RZ 2858 (#90806) 09/02/96
Computer Science/Mathematics 23 pages

Research Report

Optimistic Protocols for Fair Exchange

N. Asokan

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

Matthias Schunter

Univ. Hildesheim,
Institut für Informatik,
Marienburger Platz 22
D-31141 Hildesheim, Germany
email: schunter@acm.org

Michael Waidner

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

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

Optimistic Protocols for Fair Exchange

1. Introduction

Many commercial transactions can be modelled as a sequence of *exchanges* of electronic goods involving two or more parties. An exchange among several parties begins with an understanding about what item each party will contribute to the exchange and what it expects to receive at the end of it. A desirable requirement for exchange is *fairness*. A *fair* exchange should guarantee that at the end of the exchange, either each party has received what it expects to receive or no party has received anything.

One example for fair exchange is *non-repudiation* of message transmission which is, in essence, a fair exchange of the message and a non-repudiation of receipt token for the message. In several draft documents, ISO [ISO1, ISO2, ISO3] defines non-repudiation services for transmission of messages and describes protocols that provide them. In particular they define:

- **non-repudiation of origin** which guarantees that the originator of a message cannot later falsely repudiate having originated that message, and
- **non-repudiation of receipt** which guarantees that the recipient of a message cannot falsely repudiate having received that message (the ISO draft documents use the term "non-repudiation of delivery").

A straightforward solution for the fair exchange problem, used in these ISO proposals, is to use a third party to ensure fairness by, for example, receiving the items to be exchanged and the expectations of the participants in a first step and forwarding them in the next. A drawback of this approach is that the third party is always involved in the exchange even if both parties are honest and no fault occurred. Sending messages via a third party can in practice lead to performance problems as it becomes a bottleneck. To avoid such a bottleneck, the third party can be required to have powerful computation and communication facilities – in this case, its operation will be an expensive undertaking.

In this report, we describe generic protocols for fair exchange which do not involve a third party in the exception-less case: the third party is only involved in the presence of exceptions or in the case of dishonest participants who do not follow the protocol. The generic fair exchange protocol is "generic" because different types of items, such as data, signatures, or value (in the rest of this document, we use the more common term "payment," which really means a transfer of value) can be exchanged.

The two-party generic protocol requires at most five messages in the exception-less case. Each party has to sign one message only, which is obviously the minimum for signature-based non-repudiation. The degree of fairness guaranteed by the protocol depends on certain properties of the items to be exchanged: if the third party can undo a transfer of an item (so called revocability; like the "undo" of credit-card based payment systems [BGHH 95]) or if it is able to produce a replacement for it (so called generatability; such as signing a replacement receipt [PWP90]) the protocol achieves true fairness. Otherwise, the third party can only issue affidavits attesting to what happened during the exchange. We call this *weak fairness*, since the parties will have to use the affidavits in an external dispute resolution system, such as a court of law, to achieve fairness. In an environment where most parties do not attempt to cheat, the optimistic approach can provide efficient protocols for most types of fair exchange without creating performance bottlenecks or sacrificing the overall security.

IBM Research Report RZ 2858 (#90806) 09/02/96

In addition, we propose instantiations of the generic protocol for selected items to be exchanged, such as contract signing and certified mail (i.e., a protocol for non-repudiation of delivery and receipt). In a related report [AsSW 96a N. Asokan, M. Schunter, and M. Waidner, Optimistic Protocols for Fair Exchanges, 4th ACM Conference on Computer and Communications Security, Zürich, 1997.

AsSW 96b], an extension to the multi-party case is described in detail.

1.1 Previous Work

The difference between third parties that are actively involved in a protocol and third parties that are used only in case of exceptions was first explained in [DeMe 83]. Bürk and Pfitzmann [BüPf 90] used the latter, optimistic, approach in the fair exchange of money for goods. To the best of our knowledge, the protocol described in this report is the first proposal for fair exchange of *generic* items. Several solutions for specific instances of the fair exchange problem have been proposed previously:

- **certified mail:** fair exchange of a message and possibly a non-repudiation of origin token against a non-repudiation of receipt token,
- **contract signing:** fair exchange of signatures on a contract, and
- **payment with receipt:** fair exchange of a payment for a receipt.

For *certified mail,* most practically relevant protocols are of the same type as those in the ISO documents: they involve a third party even in the exception-less case [e.g., Ford 94, Grim 93, Herd1 95, Herd2 95, ZhGo 96]. In cryptologic protocols for certified mail [Blum 82, Gold 82, BaTy 94], the goal is to achieve fairness *without* a third party, which necessarily implies a probabilistic definition of fairness [EvYa 80]. It is achieved by the *gradual release of secrets* over many rounds: during each round, some knowledge about the message and/or the tokens are revealed. If either party stops before the protocol run is complete, both parties are left with comparable knowledge and, if one assumes comparable computational capabilities, both are able to computationally recover their respective expected items of information (message and/or non-repudiation tokens) to the same extent.

Contract signing without a third party can also be based on the same gradual release of secrets approach [EvGL 85]: the signatures on the contract are released gradually. Assuming that both parties have similar computational capabilities, both parties are able to reconstruct the signed contract to roughly the same extent at any time during a protocol run. Another approach is the *gradual increase of privileges* [BGMR 90] in which the probability that the contract will be deemed valid is increased gradually over several rounds until it is "1" in the last round. This removes the requirement that both parties have similar computational capabilities. A contract signing protocol which is similar to our instantiation of the generic protocol has been proposed by B. Pfitzmann in [Pfit 95].

Due to their gradual approach, cryptographic protocols for certified mail or contract signing are expensive with respect to communication and computation: the knowledge or privilege is increased gradually and the probability of success and the fairness is related to the number of messages exchanged between originator and recipient.

Practical protocols for *payment with receipt* are normally not described as separate protocols which are independent of the payment mechanism used but rather included as receipt mechanisms into specific payment systems [BGHH 95]. In [PWP 90], Pfitzmann *et al.* described a protocol for fair exchange of payment and receipt where the "bank" generates a receipt in case the payee refuses to do so. Bürk and Pfitzmann [BüPf 90] extended this to a protocol for payment for receipt where a third party is only involved in case of an exception. Our protocol can be considered as a generalisation of the protocol of [BüPf 90].

1.2 Overview

After an introduction, Section 2 describes an optimistic protocol for generic fair exchange, discusses the requirements, and evaluates the level of security provided by the protocol — in the most general case, the protocol guarantees “weak” fairness. Section 3 discusses the specific types of items that can be exchanged. Section 4 describes well-known instantiations like contract signing, fair purchase, or certified mail. Section 5 describes efficiency improvements and extensions including a more efficient protocol for an exchange if the parties did agree beforehand on the items to be exchanged.

2. A Generic Protocol for Fair Exchange

2.1 Service Description for Fair Exchange

A two-party exchange exchanges electronic goods between two participants, *O* (for “originator) and *R* (for “recipient”). We consider three types of electronic goods: confidential data, money (payments), and signatures on public data. In order to start an exchange, each party *X* (one of *O* and *R*) has to input the following parameters:

1. $item_X$ the item *X* wants to send[1].
2. $descr_X$ a description of $item_X$, detailed enough to identify all important properties of the item to the person receiving it. For example, the description of contract can be the text of the contract.
3. $expect_X(descr_X, descr_Y)$ a predicate which formalises the expectation of a participant. It evaluates to *true* if the user *X* is satisfied when receiving an item described by $descr_Y$ in exchange for an item described by $descr_X$.
4. *fits*(*descr, item*) a predicate which evaluates to *true* if the description fits the item. This predicate cannot be evaluated automatically for some types of items. For example, a computer can check if the value transferred in a payment is a $20 whereas it is not practical to check if a picture depicts a sunrise. For those types of items whose descriptions cannot be checked automatically, the human user may be prompted whether he likes the item received. Alternatively, if the user discovers a mismatch after the protocol run is completed, he can be allowed to use the evidence generated during the protocol to raise a dispute at a human arbiter.

In Section 3.1, we list possible choices for *descr* and *fits*() for different types of items. The service outputs to *X*

1. $item_Y$ the item *X* has received from the other participant *Y*, and
2. $descr_Y$ a description of its promised properties.

The service also results in some *evidence,* including non-repudiation tokens. The user can retrieve the evidence from the system and use it to prove properties of the exchange to an arbiter. In case of a

[1] The item may also be input at a later stage: for example, a certain party may decide to spend the effort of putting its item together only after the other party has committed to the exchange (or perhaps after actually receiving the item from the other party).

dispute, a dispute protocol is executed between one participant of the exchange in the role of the prover and any other (honest) player in the role of the arbiter: Depending on the exchange protocol and the property to be proven, additional participants in the exchange may also be required to participate in the dispute in the role of witnesses. Input to the dispute protocol are the statement to be proven and the evidence output by the exchange protocol. Example statements that can be proved are:

- A given party sent a given item (Non-repudiation of origin)
- A given party received a given item (Non-repudiation of receipt)
- The complete exchange took place (Non-repudiation of the exchange)
- The parties agreed on what to exchange

2.2 Protocol Description

We propose a generic fair exchange protocol shown in Figure 1 to Figure 3. It exchanges different types of data with non-repudiation of origin and receipt. It is based on asymmetric cryptography, namely, an arbitrary digital signature scheme with the necessary certification infrastructure, a collision-free one-way function *h*(), and a commitment scheme consisting of a procedure *commit*() to commit to an item and *open*() to verify if an opened commitment fits an item. We require from the commitment that

- nobody can change its contents without invalidating it, and
- nobody can get any information about its contents unless the committer explicitly opens it.

We assume that recipients of signatures or outputs of the one-way function check their validity even though we do not depict it in our figures. The protocol is not symmetrical. It guarantees only weak fairness for the originator if no item exchanged is revocable or generatable. Otherwise and for the recipient, strong fairness is guaranteed.

Let *O* denote the originating party that initiates the protocol, *T* the third party that ensures fairness, and *R* the recipient of the initiation message. Each party, *P*, has a pair of public and secret key of a digital signature scheme. For a message *m*, $\mathrm{sign}_P(m)$ denotes the digital signature of *P* computed on *m*. We assume that *m* and a return address (potentially anonymous) of the signer can be retrieved from $sign_P(m)$ in order to allow the third party *T* to contact the signer. This can be achieved in any signature scheme by appending the anonymous address to the text to be signed.

We use a synchronous timing model by assuming that there exist global rounds which include the time needed for transmission and processing of messages. We define an overall maximum time limit, *active-time t*, up to which a run of the protocol can remain active. The state of the run at the end of the active-time is final. We assume that only the connections between each party and *T*is reliable. In practice this can be implemented by a variety of ways:

- choosing a much higher time-out than for other connections, or
- falling back on comparatively more reliable media for communicating with *T* (e.g. from a connection over a packet-switched network, one can imagine falling back to a dial-up connection, and then to a dedicated line), or
- actually “visiting” a real arbiter such as a court.

This would result in three phases: first, the parties try to exchange the items without a third party, then they try a recovery with a third party, and finally, each computer outputs all evidence and any participant may visit a court.

Figure 1 depicts the generic exchange protocol. The basic idea of the protocol is that the originator *O* and the recipient *R* start by promising each other an exchange of items (two flows). If they do not agree on the exchange (e.g., the price of the goods) the protocol is aborted. Otherwise they proceed to exchange the items along with non-repudiation tokens (three flows). Sending certain items (e.g., a payment) may require a sub-protocol containing several messages. Potential involvement of sub-protocols is represented by the use of a thick grey arrow. If no exception occurs, the protocol only consists of these five flows and does not involve *T*. This is the case if *O* and *R* are willing to perform the exchange, and the network is functional[2]. If this is not the case, *O*, *R*, and *T* start an error-recovery phase. Recovery initiated by *O* is depicted in Figure 2. Recovery initiated by *R* is depicted in Figure 3. The initiator of the recovery phase will send *T* the messages of the initial agreement with the other party.

2 This includes the case that any lower-layer error-recovery of the network was successful.

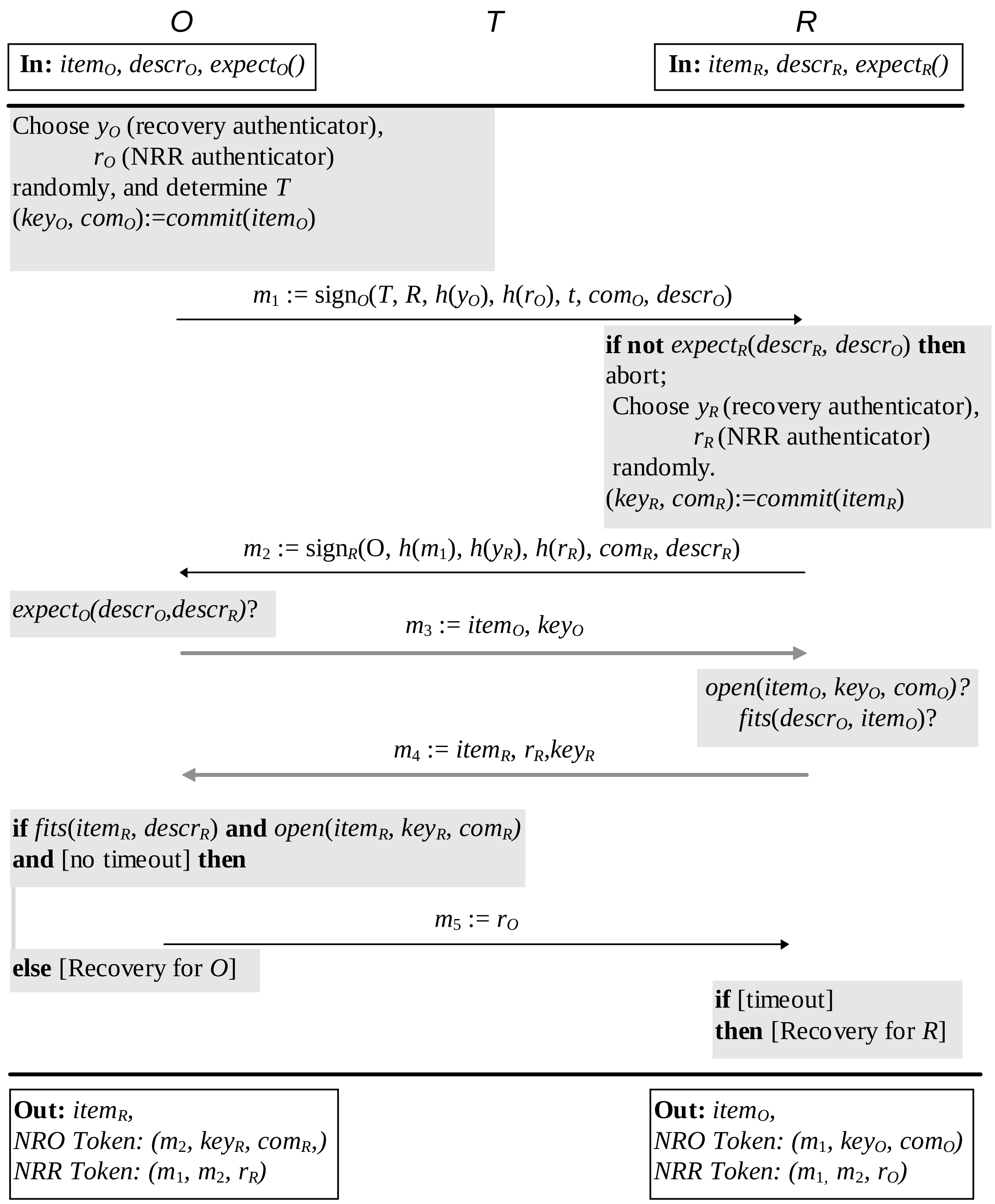

Figure 1 Optimistic Protocol for Exchange with Non-repudiation of Origin and Receipt (NRO and NRR denote non-repudiation of origin and receipt, respectively. Thick arrows denote sub-protocols)

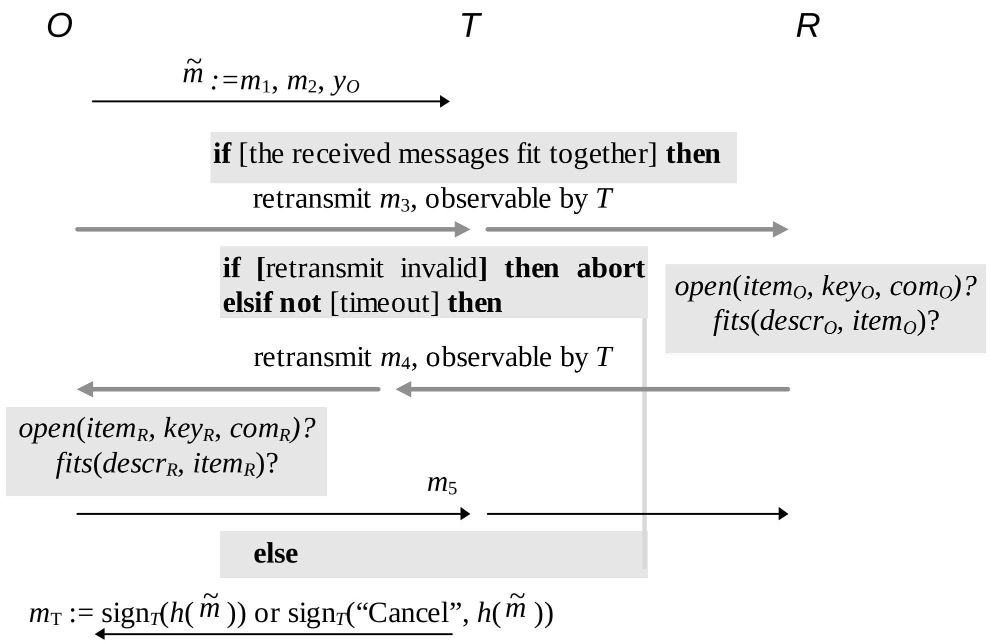

Figure 2 Recovery for *O*

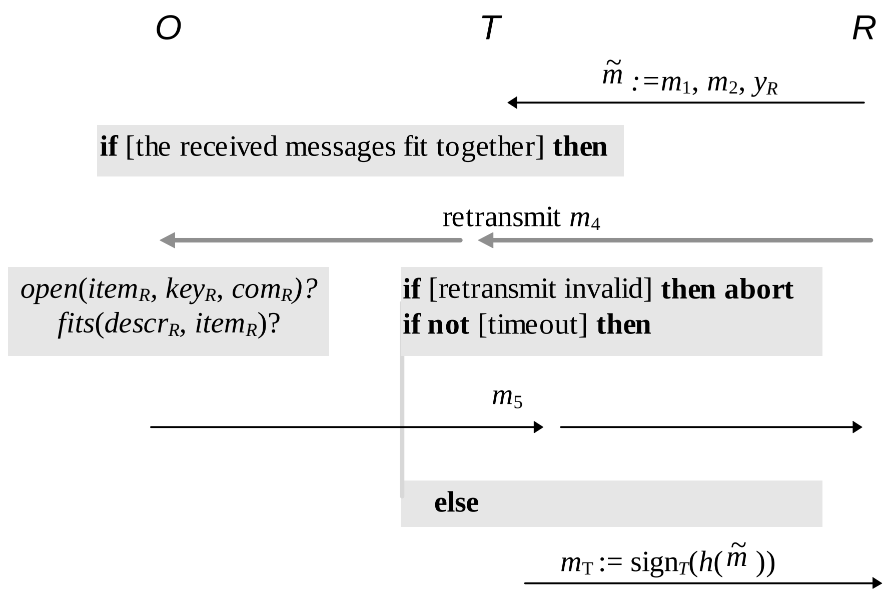

Figure 3 Recovery for *R*

We now describe the protocol depicted in Figure 1 to Figure 3 in detail. To start the protocol, each party inputs the service parameters as described above. Message m_1 fixes *O's* view of the parameters. It contains the following items:

- which third party *T* is to be used in case of an exception,
- an address of the recipient *R*,
- two commitments to the random values y_O, and r_O in the form of images of the one-way function $h()$,
- the active-time limit *t* (see Section 2.3),
- the description $descr_O$ of the item, and

- a commitment com_O to the item computed using the *commit()* procedure of a cryptographic string commitment scheme, where possible.

The commitments to the random values are used to save signatures by committing to a value x with one signature and later releasing it to authenticate an additional message. Naturally, these authentications can also be replaced by signing the messages with any given signature mechanismThis enables the protocol to produce non-repudiation tokens in a given format signed with a given signature system. In the protocol, y_O can later be used to signal in a non-reputable way that the third party T is to become involved, r_O for non-repudiation of receipt (NRR) to signal that O received $item_R$, respectively.

The commitment com_O is used to provide non-repudiation of origin (NRO) for the item. If the item is "intangible" (e.g. a payment), it is not possible to construct a commitment to it. However, the sub-protocol used for sending such an intangible item may itself provide an NRO token, making it unnecessary to provide a separate NRO token. If an NRO token is still necessary, one can leave the commitment empty, i.e., just fix the description and authenticate the non-repudiation of an item matching it by releasing key_O. Whenever the transfer of an item in a round (e.g. m_3) involves a sub-protocol, the additional information necessary for the NRO tokenis sent in an additional message.

If R does not agree with the exchange parameters after having received m_1 from O, it aborts. If it agrees, it sends m_2 containing a commitment to the item to be sent together with its description and some commitments to random values. With m_2, R acknowledges that it will send its item after having received m_3 containing the item it expects. Again, three pre-images are fixed for the same purposes as in m_1. If R sends m_2, both parties have agreed on the exchange and the protocol continues.

O sends its item, and opens the commitment by sending its key. R checks that the commitment contained this item and checks if the description fits. R then sends its item and pre-image for the NRR token together with its key to open the commitment. If O does not receive the message or if the item does not fit the commitment or its description, it starts its recovery procedure (Figure 2). Otherwise, it sends the pre-image for the NRR token. If this pre-image is not received by R, R starts its recovery procedure (Figure 3). If no fault occurred, both participants store their items and non-repudiation tokens and the protocol ends.

Recovery for O includes the following steps: In case O does not receive what it expects in m_4, it sends a message $\tilde{m}$ containing the initial agreement to T and authenticates the wish to involve T by revealing y_O. T checks the message and then provides a reliable channel between O and R via T through which O can replay m_3 to R as a first attempt (how to replay sub-protocols is examined in more detail in Section 2.7). R is then expected to reply with m_4. If the item in the replay of m_3 fits the description or the commitment and R nevertheless does not reply, T is convinced that R does not follow the protocol since we assumed that the network connection between T and R is reliable. It can therefore issue an *affidavit* m_T in the form of a signed statement certifying that all the messages and items fixed in $\tilde{m}$ were actually sent to R within the specified time (note that all messages in the protocol, including the affidavit, are implicitly tied to the timestamp t which is included in m_1). It is presumed that the affidavit can thereafter be used as evidence or to initiate revocation or replacement of an item. If R does reply with message m_4 to T, T can forward it to O. The protocol can then continue or R can ask T for message m_5 constituting the NRR token for R together with the messages of the initial agreement.

In case R does not receive m_5 after having sent m_4, it can engage in a similar recovery. Due to the asymmetry inherent in the protocol, T can in fact provide R with a strong fairness guarantee: R never sends the item it promised unless it has already received the item promised to it; Also, *T* can generate a replacement for a NRR token on behalf of *O* if *O* did not respond during recovery for *R*.

It is useful to identify when a protocol run is considered "completed." From the point of view of a party *P*, if a run of the protocol outputs the expected items (and non-repudiation tokens), then the protocol run is considered completed for *P*. The items already output to an honest party at the completion of a protocol run will not be invalidated. If the other party *Q* initiates a recovery afterwards, then the messages *P* has to send during this recovery is not part of the earlier protocol run anymore (it is just a replay of some message flows from the earlier run). At the end of active-time limit, the protocol is definitely completed for all parties.

2.3 Time-Outs

The only critical time-out of the generic protocol in Section 2 we have mentioned so far is the *active-time* limit *t* specifying the absolute time at *T* when the protocol ends. This time-limit ensures a consistent view of all honest participants: The state at time *t* is not changed afterwards: after this time everybody (and *R* in particular) will be sure that the status of an exchange is definitely final and will not be changed anymore. We express *t* in terms of the local clock at *T* since *T* is the only entity that makes decisions based on the active-time limit in a way that has an impact on the correctness of the protocol from the point of view of *other* entities: if *T* will not accept recovery requests after a certain time *t'*, i.e., if *T* decides that a recovery request came too late, no fairness may be provided to the party requesting recovery. In practice however, both *O* and *R* have to know the time on *T*'s clock in order to agree on the active-time limit as well as to compute local time-outs within rounds. Hence, we require a model in which clocks of all parties are synchronised (i.e., all parties have real-time clocks, and the differences between all local clocks of honest parties are limited by a constant).

To allow the parties to determine a reasonable active time, each party in the role of *T* will announce an estimated turn-around time t_T within which it will process exception requests from other parties.. *T* will also have a policy p_T, expressed as a function of *t* (variable, chosen by the parties of an exchange) and t_T (constant, chosen by *T*) which indicates the time after which *T* will not accept exception-handling requests from *O* or *R*. For example, p_T may be $t\text{-}2t_T$.. All pending exceptions must be processed by time *t*.

In addition to these, each party has to decide on local time-outs after sending out *critical messages*. A critical message is one such that if it is sent, an appropriate response must be obtained or, if such a response does not arrive, some alternate action must be taken instead of simply abandoning the protocol run. In the case of *O*, m_3 is a critical message. In the case of *R*, m_4 is critical. In the case of *T*, the retransmission of the goods sent by *O* or *R* to each other via *T* are critical messages.

When *O* sends out m_3, it will start a local timer to determine when it should invoke *T* by sending $\tilde{m}$. The value τ_O of this time-out should be computed based on several factors: the overall active-time limit that was agreed upon earlier, the time that has passed since the protocol run has begun, and possibly expected network latency and processing delay at *R*'s end. The exact computation can be at best based on some rules of thumb. *R* has a similar time-out τ_R. For example, if *O* sends out the critical message m_3 at time instant *t'*, and it estimates that the expected communication delay between it and *T* to be t_{OT}, then the estimate for τ_O will be $p_T(t,t_T)\text{-}\ t_{OT}$. If *O* prefers to use a safety factor *s* in its estimate, τ_O becomes $t' + (1\text{-}s)(\ p_T(t,t_T)\text{-}\ t_{OT} - t')$.

Similarly, *T* has to decide on a time-out τ_T value for the period starting from the instant m_3 was replayed via *T* to *R* to the instant when *T* decides to issue an affidavit.

In general, every protocol step that is based on whether a response was received or not (the [time-out] conditions in the protocol pictures), a specific time-out value needs to be computed.

2.4 Requirements

We now give the requirements for the originator *O*. The requirements for the recipient *R* can be obtained by exchanging *O* and *R*. For each requirement, we first list the set of parties which are assumed to be honest and are expected behave correctly (A party is considered to misbehave if it does not respond to a critical message that is valid):

I. **Unforgeability of Non-repudiation Tokens**
   A. If *O* and *T* are honest, nobody other than *O* can create a valid non-repudiation token of *O*.

II. **The Role of the Third Party**.
   A. If *T* and *O* are honest, *T* does not create affidavits in the name of *O*.
   B. If *T* and *O* are honest, *T* creates affidavits in the name of *R*, if *R* does not behave correctly.

III. **No Unconditional Trust in the Third Party**
   A. If *O* is honest, no non-repudiation token or affidavit can be produced by *T* without *O's* part of the initial agreement.

IV. **Meaning of Non-repudiation Tokens**
   A. If an arbiter *A*, *T*, and *O* are honest and a non-repudiation of origin or receipt token for an item is output to *O*, then *O* can convince *A* that *R* sent or received the item, respectively.
   B. If an arbiter *A* is honest and no non-repudiation of origin (or receipt) token for an item is output to *R*, then *R* cannot convince *A* that *O* sent (or received) the item

V. **Weak Fairness of Exchange**
   A. If *T* and *O* are honest and if *O* does not receive *everything* necessary to satisfy its expectations, namely
      - NR tokens,
      - the committed item or an affidavit from *T*

   then *R* does not get *any* of
      - any additional knowledge about the item sent by *O* except its description,
      - or a NR-token,
      - or an affidavit.

VI. **Strong Fairness of Exchange**
   A. Strong fairness is the same as weak fairness except that an affidavit does not satisfy the expectations.

In Section 2.5, we will argue in an informal manner that our protocol meets the requirements of weak fairness for *O* and strong fairness for *R*. If the item promised by *O* is revocable or that promised by *R* is generatable by *T*, strong fairness can be achieved for both participants.

2.5 Security

Now, we describe informally why our protocol meets the requirements listed in the previous section.

Unforgeability of non-repudiation tokens follows from the assumptions that:
- The signature scheme is secure (this implies security of certification, too), and
- the item cannot be changed without invalidating the commitments.

The first two requirements on **the role of the third party** (*T*) state that *T* will not create affidavits and replacement items in the name of a correctly behaving party but *can* do so in the name of an incorrectly behaving party. When *T* is invoked it first checks to see if the party invoking *T* did in fact send out a critical, valid message. For example, if *O* invokes *T*, *T* first checks to see if the commitment messages (m_1 and m_2) are in order, linked by the inclusion of $h(m_1)$ into m_2, and that the complaint is about a critical message of *O*, namely m_3. If m_2 is valid, then only *R* could have created it given our assumptions about the security of the digital signature scheme. Therefore, if *T* decides to replay m_3 to *R*, then *R* must have committed to the protocol. Since the channel between *R* and *T* is assumed to be reliable, *R* is guaranteed to receive *T*'s replaying of m_3. Thus, once *R* receives the message containing the valid item, all of its expectations must have been met. If *R* is behaving correctly, it can reply with m_4. and *T* will not send an affidavit (or a replacement item) in the name of *R*. *T* generates replacements only if it receives no response from *R*; but since we assumed reliable communication this happens only when *R* is misbehaving.

If *R* invokes *T*, *T* can check that m_1 and m_2 are in order and relay m_4 to *O*. At this point, all of *O*'s expectations must have been met. Therefore, if *O* does not release r_O to complete the NRR token, *T* can issue a replacement NRR token to *R* since it is clear that *O* does not behave correctly.

No unconditional trust in the third party *T* is required since both messages m_1 and m_2 containing the name of *T* must be included in any valid non-repudiation token or affidavit issued by *T*, i.e., if the party never successfully participated in an initial agreement, no valid token or affidavit can be produced by any party.

The intended **meaning of the non-repudiation tokens** follows from the facts that:
- non-repudiation tokens are unforgeable,
- a replacement token issued by *T* on behalf of *O* or *R* is equivalent to a token issued by *O* or *R* respectively, and
- a judge will use the same "test" for the validity of non-repudiation tokens that a recipient of the token applies during the course of the protocol.

Weak fairness of exchange for *O* follows from the fact that if *O* does not receive everything it expects, then either *O* did not send out m_3 (in which case only the description of the item has been revealed to *R*) or if message m_3 was sent without receiving the expected item and the NR-tokens, then *T* issues an affidavit. In both cases, *O*'s requirements are met.

On the other hand, assume that *R* received an affidavit instead, *T* was required to replay all expected items to *O* through the reliable channel provided by *T* before issuing the affidavit. This is a contradiction to our assumption that *T* is honest and *O* did not receive everything it expected.

Strong fairness of the exchange for *R* follows from the facts that:
- *R* never releases the item it promised unless it has received the item it wants along with the NRO token for it, and

- if *O* fails to release the pre-image necessary to complete its NRR token (r_O), *T* will provide a replacement token to *R* according to the requirements on the role of *T*.

2.6 Weak vs. Strong Fairness

During the analysis of the protocol, we stated that weak fairness is provided to *O*, whereas strong fairness is always provided to *R*. However, strong fairness can be provided to both parties, if at least one item can be revoked or if *T* can replace it without cooperation of its sender, i.e., the affidavit issued by *T* can be used to

- *revoke* or *cancel* the item already sent by *O* if it is *revocable*.
- *generate* a replacement for the item promised by *R* if it is *generatable*.

If only one of the items has one of these properties, one must take this asymmetry into account in deciding which party in the fair exchange plays the role of the originator *O*: if the participant sending a revocable item acts as the originator or if the participant sending a generatable item acts as recipient, strong fairness is guaranteed by our protocol. If both items are neither generatable nor revocable, we can only guarantee weak fairness and one may therefore rather use an exchange protocol with an on-line third party.

Revocability can be achieved in cooperation with the bank for most payment systems: for example, using a credit card payment system with cancellation or using two-showable coins [BüPf 90, PWP 90, Jako1 95]. It is not practical if the non-repudiation tokens have a meaning outside the protocol (e.g., so called “public data.” See Section 3). Both participants would be required to participate in an arbitration, since an issued token may have been revoked. Generatability or revocability can be added to confidential data by depositing the data at a third party which automatically releases it after the active time of the protocol. By showing an affidavit, this release can be prevented. Similarly, to add generatability, this party will only release the data if a proper affidavit is shown to it.

2.7 Transfers Involving Sub-Protocols

Sending certain items such as payments may involve sub-protocols. When *T* is invoked after an exchange *T* must be convinced that the receiver really got the item before issuing affidavits. In order to convince *T* there are several possibilities:

- the item can be sent to *T* who checks it and sends a similar item to the receiver,
- the protocol can be re-run while all messages are sent via *T*, or
- the protocol may have the following properties:
  - they are atomic: in case of interruptions they either recover to complete the protocol run or roll back to the state before starting it, and
  - they have the ability to produce evidence that proves that a protocol run did in fact complete.

  We call such protocols *well-defined*. If a sub-protocol is well-defined, then a party using it in an exchange will need to invoke *T* only when it has proof of protocol run completion that can be shown to *T*. To handle the exception, *T* makes sure that the proof is valid, show it to the other party. If the other party does not oblige, *T* issues an affidavit. In other words, there is no need to replay the protocol run.

Note, that any protocol where *T* can check if the item was transferred given the transcript of all messages can be extended to a well-defined protocol by sending critical protocol messages with non-repudiation. However, this will not be possible for arbitrary protocols without extending them. Counterexamples include protocols where messages are encrypted with the recipient's public key and the corresponding private key is not known to *T*.

The problem of enabling *T* to verify sub-protocol is not specific to optimistic exchanges: During an on-line-exchange *T* also needs to be able to check what has been transferred. However, the requirements on the sub-protocol for the optimistic approach are slightly stronger than for an on-line arbitration since it requires two "tries" for transferring the item: after trying to send an item directly, on-line arbitration must still be possible to enable recovery by *T*.

3. Exchangeable Items

We now describe the items which can be "plugged" into the generic protocol and the resulting exchanges.

3.1 Types of Items

In the generic fair exchange protocol described in Section 2, we used *item* and *descr* to represent the real data to be exchanged. We now describe different data types to be exchanged; namely public data, confidential data, and payments:

- *confidential data:* some data which will be released during the protocol described by an optional text, examples include digital goods and messages,
- *public data*[3]*:* data which may be released even if the protocol execution has not been successful, for example information which has already been known to both communication partners, like contracts, and
- *payments:* a payment protocol is executed to transfer a value from payer to payee.

Each type has specific descriptions. A summary is given in Table 1. Note, that in all cases a participant receives non-repudiation tokens.

	conf. data	**public data**	**payment**
item	*data*	*data*	payment of *amount* to *payee*
descr	*text*	*data*	*amount, payee*
expect() checks:	*text*	*data*	*amount, payee*
fits()	*may ask user*	*descr=item?*	*query the payment system used.*

Table 1 Different Types of Items

Confidential data is some data which must not be released without receiving the item to be exchanged for it. It may be valuable data, such as computer software or just certified mail. If the recipient of confidential data has certain expectations, such as for images or programs, the protocol must check if these expectations are met. Since the data itself cannot be checked, one needs additional information to verify this agreement on the exchange. Therefore the initial agreement fixes a description to enable the

[3] This is the special case of confidential data, with only a description.

recipient to check if it agrees on the description of the item to be received. However, the sender may still send data which does not fit its description. As countermeasures, the *fits*() predicate may be evaluated interactively. In any case the parties may later dispute non-electronically if the data fits the description at a human arbiter.

To illustrate the distinction between description and data, we consider a fair purchase of computer software. The buyer would like to buy a text processor. The buyer inputs a description like “Name, Version, Year, Word Processor for OS/2, Number of kB, provides at least the following features: …” which he has received in the offer from the seller. During the fair purchase the protocol compares this text input by the buyer with the text signed by the seller together with the commitment on the program data. If the descriptive texts are not equal, the buyer aborts. Later, the buyer checks the program and if the program does not execute under OS/2, he may invoke an arbiter which may decide on the dispute.

Public data is some data where the only purpose of the protocol is the fair exchange of non-repudiation tokens for it. The data itself is either known to both parties or may be released even in the presence of faults. Examples are contracts, the text of receipts or binding descriptions of confidential data. Note that even if the exchanged public data is empty (e.g., in exchange for confidential data during certified mail), a time-stamp and non-repudiation tokens are generated nevertheless.

A payment is the transfer of value from one party to the other. Depending on the type of payment system used, payments are revocable, i.e., during a certain time, the third party is able to cancel the payment, or generatable, i.e., the bank may enforce a bank transfer given the amount and the accounts of payer and payee.

3.2 Exchanges

The resulting exchanges are listed in Table 2. The efficiency improvements are mainly based on the omission of obsolete messages depending on the minimal service needed. The data types described above can be plugged into the generic exchange protocol. Some optimised protocols for the resulting possible exchanges are identified in Table 2. The timestamp *t* is directly or indirectly included in all messages of the protocol. Therefore, using a timestamp in an exchange is effectively the same as using an empty item.

	public data	**conf. data**	**payment**
public data	contract signing	certified mail	payment with receipt
conf. data		exchange of goods	fair purchase
payment			currency exchange

Table 2 Examples of Exchanges

4. Optimised Protocols for Fair Exchanges

We now give optimised protocols for specific instantiations of the generic protocol.

4.1 Certified Mail

This is the problem of sending an electronic mail and being able to prove the receipt of the mail to third parties afterwards. This is one of the services provided by existing non-repudiation standards [ISO1, ISO2, ISO3]. Since the *Sender* knows what to receive, we use an instantiation of the 4-pass protocol depicted in Figure 8.

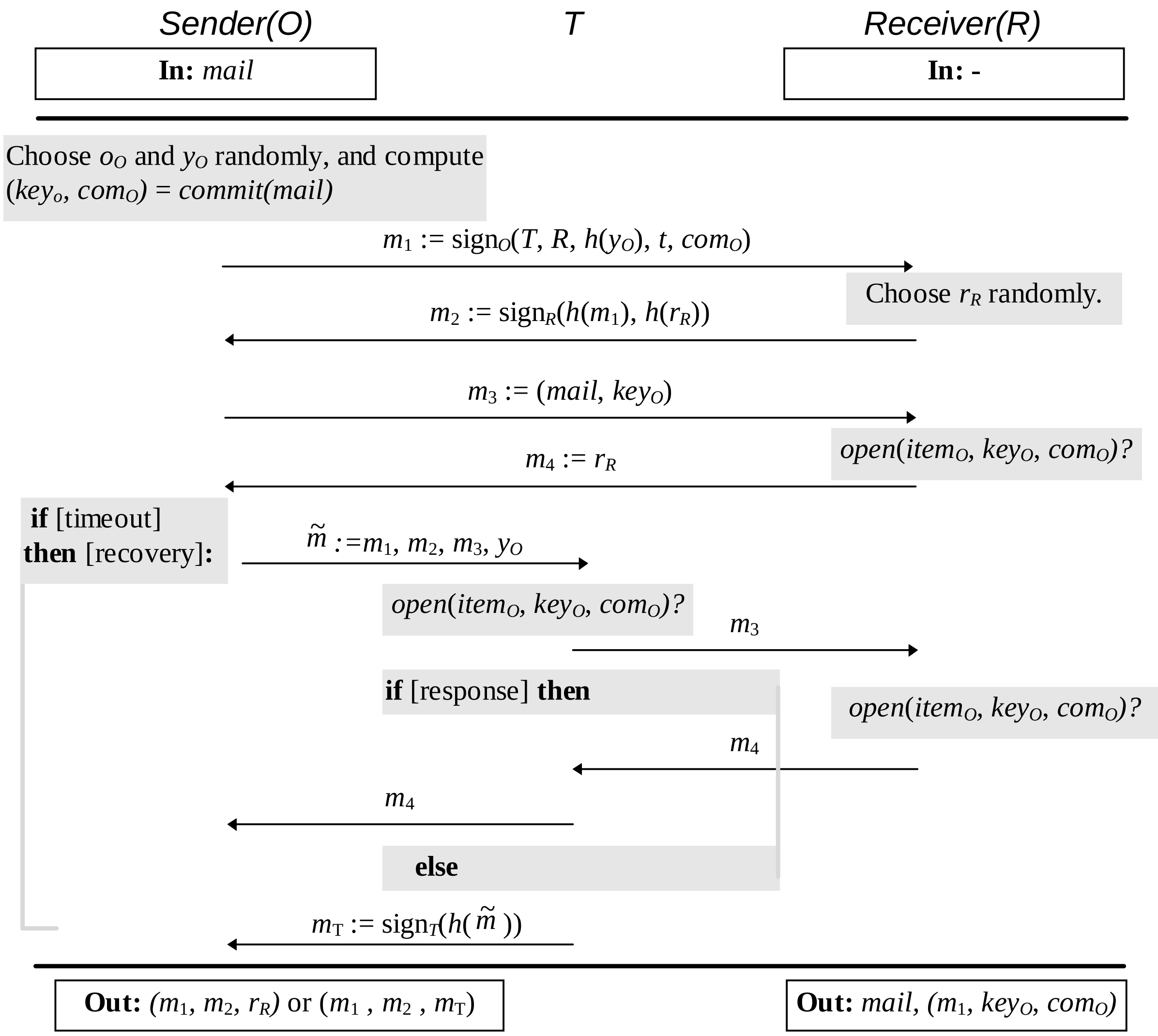

Figure 4 An Optimistic Protocol Certified Mail with Non-repudiation of Origin and Receipt.

4.2 Contract Signing

This is the problem of a fair exchange of signatures by *O* and *R* on a contract *contract* [BGMR 90, Blum 82, EvGL 85, Rabi 83, Pfit 95]. In principle, our solution for contract signing is an exchange of non-repudiation tokens after an initial agreement: the tokens exchanged are reinterpreted as signatures on the contract.

Figure 5 shows a protocol for contract signing with only three instead of four messages in the exception-less case.

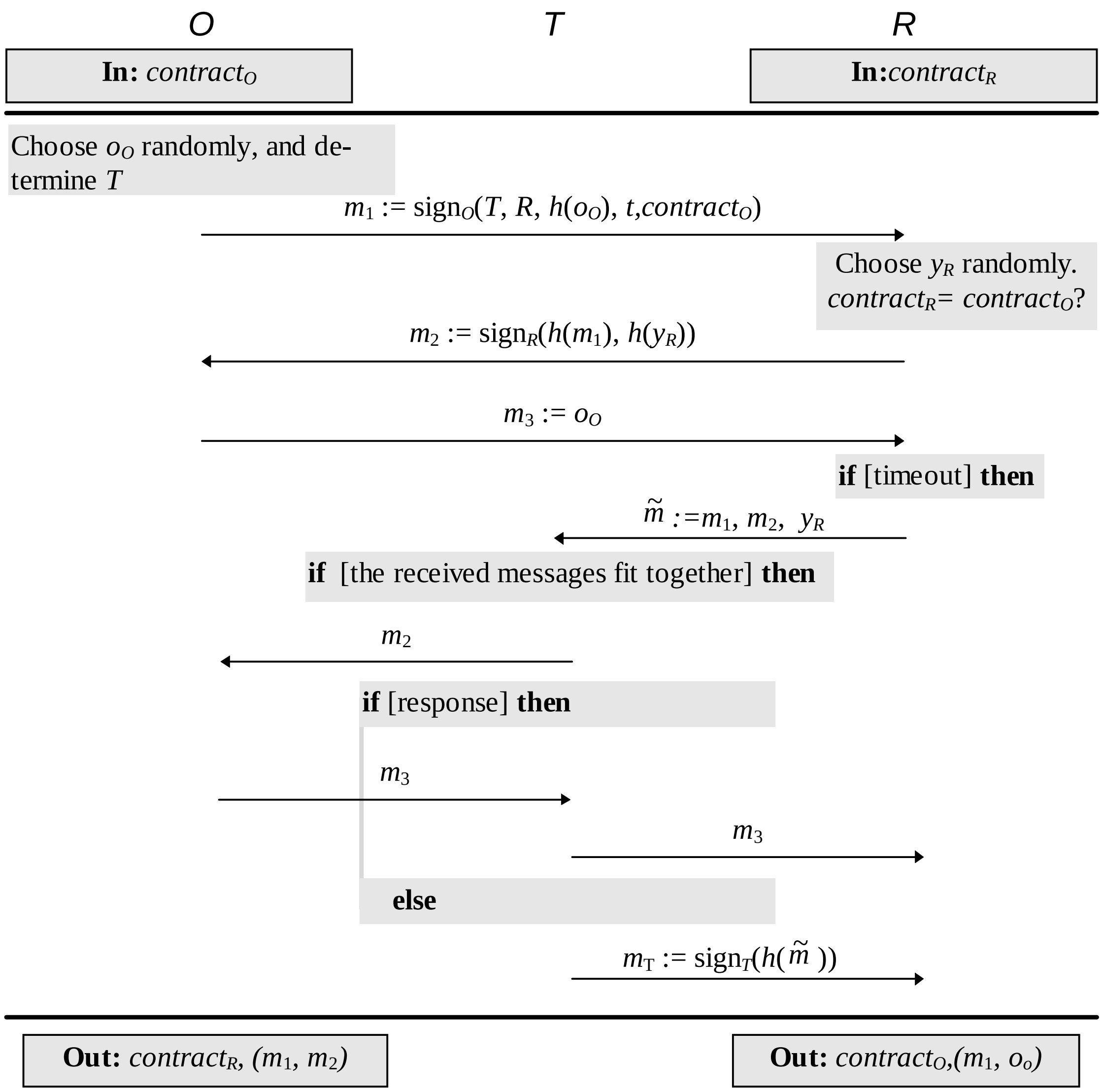

Figure 5 An Optimistic Protocol for Contract Signing.

4.3 Payment for Receipt

Payment for receipt is the problem of a fair exchange of an electronic payment and an electronic receipt. In our classification, it is an exchange of payment and public data. Note that an electronic receipt is different from a paper receipt: An important property of paper-receipts is that they can only be used once (e.g., for a tax refund) which is not the case for electronic receipts. The instantiation of the generic protocol is similar to the protocol proposed in [BüPf 90].

Note that T will not send a replacement receipt to R if:

- T can, by itself, check the validity of the payment in m_2 and determines that it is not valid, or

- O can convince T that the payment in m_2 was invalid (e.g., by producing a statement from a bank)

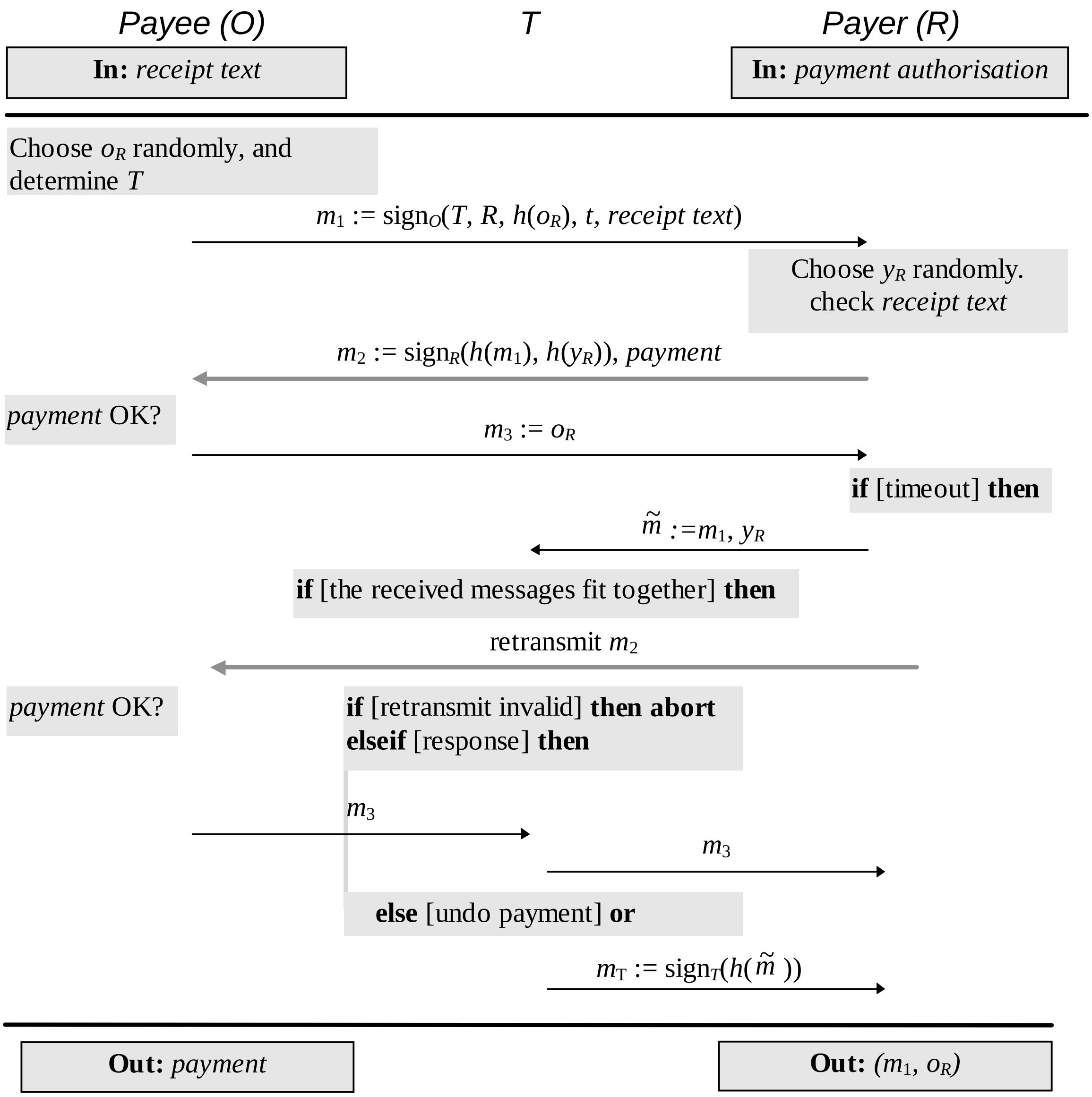

Figure 6 An Optimistic Protocol for Payment with Receipt.

4.4 Fair Purchase

Fair purchase is the problem of a fair exchange of payment for on-line delivery of goods such as the result of a database query or a program. In the protocol, the first two messages define the goods and price. The third message is the payment from O to R. The fourth message is the delivery of the goods. If the goods are not delivered in time, O would resend the payment via T which would ask R to resend the goods. If R does not do so within some time, T will issue an affidavit which can be used to undo the payment.

Using the mechanism for fair exchange of sequences of items outlined in Section 0, this fair purchase protocol can be extended to a scenario involving repeated payments.

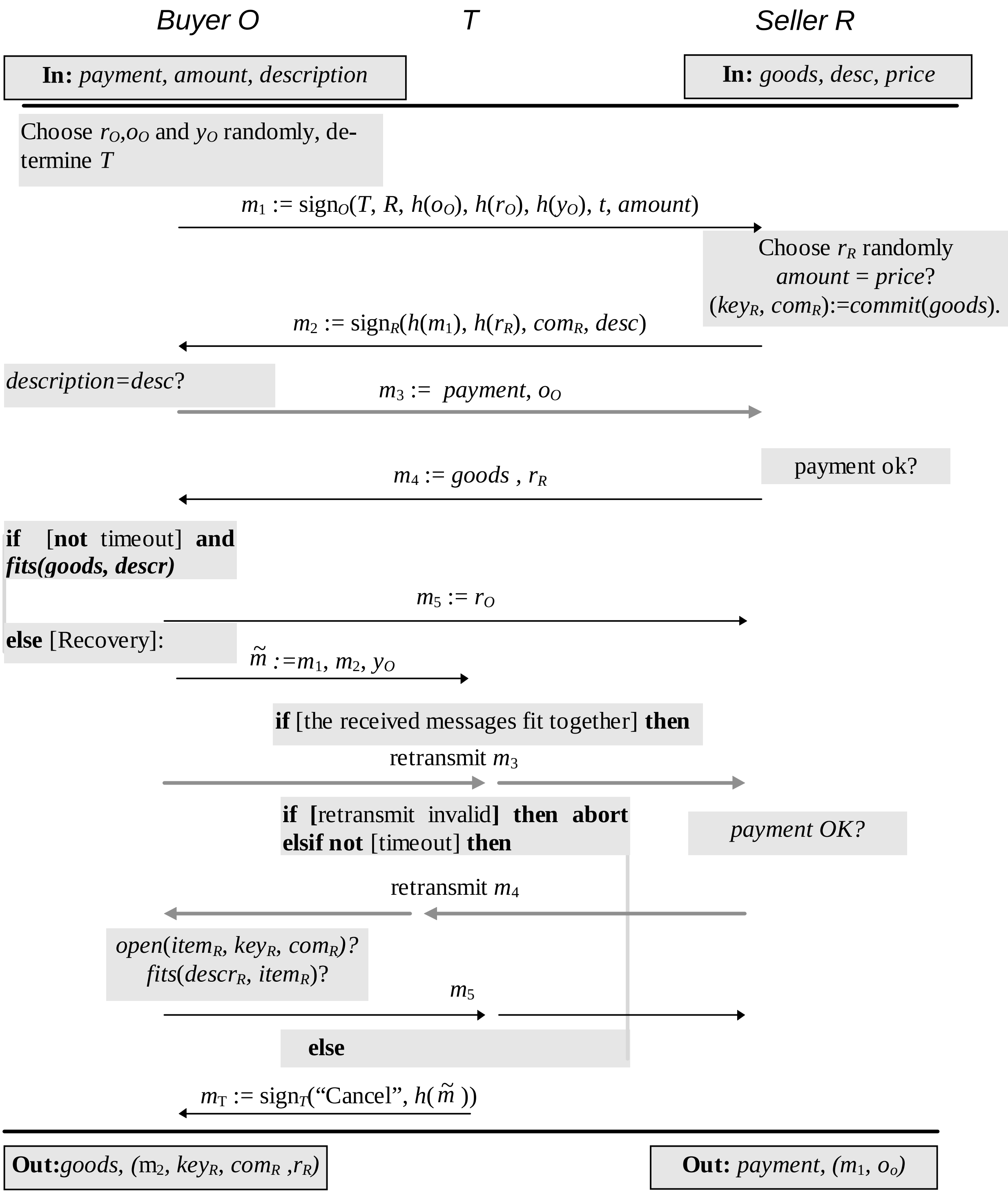

Figure 7 An Optimistic Protocol for Fair Purchase based on Payments on Hold.

5. Extensions and Variations

5.1 Four-Pass Protocol for a Fixed Exchange

In our generic protocol, for fixing the agreement, two messages are necessary. However, in general a four-pass protocol is applicable if the originator *O* knows the description of the item that it will receive. In this case, *O* proposes a fixed exchange and *R* acknowledges the agreement by sending the requested item after which *O* again sends the non-repudiation of receipt token. The resulting four-pass protocol is shown in Figure 8.

An alternative to our five-pass protocol described in Section 2 can be constructed from this four-pass protocol by *R* sending $descr_R$ to *O* in an initial message. However, the resulting protocol would be less robust since any party could start the exchange protocol which may reduce *O*'s availability.

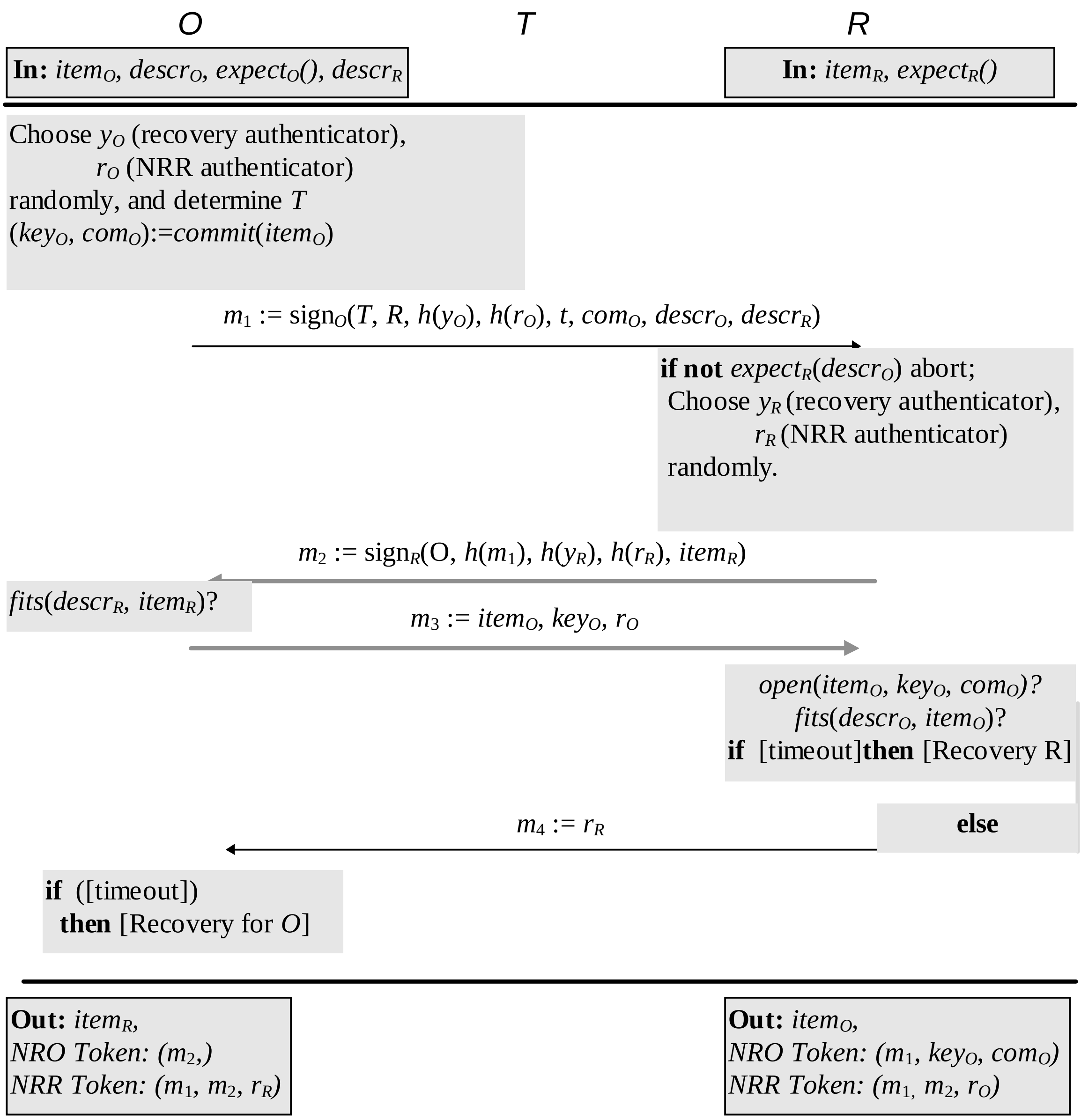

Figure 8 Optimistic four-pass Protocol for Exchange with Non-repudiation of Origin and Receipt.

5.2 Status Requests for the Receiver

If one likes to tolerate network problems between *R* and *T*, one may enable *R* to ask *T* if a started exchange has been cancelled. If *R* sends m_2 in the generic protocol depicted in Figure 1 but does not receive m_3 this extension enables *R* to contact *T* and ask for m_3. It should be included into any practical implementation of these protocols, since robustness should be provided and establishing the assumed reliable channel to *T* may not always be possible given an absolute time limit. In addition, one party may be off-line or defunct for a certain time and nevertheless wants to know the final result of the exchange (Note that this is some kind of protection for dishonest parties).

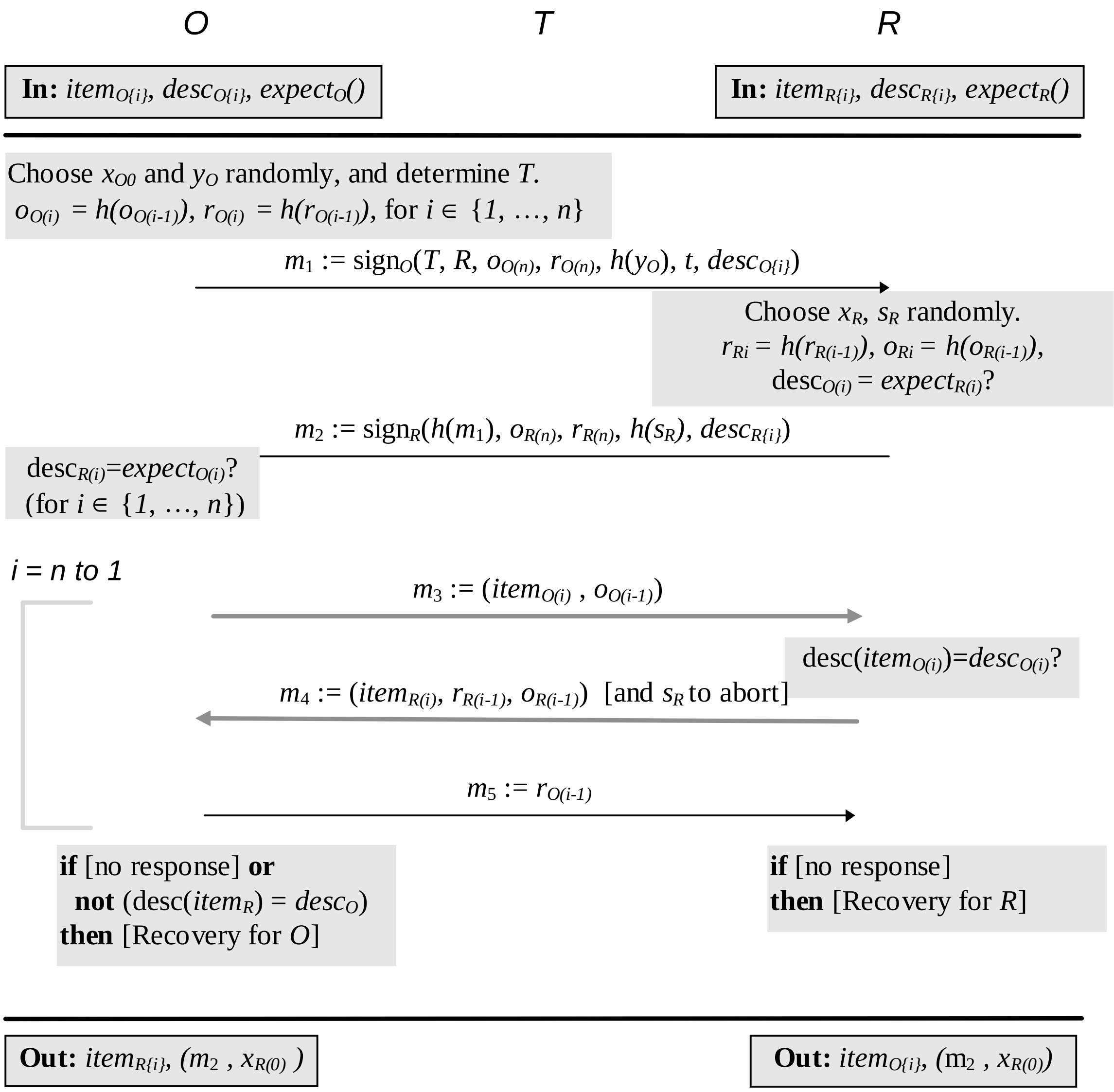

Figure 9 An Optimistic Protocol for Sequences of Items.

5.3 Sequences of Items

In some applications, sequences of items are exchanged. Examples are many web-pages for small amounts of money or a document where each individual page should be acknowledged by a NRT before the next page is sent.

The generic protocol can be extended to provide exchanges of a sequence of items by using a mechanism used for tick-payments [Pede 95]. During the initial messages each party fixes the last image of repeated applications of a collision-free one-way function *h()*. During each exchange round of the protocol a pre-image of the image sent during the last round is revealed and thus produces a sequence of non-repudiation tokens (the initial agreement is defined to be the first round). If *O* wants to terminate the exchange, it stops sending items. If *R* wants to terminate, it releases a pre-image fixed in the second

initialisation message. Note that if the individual items are of low value, *O* may not invoke the third party to recover an unsuccessful round. In this case, no third party is needed at all and the protocol is similar to the cryptographic exchange protocol by gradual release.
Figure 9 depicts the protocol.

6. Acknowledgements

We thank Birgit Baum-Waidner, Birgit Pfitzmann, and Michael Steiner for valuable comments and discussions. This work was partially supported by the European Commission and the Swiss Federal Department for Education and Science in the context of the ACTS Project AC026, *SEMPER*; however, it represents the view of the authors only. *SEMPER* is part of the *Advanced Communication Technologies and Services* (ACTS) research program established by the *European Commission Directorate General XIII*. For more information on *SEMPER*, see *http://www.semper.org/*.

7. References

AsSW 96a N. Asokan, M. Schunter, and M. Waidner, Optimistic Protocols for Fair Exchanges, 4th ACM Conference on Computer and Communications Security, Zürich, 1997.

AsSW 96b N. Asokan, M. Schunter, and M. Waidner, Optimistic Protocols for Multi-Party Fair Exchanges, IBM Research Report RZ 2892, IBM Zurich Research Laboratory, Zürich,1996.

BaTy 94 Alireza Bahreman, J. D. Tygar: Certified Electronic Mail; Proc. Symposium on Network and Distributed Systems Security, Internet Society, February 1994, 3-19.

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; Proc. First USENIX Workshop on Electronic Commerce, New York, July 1995.

BGMR 90 M. Ben-Or, O. Goldreich, S. Micali, R. L. Rivest: A Fair Protocol for Signing Contracts; IEEE Transactions on Information Theory 36/1 (1990) 40-46.

Blum 82 Manuel Blum: Coin Flipping by Telephone - A Protocol for Solving Impossible Problems; digest of papers compcon spring 1982, February 22-25, 133-137.

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

DeMe 83 Richard DeMillo, Michael Merritt: Protocols for Data Security; Computer 16/2 (1983) 39-51.

EvGL 85 Shimon Even, Oded Goldreich, Abraham Lempel: A Randomized Protocol for Signing Contracts; Communications of the ACM 28/6 (1985) 637-647.

EvYa 80 Shimon Even, Yacov Yacobi: Relations Among Public Key Signature Systems; Technical Report no. 175, March 1980, Computer Science Department, Technion, Haifa, Israel.

FIPS 180 Secure Hash Standard; Federal Information Processing Standards Publication 180 (FIPS PUB 180), May 11, 1993.

Ford 94 Warwick Ford: Computer Communications Security — Principles, Standard Protocols and Techniques; PTR Prentice Hall, Englewood Cliffs, New Jersey 1994

Gold 82 Oded Goldreich: A Protocol for Sending Certified Mail; Technion - Israel Institute of Technology, Computer Science Department, Technical Report, 1982.

Grim 93 Rüdiger Grimm: Non-Repudiation in Open Telecooperation; 16th National Computer Security Conference, September 20-23, 1993, Baltimore Convention Center, Baltimore, Maryland, 16-30.

Herd1 95 S. Herda: Nichtabstreitbarkeit (Non-repudiation): Stand der Standardisierung; Trust Center, Grundlagen, Rechtliche Aspekte, Standardisierung, Realisierung, DuD Fachbeiträge, Vieweg, Wiesbaden 1995, 271-282.

Herd2 95 Siegfried Herda: Non-repudiation: Constituting evidence and proof in digital cooperation; Computer Standards & Interfaces 17 (1995) 69-79.

ISO1 ISO/IEC JTC 1/SC 27 N 1105, 2nd ISO/IEC CD 13888-1, Information technology - Security techniques - Non-repudiation - Part 1: *General Model*

ISO2 ISO/IEC JTC 1/SC 27 N 1106, 2nd ISO/IEC CD 13888-2, Information technology - Security techniques - Non-repudiation - Part 2: *Using symmetric encipherment algorithms*

ISO3 ISO/IEC JTC 1/SC 27 N 1107, ISO/IEC CD 13888-3, Information technology - Security techniques - Non-repudiation - Part 3: *Using asymmetric techniques*

Jako1 95 Markus Jakobsson: Ripping Coins for a Fair Exchange; Eurocrypt '95, LNCS 921, Springer-Verlag, Berlin 1995, 220-230.

Rabi 83 Michael O. Rabin: Transaction Protection by Beacons; Journal of Computer and System Sciences 27/ (1983) 256-267.

Pede 95 Torben P. Pedersen: Electronic Payments of Small Amounts; DAIMI PB-495, Computer Science Department Aarhus University, August 1995.

Pfit_95 Birgit Pfitzmann: Contract Signing, Unpublished Manuscript, Uni Hildesheim, 06/08/95.

PWP 90 Birgit Pfitzmann, Michael Waidner, Andreas Pfitzmann: Rechtssicherheit trotz Anonymität in offenen digitalen Systemen; Datenschutz und Datensicherung DuD 14/5-6 (1990) 243-253, 305-315.

Tedr 85 Tom Tedrick: Fair Exchange of Secrets (extended abstract); Crypto '84, LNCS 196, Springer-Verlag, Berlin 1985, 434-438.

Schn 96 Bruce Schneier: Applied Cryptography (2nd ed.), John Wiley & Sons, New York, 1996.

ZhGo 96 Jianying Zhou, Dieter Gollmann: A Fair Non-repudiation Protocol, 1996 IEEE Symposium on Research in Security and Privacy, IEEE Computer Society Press, Oakland 1996, 55-61.