RZ 2892 29/11/96
Computer Science/Mathematics 16 pages

Research Report

Optimistic Protocols for Multi-Party Fair Exchange

N. Asokan

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

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

Optimistic Protocols for Multi-Party Fair Exchange

11/29/96

Abstract: We describe a generic protocol for fair multi-party exchange of electronic goods over unreliable networks with non-repudiation, where goods are either signatures (i.e., non-repudiation tokens of public data), confidential data, or payments. The protocol does not involve a third party except for recovery over a reliable network.

1. Introduction

Many commercial transactions can be modelled as a sequence of *exchanges* of electronic goods. 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.

A straightforward solution for the fair exchange problem is to use a third party to ensure fairness by, for example, receiving the items to be exchanged and the expectations of the parties 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 all 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 powerful in terms of computational and communication resources – but, in this case, its operation will be an expensive undertaking.

In this paper, we describe a generic protocol for multi-party fair exchange which does not involve a third party in the normal, exception-less case: it is involved only in the presence of faults or in the case of dishonest parties who do not follow the protocol. 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.

After an overview of existing work on specific fair exchange protocols, we describe the service provided by fair exchange protocols in more detail. We then describe our optimistic protocol for fair exchange and analyse its security. Finally, we illustrate the protocol by describing some optimised instantiations such as "group contract signing."

2. 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 two-party fair exchange of money for goods. Several solutions for specific instances of the two-party 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.

The protocol described in [AsSW 96a] is the first proposal for two-party fair exchange of *generic* items. In this paper, we extend the generic fair exchange protocol to the multi-party case.

For *certified mail*, most practical solutions involve a third party even in the exception-less case [e.g., BaTy 94, Ford 94, Grim 93, Herd1 95, Herd2 95, ZhGo 96]. In cryptologic protocols for certified mail [Blum 82, Gold 82], 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.

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], both in terms of the types of items exchanged and the number of parties.

3. Fair Exchange Service

We now describe the service of generic multi-party fair-exchange. First, we recall the model of two-party fair exchange from [AsSW 96a] and then describe an extension to multi-party fair-exchange.

3.1 Service Description for Two-Party 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.

[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).

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 the item received was as expected. 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.

The service exchanges the items input by both participants iff the expect predicates both are true on input of the descriptions of the items to be exchange and both items match their descriptions.In Section 3.3, 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 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.

3.2 Service Description for Generic Multi-Party Fair Exchange

The service of a multi-party fair-exchange protocol is similar to a two-party exchange. Each party now inputs the parameters of the two-party exchange for every other party and a global *expect* predicate on all descriptions of all items to be sent. Globally, this leads to two *n*x*n* matrices describing the exchange: One containing the items and one containing their descriptions. An element in row *j* and column *k* contains the item or description of the item to be sent from party *j* to party *k*

Recipient→ **↓Originator**	**A**	**B**	**C**	**...**
A	-	$descr_{AB}$	$descr_{AC}$	
B	$descr_{BA}$	-	$descr_{BC}$	
C	$descr_{CA}$	$descr_{CB}$	-	
...				-

Table 1 Description Matrix for Multi-Party Exchange

(See Table 1).

The global expectation predicate on the whole exchange is defined on the nxn matrix of descriptions. It is input by each party: the $expect_k$() predicate from party k evaluates to *true* on input of the description matrix if party k is content with the exchanges described by the input description matrix.
The service exchanges all n^2 items atomically, iff all participants agreed on the description matrix and all expectations are met.

Note that our matrix model of the exchange corresponds to a topology of a fully connected graph: each party may exchange items with all others. Other (less connected) exchange topologies such as a ring can be supported by leaving parts of the matrix to be empty. In addition, defining one expectation predicate for all exchanged items instead of describing the expectation on each individual item sent, makes global requirements on the relation of different exchanges possible. An example for such a global requirement is that all parties exchange signatures on the same contract.

3.3 Exchangeable Items

In the generic fair exchange service descriptions, we used the terms *item*, *descr,* and *fits*() to represent the items to be exchanged. We now describe some different types of items to be exchanged; namely public data, confidential data, and payments:

- *confidential data:* data which is not known to the recipient beforehand and will be released to it during the protocol run, described by an optional text; examples include digital goods,
- *public data*[2]*:* 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 value from payer to payee.

Each type has specific descriptions and definitions of *fits*(). The expect predicate in all listed cases just compares if the description of the items expected equals the description of the items promised. A summary is given in Table 2. Note that evidence of the exchange is produced in any case, i.e., even exchanging something for nothing (i.e., just the evidence) makes sense, since the fact that something has been transferred can be proved afterwards.

	Conf. data	Public data	Payment
item	*data*	*data*	payment of *amount* to *payee*
descr	*specification*	*data*	*amount, payee*
***fits*()**	*may ask user*	*descr=item?*	compare *desc* with description output by payment protocol

Table 2 Different Types of Items

Additional item types, such as rights (credentials), can be exchanged, too, by defining appropriate descriptions and predicates.

We now describe these item types in more detail. 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 items should only be exchanged if these expectations are met. Since the

2 This is the special case of confidential data, with only a description.

data itself can in most cases not be checked automatically, the arbiter needs additional information to verify this agreement on the exchange. Therefore the initial agreement fixes a description to enable the recipient to check if it agrees on the description of the item to be received.

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, Word Processor for OS/2, Number of kB, provides at least the following features: …" which he has received as part of 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 protocol aborts. Later, the customer checks the program and if the program does not execute under OS/2, he may invoke an arbiter which may decide on the dispute.

In the case of public data, the primary purpose of the protocol is to produce evidence of its transfer. The data itself is either known to both parties or may be released even in the presence of faults. Examples are contract texts, the text of receipts, or the text of an order. Note that even if the exchanged public data is empty (e.g., in exchange for confidential data during certified mail) non-repudiation tokens are generated nevertheless; these can be used as evidence to later prove what was exchanged.

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 or undo the payment) or generatable (i.e., the bank may enforce a bank transfer given the amount and the accounts of payer and payee).

3.4 Instantiations of the Generic Services

We now list the exchanges which result from exchanging the described items using the generic exchange services. Table 3 lists instantiations of the one-to-one (i.e. two-party) case. Selected optimised two-party protocols for the resulting exchanges are described in [AsSW 96a]. Table 4 lists instantiations of the one-to-many case (either one sender and multiple receivers or vice versa), whereas Table 5 lists some instantiations of the many-to-many case where everyone sends the same type of item. Some optimised instantiation of our multi-party exchange protocol are described in detail in Section 3.4.

Single Party→ ↓Single Party	**Public data**	**Conf. data**	**Payment**
Public data	contract signing	certified mail	payment with receipt
Conf. data		exchange of digital goods	fair purchase
Payment			currency/type exchange

Table 3 **One-to-One Exchanges**

Single Party→ ↓Group of Parties	**Public data**	**Conf. data**	**Payment**
Public data	Group contract signing	Reliable certified broadcast	Fair group pay-out with receipt
Conf. data	Group commitment	Simultaneous fair exchange	Fair group sale

Payment	Simultaneous payment for receipt	Fair group purchase	-

Table 4 One-to-Many Exchanges

Public data	**Confidential data**	**Payment**
Group contract signing	Group fair exchange	-

Table 5 Many-to-Many Exchanges.

4. An Optimistic Multi-Party Fair Exchange Protocol

4.1 Protocol Description

The protocol for generic multi-party fair exchange is depicted in Figure 1. 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, optionally, a commitment scheme consisting of a protocol *commit*() to commit to an item and *verify*() to check if an opened commitment fits an item. The commitment scheme is necessary when the item to be transferred is confidential. We require from the commitment that nobody can change its contents without invalidating 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. *P* denotes the fixed list of parties. The protocol is depicted from the view of one party "*i*". The symbol $\{x_j\}_{j\in L}$ denotes a list of values one for each index *j* in the set *L*. "To *S*:" denotes sending a message to all parties in the set *S*.

The basic idea of the protocol is that each party signs the expected global description (i.e., the whole description matrix) of the exchange and commits to the items he will send to every other party. If all parties signed the same description matrix, the parties send the promised items. If someone does not receive what was expected, two-party recovery procedures are started. Our protocol does not assume either the availability of the network in general or the integrity of the messages received: they may be modified or not transmitted at all. Only reliability of the connection to the third party is required.

We now describe the protocol in more detail. Initially, each party *i* broadcasts its row $\{descr_{ij}\}_{j\in P}$ of the description matrix containing descriptions of the items it promises to send during the exchange. This message is not signed and may be omitted if all parties know the description matrix beforehand. After having received these rows of the description matrix, player *i* constructs its local view $descr^i$ of the description matrix. It then checks if the $expect_i()$ predicate returns "true" on input of the matrix. In the simplest case, $expect_i()$ should check if each description $descr_{ij}$ matches a fixed pre-defined description. In more elaborate scenarios, it may check to see if the matrices satisfy some additional requirements. For example, in the case of *n* parties signing a contract, the additional function would check to see if all proposed contracts are in fact the same (i.e. whether $\forall\ i,j\in P$, $descr_{ij}= Contract$). If the expectations are not fulfilled for *i,* i.e., $expect_i(descr^i)=false$, it aborts its protocol run..

In the second round, every party *i* broadcasts a signed message m_2 which includes:
1. The distinguishing name *T* of the third party to be invoked for resolving exceptions.
2. The list of parties *P*.

3. The active-time limit t, after which all parties are guaranteed that the state of the transaction is final. Requests for exceptions must be made before an earlier deadline. Each exception-handling third party T will announce a policy indicating how to determine the deadline for requests, given an agreed upon value for the active-time limit. We explain time-outs further, in Section 4.2.
4. Images $\{h(r_{ij})\}_{j\in P/\{i\}}$ of the one-way function $h()$. The pre-images r_{ij} are later used as non-repudiation tokens for receipt: for example, r_{ij} is released to signal that $item_{ij}$ has been received as expected. Similarily, the key key_{ij} of the commitments can be used to prove non-repudiation of origin of $item_{ij}$.
5. Commitments to the items to be sent $\{com_{ij}\}_{j\in P}$.
6. An image $h(y_i)$ where y_i may later be used to involve the third party T.
7. The local view $descr^i$ of the description matrix.

After receiving all messages $\{m_{2i}\}_{j\in P}$, each party can then compare the local views of all parties, i.e., the signed lists of parameters 1-3 and 6. If the views are not consistent, i.e., not all signed messages are identical with respect to these four parameters, or some messages are missing, the party suspends its protocol run and waits until the end of the active-time period — if there is no recovery request from T (initiated by some other party) during this period, the run is aborted at the end of active-time. If a party has received a consistent view, it continues. Note, that this may lead to the problem where some partners start the protocol after having received a consistent view and some will not. How this problem is resolved will be addressed later.

If i decides to proceed, then i sends the items $item_{ij}$ and the keys key_{ij} to open the commitments to each other party j. Similarly each party j waits for *message* m_{3ij} and broadcasts to *everyone* the pre-images r_{ji} in return in order to authenticate the non-repudiation of receipt tokens and signal the receipt of the item. Furthermore each party collects the non-repudiation of receipt tokens broadcast from other successful pairwise exchanges in which it did not participate. The protocol run for i ends successfully when it has received all items and tokens; i may still participate in any exceptions raised by others, by either replaying messages from the completed protocol run or extracting and presenting evidence generated during that run.

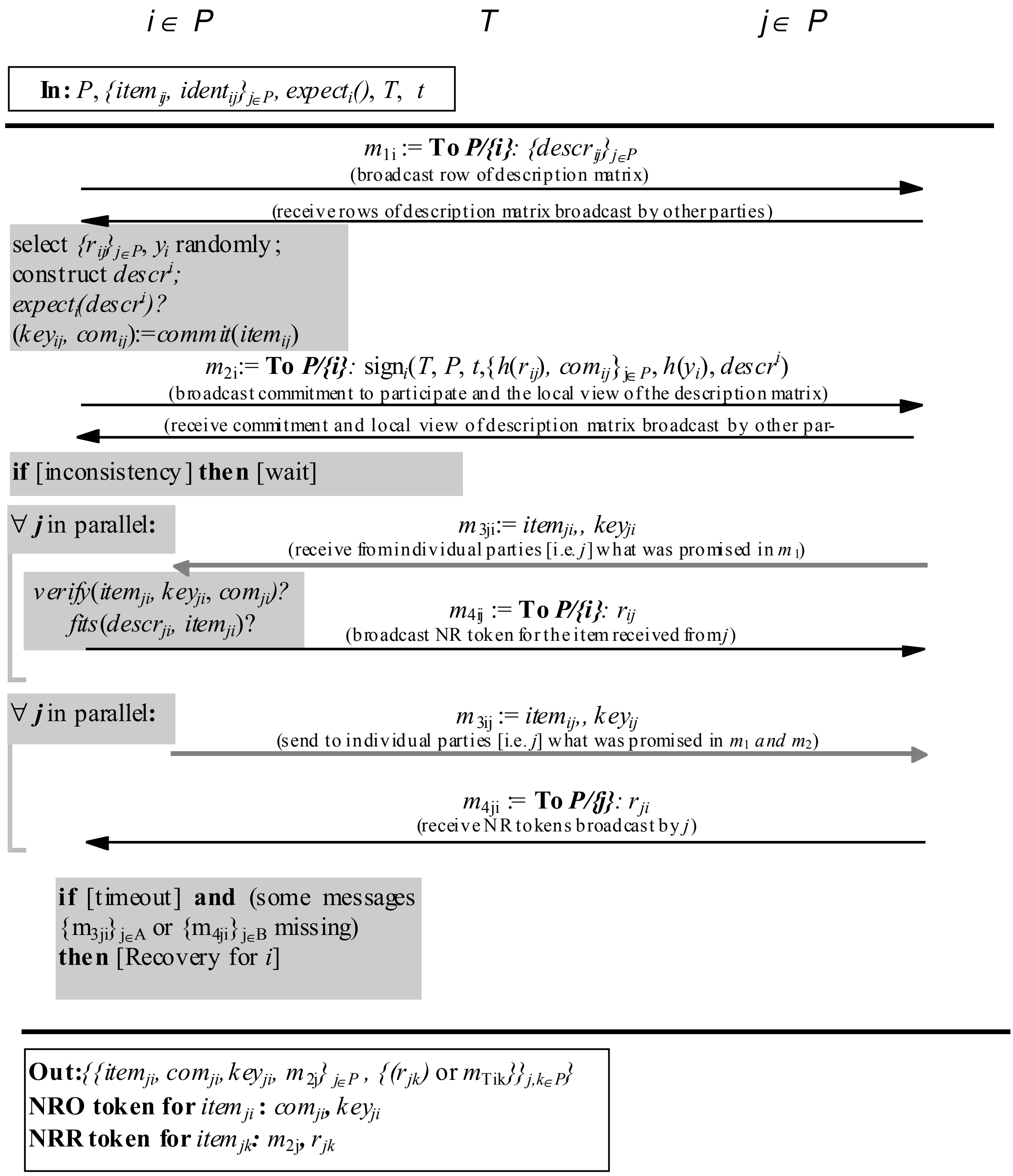

Figure 1 Generic Protocol for Multi-Party Exchange

If some non-repudiation of receipt tokens of any pairwise exchange or some expected items or keys to open the commitments are not received correctly within a certain period (each party may decide independently when to time out), T is invoked with necessary information to raise an exception. This is illustrated in Figure 2. Each party i can initiate recovery at most once. For each recovery request,, T starts a recovery phase by requesting the pairs of parties involved with each missing item to repeat the exchange while being observed by T.Notice that each honest party sends its item first and then waits for the items and tokens it expects to receive.

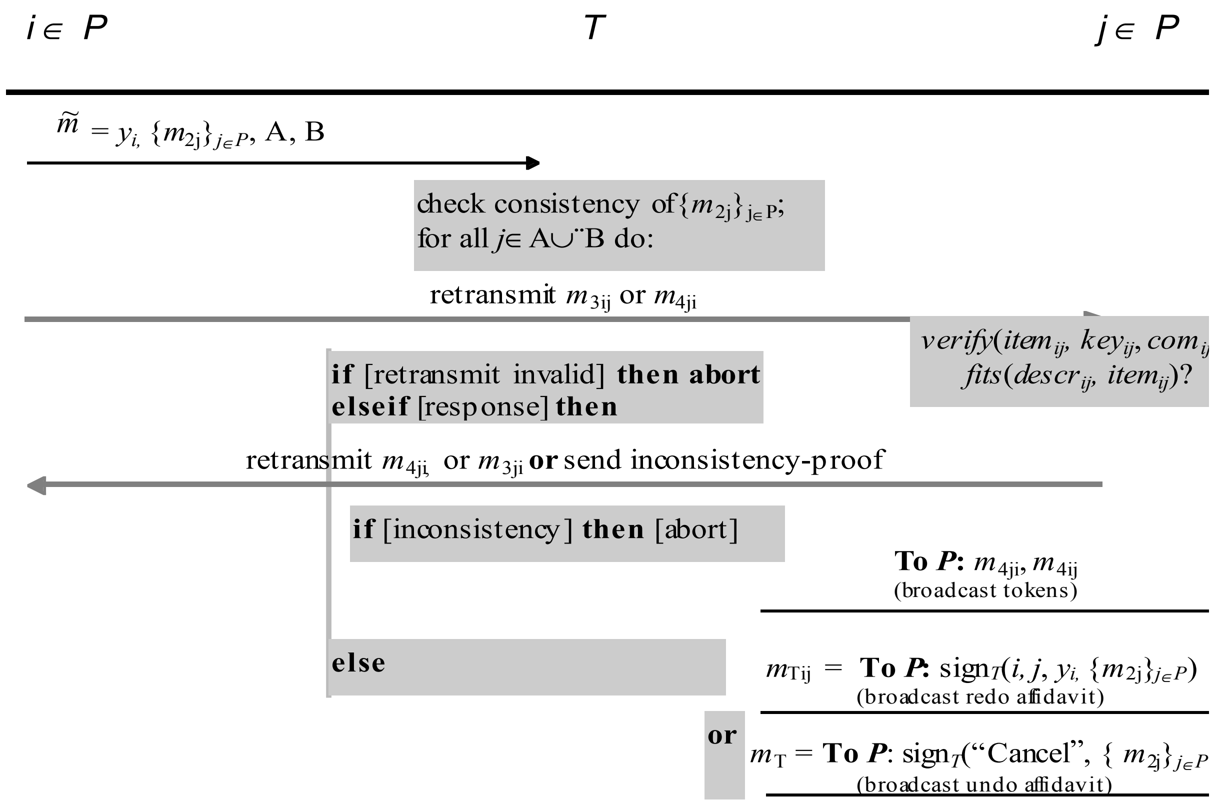

Figure 2 Recovery for *i*

We now discuss some additional exceptions which may occur and how they are resolved by the protocol. One exception is that all honest partners send their consistent local views but some messages are not delivered. Some partners will receive a complete set of signed messages and some will not. Only the partners who received a complete set will start the exchange. Naturally, the party who did not receive a consistent view containing all initial signed messages will not send its items. Therefore the party which received the complete consistent view and which has sent its items will complain to T that the party not having received a consistent view did not respond to the items sent During recovery, the party having an incomplete view will receive a consistent view from T and will now participate in the exchange.

Another exception is the presence of inconsistent local views: One dishonest party sends different messages m_2 to some partners in the second round, i.e., some partners receive a consistent view and some do not. Actually, this is no real problem: Since every honest party signs m_2 only once, the inconsistencies relate to dishonest parties signing multiple versions of m_2 (Note that due to this fact one single consistent view exists among honest parties). The recovery from this exception is straightforward: when T is invoked by an honest party presenting the only consistent set of signed messages, it tries to enforce these exchanges and thus forwarding the consistent view to all parties. If the party i invoking T is dishonest, i.e., has sent another message m_2' to the other parties different from the one sent to T, the provided view may be consistent but different from the view of the honest party forced to participate. In this case, the honest party may abort the recovery by proving the dishonesty of the party invoking T: It sends an inconsistency proof "inconsistency(i)" which contains the message m_2' which is signed by i but different to the message m_2 signed by i which has been sent by T. Note that the recovery is aborted only if the dishonesty of the party *initiating* the recovery is proven: a dishonest party cannot refute a recovery by proving the dishonesty of any of the other parties, including itself.

If an exchange is enforced but a (dishonest) party does not co-operate, T may either generate replacement items where possible/applicable or revoke all items and broadcast a cancel affidavit to all parties. Where neither generation nor revocation is possible, T may simply sign an affidavit containing all the messages exchanged during the exception handling. The other parties may then use this affidavit in an external dispute handling system, such as a court of law.

4.2 Time-Outs

The only critical time-out of the generic protocol in Section 4.1 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 final time — all parties are guaranteed that the status is not changed after t. 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, all parties 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. 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 T, the retransmission of the goods sent via T are critical messages. For the other parties, m_3 and m_4 are critical messages.

When i sends out m_{3ij}, it will start a local timer to determine when it should invoke T by sending $\tilde{m}$. The value τ_{ij} 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 i's end. The exact computation can be at best based on some rules of thumb. For example, if i sends out m_{3ij} at time instant t', and it estimates that the expected communication delay between it and T to be t_{iT}, then the estimate for τ_i will be $p_T(t,t_T)\text{-}\ t_{iT}$. If i prefers to use a safety factor s in its estimate, τ_i becomes $t' + (1\text{-}s)(\ p_T(t,t_T)\text{-}\ t_{iT} - t')$.

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.

4.3 Requirements on Multi-Party Fair Exchange

We state the requirements on fair multi-party exchange service in more detail. As mentioned above, the exchange is described by a so-called description matrix. The non-repudiation of exchange token

NRX is defined as a matrix of all NRR tokens for individual two-party exchanges. It indicates that every party was satisfied after the exchange.

I. **Consistency**
   A. The exchange must not start if the local views of the honest parties regarding what will be exchanged (i.e., the description matrices) are not identical.

II. **Meaning of Non-repudiation Tokens**
   A. If party *h* is honest and an NRX token for an exchange is output to a *h* then *h* can convince a party *j* that a complete exchange took place.

III. **Weak Fairness of Exchange and Atomicity**
   A. If party *h* is honest and *h* does not receive the expected items or the NRX or an affidavit for them from T, neither an NRX or part of the items sent by *h* is output to any other party.
   B. If participant *h* is honest and *h* receives the expected items and the NRX for the exchange, all honest parties received the expected items and the NRX or an affidavit from T, too.

IV. **Strong Fairness of Exchange and Atomicity**
   A. If party *h* is honest and *h* does not receive the expected items or the NRX, neither a NRX or part of the items sent by *h* is output to any other party.
   B. If party *h* is honest and *h* receives the expected items and a NRX for the exchange, all honest parties received the expected items and the NRX, too.

V. **No Unconditional Trust in the Third Party**
   A. If *T* and a party *h* are honest, *T* does not create affidavits in the name of *h*.
   B. If a party *h* is honest, no non-repudiation token or affidavit can be produced by *T* without any participation of *h*.

In the next section, we will argue that our protocol always achieves weak fairness. In addition, *T* can guarantee "strong" fairness if the items involved are either *all* revocable or *all* generatable. The notions of strong and weak fairness were introduced in [AsSW 96a]. We elaborate more on it in Sections 4.3 and 4.4.

4.4 Security Analysis

We now explain why the requirements on multi-party fair exchange stated in the previous section are fulfilled.

- **Consistency**
   A. The exchange must not take place, if the local views of the parties regarding what will be exchanged (i.e., the description matrices) are not identical.

If *no* honest party i has released any item, the protocol is correct. If a honest party i has released any item, there exists at least one consistent view in the system because i will never sign two versions of $desc^i$ and i's signature on $descr^i$ is required to complete the agreement phase.

- **Strong Fairness of Exchange and Atomicity**
   A. If party *h* is honest and *h* does not receive the expected items or the NRX, neither an NRX or part of the items sent by *h* is output to any other party.
   B. If party *h* is honest and *h* receives the expected items and an NRX for the exchange, all honest parties received the expected items and the NRX, too.

Assume that i did send out an item. It does so if and only if it has received a consistent view $desc^i$ of the description matrix. In particular, this is the same view signed by i in m_{1i}.

If i now receives all the items it expects, then the protocol is fair. Assume that it does not receive $item_{ij}$ (in message m_{3ji}) promised to it by party j or m_{4ji}, the NRR token for $item_{ij}$ from j. In this case, i will eventually initiate a recovery protocol with T. Since i has a consistent view of the description matrix *and* the commitment of every party on it, T will be convinced that i is honest. T will therefore allow i to repeat the last flow; if j is honest, it continues the protocol thus satisfying the correctness requirement from i's point of view. Note that if j is asked to continue, it receives a consistent view of the exchange signed by j itself. If j is dishonest and does not cooperate, T has two options to achieve strong fairness: If all items are generatable, it can send a replacement item to i, or if all items are revocable it can undo the complete exchange by revoking all items.

- **Weak Fairness of Exchange and Atomicity**
  - A. If party h is honest and h does not receive the expected items or the NRX or an affidavit for them from T, neither an NRX or part of the items sent by h is output to any other party.
  - B. If party h is honest and h receives the expected items and the NRX for the exchange, all honest parties received the expected items and the NRX or an affidavit from T, too.

An affidavit is a statement from T which is issued instead of generating an item. It states that some parties have been dishonest and did not participate in recovery. Therefore weak fairness follows immediately from the argument for generatable items.

- **Meaning of Non-repudiation Tokens**
  - A. If party j is honest and a non-repudiation token for an exchange is output to a party h then h can convince a party j that an complete exchange took place.

In case of an exchange without invocation of the third party, a valid NRX token contains signatures under m_{2i} from all parties fixing the items to be exchanged as well as non-repudiation of receipt tokens r_{ij} which state that the parties received what they expected. Since a honest party does not issue such tokens without receiving the item, all honest parties received what they expected, i.e., an complete exchange took place. In case the third party has been invoked, the same argument holds for all completed two-party exchanges. The missing two-party exchanges have been settled, if the invoking party was honest.
Note that a NRR token may not be valid on its own, i.e., if one allows revocation of tokens, one must contact all parties or the third party to check the validity of a token.

- **No Unconditional Trust in the Third Party**
  - A. If T and a party h are honest, T does not create affidavits or items in the name of h.
  - B. If a party h is honest, no non-repudiation tokens, item or affidavit can be produced by T without any participation.

If a party is honest, it sends the requested item after receiving the expected item from T. Therefore, according to the protocol, there is no need for T to generate an affidavit or replacement items.
For all actions during recovery, T is required to verify h's signature on the initial messages. Therefore without any participation by h, T is not able to start recovery of an exchange involving h and thus will not generate any replacement or affidavit on behalf of h.

4.5 Requirements for Strong Fairness

In the preceding sections, we stated that strong fairness can be provided for all parties if the items involved are either all revocable or all generatable. It is possible to provide strong fairness even with weaker assumptions. First, consider the case where some items are revocable and all the rest are generatable. Let *descr* be the description matrix for these items. Then, we can provide strong fairness for all parties by composing a "super" exchange protocol run consisting of three phases:
1. An overall agreement phase in which all parties exchange m_2 indicating their agreement in *descr*,

2. a run of the generic exchange protocol in which all the revocable items are exchanged, and
3. a run of the generic exchange protocol in which all the generatable items are exchanged.

The description matrix of the exchange in phase 2 contains only the revocable items of *descr* and that for the exchange in phase 3 contains only the generatable items of *descr*. If there is an exception during phase 2, then T may either revoke all items sent so far (in which case, the entire protocol is terminated) or ensure that all recipients of revocable items have received them. Notice that there is no need for global synchronisation between phases. A party i can proceed from phase 2 to phase 3 as soon as it has received NRR tokens for *all* the items sent in phase 2 (i.e. all revocable items in *descr*): either all other parties would have also received these NRR tokens or they would eventually invoke T which will issue replacement tokens. If phase 3 fails in the agreement stage, T can be invoked with the overall agreement obtained in phase 1. Alternatively, the agreement stage may be omitted in phase 3.

What about items that are neither revocable nor generatable? Strong fairness may be provided if there is at most one party in an exchange which wants to send out such items. Between phases 2 and 3, the party with these items (say k) sends them out. It is easy to see that this preserves the strong fairness property. Consider the security of k: it receives only revocable and generatable items; by the time it releases the non-revocable/non-generatable items, k has received all the revocable items it expects; by definition, it can receive all the generatable items it expects either directly from the senders or from T. If k cheats by not sending all of these items, no correctly behaving party will enter phase 3. Instead, T will be invoked to revoke all the items sent in phase 1. Note that if k only sends out *some* of these items, it is not guaranteed any fairness — this is not surprising since k's behaviour was dishonest. Also, it is straightforward to see that if two or more parties were to send out such items, it is not possible to guarantee strong fairness since no recovery is possible if one of them is dishonest and does not send the item.

5. Optimised Protocols for Multi-Party Fair Exchange

5.1 Group Contract Signing

This instantiation of the generic protocol exchanges non-repudiation tokens on a contract text previously known to all parties. Thus, there is no need for sending the contract text in message m_1. We use r_j to indicate that user j has received a consistent view. The "description" of the item is the contract text itself. There is no need to explicitly send the item in m_3 since nothing is confidential. The *expect()* primitive checks to make sure that the descriptions in all m_{2j} are identical to the contract text. During a dispute, a respondent can prove inconsistency of a party k by producing m'_{2k} which differs from m_{2k} only in the last field (i.e. text of the contract). In case of inconsistency, T may either complete the contract (either by obtaining and broadcasting missing contracts and/or non-repudiation tokens or by issuing replacement non-repudiation tokens) or invalidate it (by broadcasting a cancellation token). The choice may depend on T's policy or may be included as a parameter in m_2.

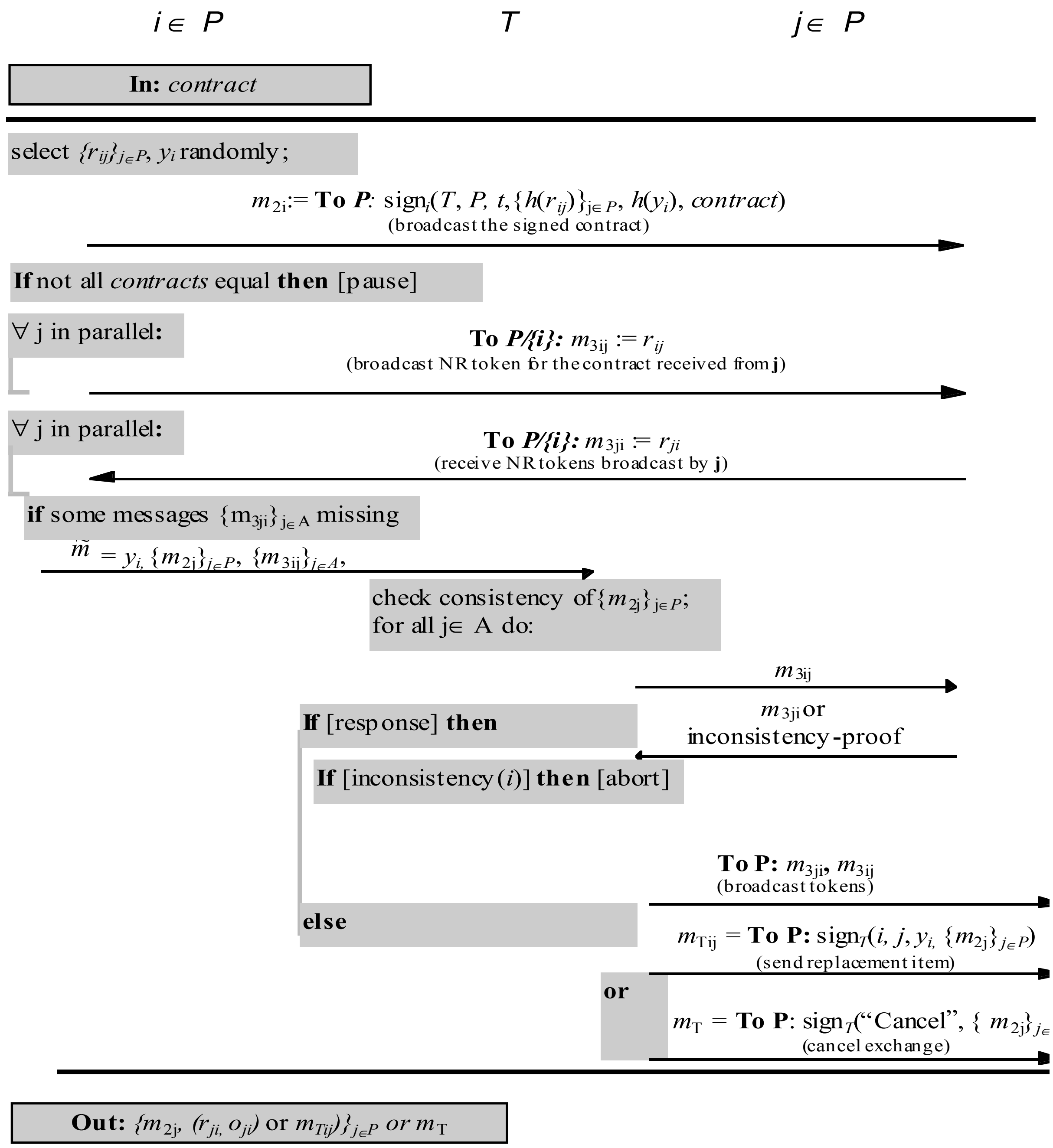

Figure 3 Protocol for Group Contract Signing

5.2 Reliable Certified Broadcast

We now describe a protocol for reliable certified broadcast: one sender i sends certified mail to a set of receivers $j \in P/\{i\}$. If we would strictly follow the generic protocol, in m_1 the sender (i) will broadcast the commitment *com* on the text of the message. In m_2, i will broadcast the signature on *com* and all other parties will broadcast their commitment on the NRR tokens for an item committed to in *com*. We can avoid m_1 by requiring that i send out m_{2i} first and then wait for others to send m_{2j}.

In m_3, only i sends out an item; none of the other recipients have anything to send. In m_4, all the recipients send NRR tokens. During a dispute, there is no inconsistency to prove — T always broadcasts a NRR token on behalf of any j that has not responded even though it has received a consistent set of messages.

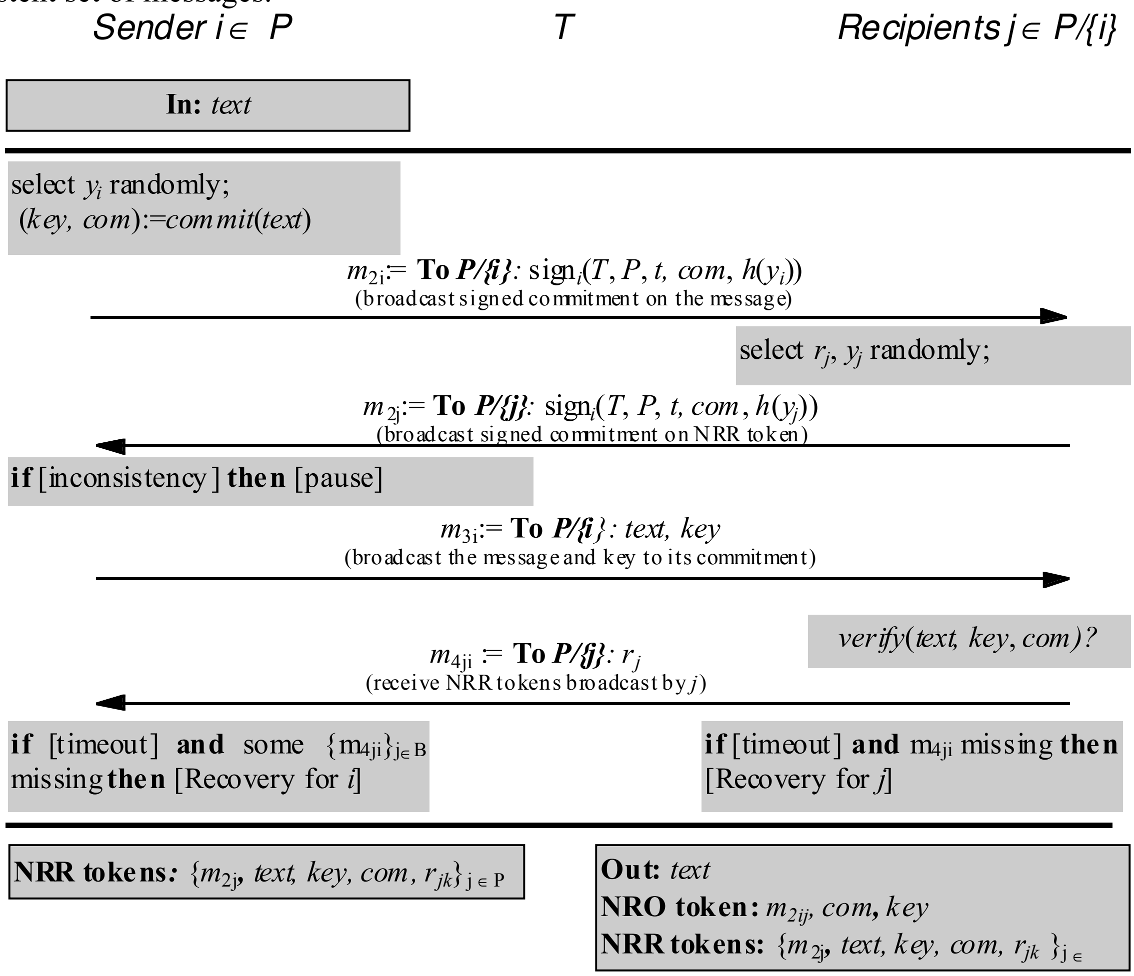

Figure 4 Protocol for Reliable Certified Broadcast

6. References

AsSW 96a N. Asokan, M. Schunter, M. Waidner: Optimistic Protocols for Fair Exchanges, to be presented at the Fourth ACM Conference on Computer and Communications Security, Zürich, 1997.

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.

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 Sigfried 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.

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.

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.