishonest buyers are not convicted for overspending a larger amount than they did, but this is not always fulfilled.

Weak integrity is a characteristic property of cryptographic offline payment systems, in particular untraceable ones, such as [ChFN 90, FrYu 93, Bran 94].

Note that overspending disputes lead back to entities, not to people. In particular, if someone breaks the tamper-resistance of a stolen device and uses this for overspending, the device or its owner, and not the thief, will be identified. Nevertheless, flanked by non-technical measures, weak integrity as a fall-back makes breaking devices more risky.

3.3 Privacy

Privacy means that no unnecessary information about users is divulged to other parties by the electronic payment system. Note that privacy is quite independent of the fact that almost every payment system must keep some internal details secret to ensure integrity.

Among conventional payment systems, cash offers a lot of privacy: A buyer can buy a book, ride a bus, or use a public phone without identifying herself to the merchant, and no outsider — and even a bank is an outsider in such a purchase — needs to know about the payment. In contrast, conventional systems typically used for higher-value payments, such as credit cards and money transfers, offer almost no privacy. Roughly, such a division is also to be expected for electronic payments. However, before listing the most common types of privacy in proposed systems or even discussing politics, let us first look at types of privacy systematically.

3.3.1 Types of Privacy

The following terminology distinguishes different aspects of privacy.

- *Unobservability.* An event in a system, e.g., a payment, is called unobservable by a certain party, e.g., the buyer's neighbor, if the party does not even learn that the event occurred.

  More precisely, the party should not gain any additional information or knowledge, e.g., it should not be able to guess better than with a priori probability. For brevity, we do not explicitly mention that all the following properties are to be interpreted in this way.

- *Parameter secrecy.* A parameter of an event, e.g., the amount of a payment, is called secret from a certain party if that party does not learn anything about this parameter. For technical reasons, it is often allowed that the length of the parameter is learned.

- *Anonymity*. A party who takes part in an event in a certain role, e.g., the buyer in a payment, is called anonymous with respect to, or *untraceable* by, a certain other party who knows about the event if the second party cannot identify the first party. More precisely, one can see this as a special case of the next property.

- *Unlinkability.* Two roles, say R_1 and R_2, in two events, E_1 and E_2 are called unlinkable for a certain party if that party does not learn whether the same user played the two roles. For instance, buyers are called unlinkable in payments if other parties do not learn whether two payments were made by the same person. Unlinkability between different classes of events leads to a classification of pseudonyms, if we call any way to denote a person a pseudonym. A minimal pseudonym is „the person who played role R_i in event E_i".

  - *Person pseudonyms* are used over a long time and with different partners, i.e., many actions are linkable. One can further distinguish *public*, *non-public*, and *anonymous* person pseudonyms, according to whether the underlying identity is generally known (e.g., "the identity" itself or a phone number), or should only be known to one or very few parties (e.g., account numbers, depending on their use), or only to the user. However, every person pseudonym is likely to become public in the long run, because some partners the pseudonym is used with will know the identity from other sources, and thus it may leak, and even if not, enough information for deanonymization will accumulate.

- *Long-run role pseudonyms* are only used with one partner, e.g., an identifier for an anonymous account if the buyer only has to use it with the bank, but need not show it to merchants. Enough information for deanonymization may still accumulate, but more slowly and only at one place.
- *Short-run role pseudonyms* are only used in one or very few transactions. For instance, the number of a bank note may be seen as a pseudonym used in two transactions. In other words, the payment where one receives the bank note and the one where one spends it again are (probabilistically) linkable for observers who look at both bank notes, but not to other transactions.

Now we can classify systems according to which roles are linkable. An orthogonal distinction is by whom they are linkable. In particular, there are sometimes k-out-of-n models, which means that k out of certain n parties have to pool their information in order to link.

3.3.2 Common Types of Privacy

Note that not much will be said about unobservability against outsiders, because encryption can be used here almost independently of the payment system. For instance, if an Internet purchase should be secret from outsiders, the order and the goods will be encrypted, and one may use the same keys to encrypt the payment information such that outsiders cannot even distinguish if a payment takes place at all.

Extremes. One extreme with respect to privacy is that all roles are linkable by all participants, i.e., there is no privacy at all. The other extreme is that buyers and merchants are never linkable in any of their actions, not even if all other parties pool their information. Banks, having to guarantee for payments, are never supposed to be anonymous. This maximum privacy can be achieved, see [PWP 87], but it is not one of the usual models.

Standard privacy of cryptology. The standard type of privacy in the cryptologic literature on payment systems is the so-called payer anonymity from [Chau 89] and many other papers, e.g., [ChFN 90, FrYu 93, Bran 94]. It assumes a prepaid system. Normal bank accounts are used, i.e., public person pseudonyms for withdrawals and deposits. However, the buyers in payments are anonymous, and all their payments are unlinkable. The buyers can identify the merchants to whom they paid at least with the help of the bank. There is no transferability.

Some other types. Anonymous accounts, i.e., anonymous person pseudonyms, can easily be introduced into almost any payment system, because it does not actually matter in the system design if a person pseudonym is public or anonymous (see [BüPf 89, PWP 87]).

Accounts under person pseudonyms that are not quite anonymous, but linkable to a "real" identity only by n out of n parties, can generally be constructed as follows: One opens a normal account with the first party and an anonymous account with each of the others. One transfers the money from the first account via all the others with standard pay-now money transfers. Hence each party can link two successive accounts and all parties together can link the last account, which is actually used for spending the money, to the first, non-anonymous one. This underlies [MeNe 93, LoMa1 94]. One can freely choose whether the same n-out-of-n anonymous account is used for a long or a short period of time.

Another simple generic construction is to transfer money from a normal account to a fully anonymous account with a system that offers the standard privacy of cryptology, and then use the anonymous account just like any account. This underlies [CaPS 94], and, in principle, [OkOh 92].

More types are mentioned in [BüPf 89, PWP 87].

3.3.3 Technical Environment

Among the questions about how much privacy is possible and necessary, two items are purely technical.

1. Privacy of one system can never undo identification in the environment.
2. Privacy and security need not be in conflict.

More technically, the first item simply restates that privacy can formally only be defined as follows: the information an observing party has about the events that should be secret is not significantly different from the information it would have if it were not given the system observations. Addresses, timing, and amounts are information particularly likely to unintentionally undo the privacy of a payment system.

- Addresses. Technically, almost whenever one wants to act under a pseudonym, one has to be able to send and receive messages under this pseudonym. For payments in shops and the standard privacy of cryptology, this is no problem: The client just connects her device to the shop's device and they can communicate without identification. Payments over networks cannot be more private than the messages exchanged in them. For some restricted types of privacy, this is no problem again; e.g., if buyers only want to be anonymous from merchants, they can exchange messages via their bank. More privacy requires anonymity in the network, e.g., anonymous remailers or real cryptologic techniques [Chau 81, Chau 88].
- Timing. If two events obviously belong to the same transaction, or if they are at least statistically likely to belong together because of a relationship in time, they are linked. For instance, in a pay-now system with non-anonymous accounts, the two banks know who paid to whom, because the events *deduct* for the buyer and *add* for the merchant are in the same transaction. Therefore, most systems offering privacy are prepaid, and some are reservation systems. Even then, if most people withdrew money exactly when they needed it, statistical linkage would still be possible.
- Amounts. Similarly, for strong privacy, a system should not require that the amount in *deduct* is the same as in *add*.

Potential conflicts between security and privacy have to be excluded in several areas.

- For security within the payment systems, it can simply be seen from the existing systems.

  In particular, privacy need not even be a problem in classes of payment systems that are technically not perfectly secure and where the risk of identification is an important deterrent, because measures to enable selective identification of cheaters exist. An important example is that weak integrity, i.e., the possibility to identify cheaters who increased their money, typically by breaking the tamper-resistance of an offline system, can be combined with privacy.

- Measures for secure exchange of payment and goods exist for an exchange under pseudonyms just as well as under public identities [BüPf 90, PWP 87].
- Where necessary, identification outside the payment system is possible. For instance, if certain goods, such as weapons, are not to be bought anonymously, merchants of such goods have to require signed orders from their clients, and auditors have to require these orders from the merchants.

3.3.4 Social Aspects

To see why privacy in payment systems is desired at all, recall the examples of the purchase of books, bus rides, and phone calls, and note that nowadays the individual goods bought are almost always registered automatically by bar codes. One can find more spectacular examples of particular purchases requiring privacy for good reasons, but the greater danger to a society where people dare to behave freely lies in large collections of normal data that in their totality tell far too much about a person's life and thoughts.

Anyway, most people do not doubt that there is something to be said for privacy, only some fear that there might be more to be said against. Given that privacy need not impair security (Section 3.3.3), the remaining arguments are mainly about law enforcement and other audits.

First note that several technical papers propose to restrict anonymity by linkability between many payments of the same buyer, see the "other types" in Section 3.3.2. This may increase efficiency, but has nothing to do with law enforcement and audits.

Secondly, in systems with the standard privacy of cryptology, some audits are automatically possible, in particular income taxes, because the income can be seen on the non-anonymous bank accounts, as there is no transferability. Furthermore, if the system guarantees that the client can identify the merchant, identification of blackmailers and dealers is ensured. This assumes that a buyer cooperates with the law enforcement authority, which is necessary anyway, because otherwise the nature of the payment would not be known.

Still, it may be argued that under certain circumstances, a buyer should be identified with the cooperation of the merchant, or buyers and merchants should be linked with only the cooperation of the bank or even without. To do this without unnecessary identification of people who are under no suspicion, one can use special methods with distributed deanonymization information similar to the widely discussed schemes for key escrow of encrypted message transfer (parts of group cryptology, also called fair cryptosystems or simply Clipper proposals) [StPC 95].

In any case, one should weigh the dangers of the data collection, also their economic worth, against what it may help for. For instance, even if only the bank together with two different law enforcement agencies were supposed to be able to trace payments, an attacker, say, a rich one well into blackmail or industrial espionage only needs to find insiders to break into three computer systems, maybe not even independently designed, in order to obtain the keys to all this valuable information. At the very least, data that merchants need should be separated from data that banks need, and it would be a good idea to separate the functions of banks and law enforcement, too.

3.4 Structure

Essentially, all electronic payment systems assume one entity per access point, corresponding to the natural distribution of one device for each buyer and for each merchant. The main structural differences are in the necessary communication and in the need for tamper-resistant hardware.

3.4.1 Communication During Payment

A very important distinction is into

- *online* systems, where some communication with a third party is needed during a payment, and
- *offline* systems, where only the buyer's and the merchant's entity communicate during a payment.

For small-value payments for real goods, online systems are currently considered too expensive, in particular, if payments can also be made in buses and taxis. In contrast, systems proposed for Internet payments are usually online in this sense, and so are most proposed systems for high-value payments. Actually, any electronic payment system is either online or relies on tamper-resistant hardware in the buyers' devices, because electronic money, i.e., the amount that has been freed with *allow* already could otherwise be spent several times by resetting the buyer's device to the state after *allow* each time. The choice between these alternatives depends on the price of the online connections versus the price and market introduction of tamper-resistant hardware and on the fact that tamper-resistance is not absolute.

Off-line system structure is only possible for those system types where only the buyer and the merchant give inputs and receive outputs in a payment, i.e., prepaid deposit-later and some provisional or rated payment systems. This is an important advantage of prepaid systems.

3.4.2 Use of Tamper-Resistant Hardware

In some sense, all payment systems need tamper-resistant hardware for the implementation of the secure access points, i.e., to guarantee to some extent that the device of each user can only be used by this particular user. (With the bank's device, this means protection against insider attacks.) This was discussed in Section 2.4.

What distinguishes systems is to what extent one needs hardware that protects devices even against their owners. In particular, in some systems the bank has to trust the buyers' devices or the merchant's devices or both. As there are far more buyer's devices, and they must be portable at least for shop payments, they will typically be less well protected than merchant's devices. Anyway, the need for tamper-resistant devices is currently considered prohibitive for Internet payments, but as soon as credit cards start to be smartcards, this may change.

4 Related Literature

4.1 Previous Definitions of Payment Systems

Unfortunately, no fairly comprehensive definition of a secure payment system existed so far, and thus, even for the protocol part, no payment system has really been verified, i.e., even if one doubts neither the security of the devices nor the underlying explicit cryptologic assumptions. Some systems are called "provably secure" nevertheless, but this only means that some specific properties, formulated in terms of internal details of the given system, have been proved.

This is particularly unsatisfactory for systems considered in cryptology, which achieve non-trivial combinations of properties and therefore seem to need proofs most. In particular, [Damg1 90] presented a so-called provably secure payment system for untraceable on-line payments with special, construction-related definitions, but the system was not secure, and the flaw was not in the proof, but in the definitions [PfWa2 92]. (In general, it is disputable if one can "break" a system by telling that it should have fulfilled another definition, but in this case, given the introduction and the references to other systems whose properties were to be achieved in a provably secure way, no one seems to have doubted it.)

Even what is both in public and our own opinion the best current definition, that of untraceable off-line systems in [FrYu 93], is a sketch and would need at least one major modification when worked out, because it strongly operates with a system-internal notion of "coin" that is not defined if there are certain attacks; see [PfSW 95], Footnote 4. To remove this requires to work with something more like our interface events. An earlier related definition is given in [ChEv 87]; the kind of security properties that are being proved for practical cryptographic payment systems can be seen in [Bran 94].

Early definitions of simpler types of systems were given in [Waid 85, WaPf 85, Bürk 87, Stas 88]; the former consider algebraic abstractions of cryptographic primitives, following [DoYa 83], but is nearer to considering the systems comprehensively, i.e., in terms of an interface behaviour, than other definitions. Examples that the algebraic abstractions do not capture all properties of cryptographic primitives, even if those are provably secure with respect to cryptographic definitions, were given in [Pfi1_88]. Another definition of a simple system is given in [Ande2 92]. It is strongly in terms of one given construction (e.g., it defines that a key should be fresh, not anything about money), but it was used for a proof of this system in a certain formal model (algebraic abstractions as in [WaPf 85]), not just for sketches of rigorous mathematical proofs as is the intention of the other definitions. [Kail 95] seems to be the first paper except for all ours that emphasizes that disputes should be handled explicitly.

Earlier surveys, in particular of possible privacy properties, were given in [Bürk 86, BüPf 89, PWP 87].

4.2 Other Comprehensive Definitions

The type of a general specification and classification sketched below for payment systems closely follows that for signature schemes in [Pfit4 93, Pfit9 93]. It was briefly anticipated in [PfSW 95], and a version for one particular payment system was worked out in an ad-hoc mathematical notation in [Krüg 95]; some problems remained.

A similar approach at specifying requirements on cryptographic systems independently of the structure and in temporal logic was taken slightly earlier in [SyMe 93] for schemes usually treated with so-called logics of authentication [BuAN 90]. The systems considered are symmetric authentication protocols and the semantics is algebraic abstractions, not real cryptographic systems as we intend them here. The general type of requirements is very similar, except that [SyMe 93] has no explicit interface.

As these papers use temporal logic, they rely on a large body of work on specifications outside the field of security, see, e.g., [MaPn1 91]. Of course, the semantics with attacks and computational restrictions is specific to cryptology.

Privacy properties were not really handled in [Pfit4 93], nor in [SyMe 93], and they need a different type of specification. An overview of possibilities was given in [PfWa 94]. Some further details can be found in [Pfit3 95]; however, there are some unsolved problems with general definitions of individual privacy properties [Schu 94].

For an overview of other formal specifications in security in general, see [Mead 95].

5 Conclusion and Outlook

We have presented a general framework for defining and classifying digital payment systems. The main idea was to identify general interface events that they offer, independent of the transactions into which these events are grouped, and to use them to formulate general requirements meaning that the money flow is secure for all parties. We also presented further properties that can serve to classify systems within this framework.

The approach remains to be extended in two directions: On the one hand, the existing systems should be classified in this framework. On the other hand, the requirements should be formalized further and used in security proofs of individual payment systems. For the integrity properties, the semantics, i.e., what it means for such requirements to be fulfilled in the presence of attackers and with different degrees of security, are predefined in [Pfit4 93, Pfit9 93]. The formalization of the integrity requirements can also follow these examples quite closely, but one could also try out other languages than temporal logic that also define sets of sequences. A precise definition of the interface events alone, with only informal requirements, can also be used as a framework for application programming interfaces of payment systems.

For real life, one may already draw the following conclusions from the presentation of properties:

- It is likely and desirable that there will be more than one type of electronic payment system. In particular, different systems for low-value and high-value payments, similar to cash and credit cards, are useful, and also with the privacy of cash and credit cards. Note that the limit for anonymous payments can be set much higher than with conventional credit card systems in systems where fraud is impossible a priori, e.g., in a prepaid deposit-now system. Very small payments may again be treated separately, in particular, if they can be treated as rated payments.
- Online and offline payment systems, however, even though currently usually separate, are likely to grow together.
- It is neither to be expected nor to be desired that the first electronic payment systems that succeed in the market are already the final ones. Gradual development is to be expected at least in the field of personal devices. In particular, for Internet-payments, the first systems

will necessarily assume that the buyers have only PCs, but in the long run, the operating system security of PCs is not sufficient to support high security applications. Even with offline systems, most of the current trials only use smart cards, for reasons of price, but in the long run, it is essential that buyers have their own keypads and little displays.

Finally, note that electronic marketplaces do not consist of payment systems alone. For instance, there must be similarly secure and private means for secure exchange, for getting to know each other), and for copyright protection; see, e.g., the architecture of the SEMPER project [Waid 96].

Acknowledgments

We thank *N. Asokan, Mihir Bellare*, *Jens Krüger*, *Phil Janson*, *Matthias Schunter* and *Michael Steiner* for interesting discussions.

References

AbAW 96 Jose Abad Peiro, N. Asokan, Michael Waidner: Payment Manager — Overview; SEMPER Report 221ZR054, Mar. 21, 1996. <http://semper.zurich.ibm.com/info/221ZR054.ps.gz>

AlSc 85 Bowen Alpern, Fred B. Schneider: Defining Liveness; Information Processing Letters 21 (1985) 181-185.

Ande2 92 Ross J. Anderson: UEPS - A Second Generation Electronic Wallet; ESORICS '92 (Second European Symposium on Research in Computer Security), Toulouse, LNCS 648, Springer-Verlag, Berlin 1992, 411-418.

Ande3 94 Ross J. Anderson: Why Cryptosystems Fail; Communications of the ACM 37/11 (1994) 32-40.

BBCM 94 Jean-Paul Boly, Antoon Bosselaers, Ronald Cramer, Rolf Michelsen, Stig Mjølsnes, Frank Muller, Torben Pedersen, Birgit Pfitzmann, Peter de Rooij, Berry Schoenmakers, Matthias Schunter, Luc Vallée, Michael Waidner: The ESPRIT Project CAFE — High Security Digital Payment Systems; ESORICS 94 (Third European Symposium on Research in Computer Security), Brighton, LNCS 875, Springer-Verlag, Berlin 1994, 217-230.

BGHH 95 Mihir Bellare, Juan A. Garay, Ralf Hauser, Amir Herzberg, Hugo Krawczyk, Michael Steiner, Gene Tsudik, Michael Waidner: iKP — A Family of Secure Electronic Payment Protocols; Usenix 1995.

BoCh 90 Jurjen Bos, David Chaum: SmartCash: A Practical Electronic Payment System; Centrum voor Wiskunde en Informatica, Computer Science/Departement of Algorithmics and Architecture, Report CS-R9035, August 1990.

Bran 94 Stefan Brands: Untraceable Off-line Cash in Wallet with Observers; Crypto '93, LNCS 773, Springer-Verlag, Berlin 1994 302-318.

BuAN 90 Michael Burrows, Martin Abadi, Roger Needham: A Logic of Authentication; ACM Transactions on Computer Systems 8/1 (1990) 18-36.

Bürk 86 Holger Bürk: Digitale Zahlungssysteme und betrugssicherer, anonymer Wertetransfer; Studienarbeit am Institut für Informatik IV, Universität Karlsruhe, April 1986.

Bürk 87 Holger Bürk: Modellierung und Verifikation der Sicherheit und Lebendigkeit von Kryptoprotokollen; Diplomarbeit am Institut für Rechnerentwurf und Fehlertoleranz, Universität Karlsruhe, März 1987.

BüPf 89 Holger Bürk, Andreas Pfitzmann: Digital Payment Systems Enabling Security and Unobservability; Computers & Security 8/5 (1989) 399-416.

BüPf 90 Holger Bürk, Andreas Pfitzmann: Value Exchange Systems Enabling Security and Unobservability; Computers & Security 9/8 (1990) 715-721.

CaPS 94 Jan L. Camenisch, Jean-Marc Piveteau, Markus A. Stadler: An Efficient Electronic Payment System Protecting Privacy; ESORICS 94 (Third European Symposium on Research in Computer Security), Brighton, LNCS 875, Springer-Verlag, Berlin 1994, 207-215.

Chau 81 David Chaum: Untraceable Electronic Mail, Return Addresses, and Digital Pseudonyms; Communications of the ACM 24/2 (1981) 84-88.

Chau 88 David Chaum: The Dining Cryptographers Problem: Unconditional Sender and Recipient Untraceability; Journal of Cryptology 1/1 (1988) 65-75.

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.

ChEv 87 David Chaum, Jan-Hendrik Evertse: A secure and privacy-protecting protocol for transmitting personal information between organizations; Crypto '86, LNCS 263, Springer-Verlag, Berlin 1987, 118-167.

ChFN 90 David Chaum, Amos Fiat, Moni Naor: Untraceable Electronic Cash; Crypto '88, LNCS 403, Springer-Verlag, Berlin 1990, 319-327.

ChPe 93 D. Chaum, T.P. Pedersen: Transferred cash grows in Size; Eurocrypt '92, LNCS 658, Springer-Verlag, Berlin 1993, 390-407.

ChPe1 93 David Chaum, Torben Pryds Pedersen: Wallet Databases with Observers; Crypto '92, LNCS 740, Springer Verlag, Berlin 1993, 89-105.

Damg1 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.

DaPr 89 Donald W. Davies, Wyn L. Price: Security for Computer Networks, An Introduction to Data Security in Teleprocessing and Electronic Funds Transfer; (2nd ed.) John Wiley & Sons, New York 1989.

Deth 94 Jürgen Dethloff: 25 Jahre Chipkarten-Technik – Rückblick und Ausblick; 4. GMD-SmartCard Workshop, 8.-9. Februar 1994, GMD, Rheinstr. 75, D-6100 Darmstadt (jetzt 64295), Herausgeber: Bruno Struif, Tagungsband.

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.

JaWa1 95 Philippe Janson, Michael Waidner: Electronic Payment over Open Networks; Informatik 3/ (1995) 10-15. Also: SEMPER Report 211ZR017, January 30, 1996. <http://semper.zurich.ibm.com/info/211ZR017.ps.gz>

Kail 95 Rajashekar Kailar: Reasoning about Accountability in Protocols for Electronic Commerce; 1995 IEEE Symposium on Research in Security and Privacy, IEEE Computer Society Press, Los Alamitos 1995, 236-250.

Krüg 95 Jens Krüger: Sicherheitsanforderungen an digitale Zahlungssysteme und ihr Bezug zu Sicherheitskriterien; Diplomarbeit, Institut für Informatik, Universität Hildesheim, März1995.

Krus 91 Dietrich Kruse: The new Siemens computer card; David Chaum (ed.): Smart Card 2000, Selected Papers from the Second International Smart Card 2000 Conference, North-Holland, Amsterdam 1991, 3-7.

Lini 83 Peter F. Linington: Fundamentals of the Layer Service Definitions and Protocol Specifications; Proceedings of the IEEE 71/12 (1983) 1341-1345.

LoMa1 94 Steven H. Low, Nicholas F. Maxemchuk: Anonymous Credit Cards; 2nd ACM Conference on Computer and Communications Security, Fairfax, November 1994, ACM Press, New York 1994, 108-117.

MaPn1 91 Zohar Manna, Amir Pnueli: The Temporal Logic of Reactive and Concurrent Systems: Specification; Springer-Verlag, New York 1991.

Mead 95 Catherine A. Meadows: Formal Verification of Cryptographic Protocols: A Survey; Asiacrypt '94, LNCS 917, Springer-Verlag, Berlin 1995, 133-150.

MeNe 93 Gennady Medvinsky, Clifford Neumann: NetCash: A design for practical electronic currency on the Internet; 1st ACM Conference on Computer and Communications Security, Proceedings, Fairfax, November 1993, acm Press, New York 1993, 102-106.

Okam 95 Tatsuaki Okamoto: An Efficient Divisible Electronic Cash Scheme; Crypto '95, Proceedings, LNCS 963, Springer-Verlag, Berlin 1995, 438-451.

OkOh 92 Tatsuaki Okamoto, Kazuo Ohta: Universal Electronic Cash; Crypto '91, LNCS 576, Springer-Verlag, Berlin 1992, 324-337.

Pate 91 Mike Paterson: Secure single chip microcomputer manufacture; David Chaum (ed.): Smart Card 2000, Selected Papers from the Second International Smart Card 2000 Conference, North-Holland, Amsterdam 1991, 29-37.

Pfit 88 Birgit Pfitzmann: Vergleich der algebraischen und kryptographischen Modellierung von Kryptoprotokollen; Studienarbeit am Institut für Rechnerentwurf und Fehlertoleranz der Universität Karlsruhe 1988.

Pfit4 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.

Pfit9 93 Birgit Pfitzmann: Fail-Stop Signature Schemes; Dissertation, Institut für Informatik, Universität Hildesheim, Einreichung 4.9.1993, Verteidigung 20.5.1994.

Pfit3 95 Birgit Pfitzmann: Kryptographie; Skripten des Fachbereichs Mathematik, Informatik, Naturwissenschaften no. 5, Universität Hildesheim, April 1995.

PfSW 95 Birgit Pfitzmann, Matthias Schunter, Michael Waidner: How to Break Another "Provably Secure" Payment System; Eurocrypt '95, LNCS 921, Springer-Verlag, Berlin 1995, 121-132.

PfWa 94 Birgit Pfitzmann, Michael Waidner: A General Framework for Formal Notions of "Secure" System; Hildesheimer Informatik-Berichte 11/94, ISSN 0941-3014, Institut für Informatik, Universität Hildesheim, April 1994.

PfWa 95 Birgit Pfitzmann, Michael Waidner: Strong Loss Tolerance for Untraceable Electronic Coin Systems; Hildesheimer Informatik-Berichte 15/95, ISSN 0941-3014, Institut für Informatik, Universität Hildesheim, Juni 1995.

PfWa2 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.

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

PWP 87 Birgit Pfitzmann, Michael Waidner, Andreas Pfitzmann: Rechtssicherheit trotz Anonymität in offenen digitalen Systemen; Computer und Recht 3/10,11,12 (1987) 712-717, 796-803, 898-904. Revised version: Datenschutz und Datensicherung DuD 14/5-6 (1990) 243-253, 305-315.

Schu 94 Matthias Schunter: Spezifikation von Geheimhaltungseigenschaften für reaktive kryptologische Systeme; Diplomarbeit am Institut für Informatik, Universität Hildesheim, Januar 1994.

Sing 95 Andrew Singleton: Cash on the Wirehead; Byte 20/6 (1995) 71-78.

Stas 88 Mark G. Staskauskas: The Formal Specification and Design of a Distributed Electronic Funds-Transfer System; IEEE Transactions on Computers 37/12 (1988) 1515-1528.

StPC 95 Markus Stadler, Jean-Marc Piveteau, Jan Camenisch: Fair Blind Signatures; Eurocrypt '95, LNCS 921, Springer-Verlag, Berlin 1995, 209-219.

SyMe 93 Paul Syverson, Catherine Meadows: A Logical Language for Specifying Cryptographic Protocol Requirements; 1993 IEEE Symposium on Research in Security and Privacy, IEEE Computer Society Press, Los Alamitos 1993, 165-177.

Tane 88 Andrew S. Tanenbaum: Computer Networks; 2nd ed., Prentice-Hall, Englewood Cliffs 1988.

Waid 85 Michael Waidner: Datenschutz und Betrugssicherheit garantierende Kommunikationsnetze. Systematisierung der Datenschutzmaßnahmen und Ansätze zur Verifikation der Betrugssicherheit; Diplomarbeit am Institut für Informatik IV, Universität Karlsruhe, August 1985, Interner Bericht 19/85 der Fakultät für Informatik.

Waid 96 Michael Waidner: Development of a Secure Electronic Marketplace for Europe; SEMPER Report 431ZR031, Feb. 19, 1996. <http://semper.zurich.ibm.com/info/431ZR031.ps.gz>

WaPf 85 Michael Waidner, Andreas Pfitzmann: Betrugssicherheit trotz Anonymität. Abrechnung und Geldtransfer in Netzen; 1. GI Fachtagung Datenschutz und Datensicherung im Wandel der Informationstechnologien, IFB 113, Springer-Verlag, Berlin 1985, 128-141; Überarbeitung in: Datenschutz und Datensicherung DuD 10/1 (1986) 16-22.

WaPf 90 Michael Waidner, Birgit Pfitzmann: Loss-Tolerance for Electronic Wallets; Proceedings 20th International Symposium on Fault-Tolerant Computing (FTCS 20), Newcastle upon Tyne (UK), 140-147.