A Multi-party Optimistic Non-repudiation Protocol

Olivier Markowitch and Steve Kremer
{omarkow, skremer}@ulb.ac.be

Université Libre de Bruxelles
Computer Science Department
Bd du Triomphe C.P. 212
1050 Brussels Belgium

Abstract. In this paper we consider the optimistic approach of the non-repudiation protocols. We study a non-repudiation protocol with off-line trusted third party and we keep on with the definition of the multi-party non-repudiation, compare it to multi-party fair exchange and show some fundamental differences between these two problems. Finally, we generalize our protocol and propose a multi-party non-repudiation protocol with off-line trusted third party.

1 Introduction

The impressive growth of open networks during the last decade has given more importance to several security related problems. The non-repudiation problem is one of them. Non-repudiation services must ensure that when Alice sends some information to Bob over a network, neither Alice nor Bob can deny having participated in a part or the whole of this communication. Therefore a non-repudiation protocol has to generate non-repudiation of origin evidences intended to Bob, and non-repudiation of receipt evidences destined to Alice. In case of a dispute (e.g. Alice denying having sent a given message or Bob denying having received it) an adjudicator can evaluate these evidences and take a decision in favor of one of the parties without any ambiguity.

In comparison to other security issues, such as privacy or authenticity of communications, non-repudiation has not been studied intensively. However many applications such as electronic commerce, fair exchange, certified electronic mail, etc. are related to non-repudiation. Non-repudiation of origin can easily be provided by signing the sent information. A digital signature provides an irrefutable non-repudiation of origin evidence: a digital signature creates a link between a message and a public verification key. A certification authority assures the correspondance between this public signature verification key and an identity. Non-repudiation of receipt is more difficult to achieve: therefore Alice and Bob have to follow a protocol that assures both services. Such a protocol is said to be fair if either Alice receives a non-repudiation of receipt evidence and Bob receives a non-repudiation of origin evidence or none of them obtains any valid evidence.

First solutions to those problems involve a trusted third party (TTP) that acts as an intermediary between the participating entities. The major disadvantage of this approach is the communication bottleneck created at the TTP. Therefore more efficient solutions have been proposed. Two different approaches have been considered: one consists in designing protocols without a TTP, the other tries to minimize its involvement.

The approach without TTP involvement is often based on a gradual release of the knowledge. However it generally requires that all involved parties have the same computational power. Another disadvantage is the important number of transmitted messages. In [6] a protocol for digital contract signing without TTP is proposed: the probability that the contract has been signed, is increased each round until reaching one. The assumption of same computational power is not needed. Another recently presented probabilistic non-repudiation protocol [13], also succeeded in relaxing the condition on the computational power. Here the idea is that the recipient does not know a priori which transmission will contain the message. The probability of guessing the transmission including the message is arbitrarily small. However to decrease the probability the number of messages has to be increased.

The other approach, trying to minimize the TTP involvement has got more attention in literature during the last years. Asokan et al. presented, in the context of fair exchange, the optimistic approach[4, 1]: usually all participants are honest and only in the case of a misbehaving party, the TTP has to be involved. The protocols inspired by this approach are said to be protocols using an off-line TTP.

The most complete non-repudiation protocols with off-line TTP have been presented in [16] and independently in [12]. The optimistic approach assumes that in general Alice and Bob are honest, i.e. they correctly follow the protocol, and the TTP only intervenes, by the mean of a recovery protocol, when a problem arises.

Most of the proposed non-repudiation protocols are two-party protocols. In fair exchange first works have been done to generalize them to the case of n participants [3, 2, 9, 5]. Considering non-repudiation protocols, the only to us known work generalizing these protocols has been presented in [11] and concerns a non-repudiation protocol with on-line TTP (which intervenes in each session of the protocol, even if the parties behave correctly).

The protocol presented here considers an off-line TTP. To the best of our knowledge, the only comparable work is the multi-party certified mail protocol proposed by Asokan et al. in [2]. However the here proposed protocol is more general: as it will be outlined later in more details, the certified mail protocol only continues if the whole set of receivers is willing to do so. Our protocol leaves the choice to the sender to finish with only the subset of the responding receivers, or to stop in the case of a non-responding receiver, as the certified mail protocol does.

We first remind the properties required by non-repudiation protocols. Then, we go on presenting a two-party non-repudiation protocol with off-line TTP.

Afterwards, we define the multi-party non-repudiation problem, showing some differences with multi-party fair exchange. The requirements of a multi-party non-repudiation protocol are defined as well. Finally, we present a generalization of the previously presented two-party non-repudiation protocol to the case of n parties, using a group encryption scheme.

2 Properties

2.1 The Communication Channels

In the framework of such exchange protocols, we can distinguish three classes of communication channels: unreliable channels, resilient channels and operational channels. No assumptions have to be made about unreliable channels: data may be lost. A resilient channel delivers data after a finite, but unknown amount of time. Data may be delayed, but will eventually arrive. When using an operational channel data arrive after a known, constant amount of time. Operational channels are however rather unrealistic in heterogenous networks.

2.2 Requirements on Non-repudiation Protocols

A first property non-repudiation protocols must provide is fairness. One can distinguish between the two notions of *strong fairness* and *weak fairness*.
A non-repudiation protocol is said to provide *strong fairness* if at the end of the protocol Alice has got a complete non-repudiation of receipt evidence if and only if Bob has got the message with a complete corresponding non-repudiation of origin evidence.
A non-repudiation protocol provides *weak fairness* if either the protocol provides strong fairness, or the protocol provides evidences that can prove to an adjudicator until which state the protocol has been executed.

A second property we require is timeliness: a protocol must be finished after a finite amount of time for each participating entity that is behaving correctly with respect to the protocol.

Each protocol must ensure both properties to be acceptable.

3 A Two-Party Non-repudiation Protocol with Off-line TTP

3.1 Introduction

We will now present a two-party non-repudiation protocol with an off-line TTP [12]. In this protocol, Alice wants to exchange a message m and its corresponding non-repudiation of origin evidence against a non-repudiation of receipt evidence, issued by Bob.

This protocol results from modifications made on the Zhou-Gollman optimistic protocol [18] in which an operational channel is needed between the TTP and Alice in order to assure fairness. The here presented protocol only needs a resilient channel between the TTP and respectively Alice and Bob. The channel between Alice and Bob may even be unreliable. The protocol is similar to the independently developed autonomous two-party non-repudiation protocol proposed earlier by Zhou et al. in [16].

The protocol is composed of three sub-protocols: the main protocol, the recovery protocol and the abort protocol. The main protocol consists of messages exchanged directly between Alice and Bob. In case of problems during this main protocol, two (mutually exclusive) possibilities are offered to the entities. Either Alice contacts the TTP to abort the protocol in order to cancel the exchange, or Alice or Bob contacts the TTP to launch the recovery protocol in order to complete the exchange.

3.2 Notations

We use the following notation to describe the protocol:

- $X \rightarrow Y$: transmission from entity X to entity Y
- $X \leftrightarrow Y$: ftp get operation performed by X at Y
- $h()$: a collision resistant one-way hash function
- $E_k()$: a symmetric-key encryption function under key k
- $D_k()$: a symmetric-key decryption function under key k
- $S_X()$: the signature function of entity X
- m: the message sent from A to B
- k: the message key A uses to cipher m
- $c = E_k(m)$: the cipher of m under the key k
- $l = h(m, k)$: a label that, in conjunction with (A, B) uniquely identifies a protocol run
- f: a flag indicating the purpose of a message
- $\mathsf{EOO} = S_A(f_{\mathsf{EOO}}, B, l, h(c))$
- $\mathsf{EOR} = S_B(f_{\mathsf{EOR}}, A, l, h(c))$
- $\mathsf{Sub} = S_A(f_{\mathsf{Sub}}, B, l, E_{TTP}(k))$
- $\mathsf{EOO}_k = S_A(f_{\mathsf{EOO}_k}, B, l, k)$
- $\mathsf{EOR}_k = S_B(f_{\mathsf{EOR}_k}, A, l, k)$
- $\mathsf{Rec}_X = S_X(f_{\mathsf{Rec}_X}, Y, l)$
- $\mathsf{Con}_k = S_{TTP}(f_{\mathsf{Con}_k}, A, B, l, k)$
- $\mathsf{Abort} = S_A(f_{\mathsf{Abort}}, B, l)$
- $\mathsf{Con}_a = S_{TTP}(f_{\mathsf{Con}_a}, A, B, l)$

3.3 Main Protocol

The basic idea of the main protocol is to first exchange the cipher of the message m against a receipt for this cipher. Secondly, we exchange the decryption key against a receipt for this key. Each transmission is associated to some maximum time-out. Once this time-out value is exceeded the recipient supposes the transmission will not arrive and initiates either a recovery or an abort protocol.

1. $A \rightarrow B:$ $f_{\mathsf{EOO}}, f_{\mathsf{Sub}}, B, l, c, E_{TTP}(k), \mathsf{EOO}, \mathsf{Sub}$
2. $B \rightarrow A:$ $f_{\mathsf{EOR}}, A, l, \mathsf{EOR}$ (time-out: abort)
3. $A \rightarrow B:$ $f_{\mathsf{EOO}_k}, B, l, k, \mathsf{EOO}_k$ (time-out: recovery$[X := B, Y := A]$)
4. $B \rightarrow A:$ $f_{\mathsf{EOR}_k}, A, l, \mathsf{EOR}_k$ (time-out: recovery$[X := A, Y := B]$)

Alice starts the protocol by sending the cipher of the message, as well as the decryption key, ciphered under the public key of the TTP, to Bob. The message does also contain Alice's signature on the encrypted key and the hash of the cipher. These signatures serve as evidences of origin for the ciphers. In this first message we use two purpose flags as it conveis two proofs (EOO and Sub).

If Bob receives the first message he replies with a receipt to confirm the arrival of the first message. This receipt contains Bob's signature on the hash of the cipher c and serves to Alice as an evidence of receipt for the cipher.

In the case that Alice does not receive message 2 from Bob before a given time-out, she initiates the abort protocol. Note that Alice cannot perform a recovery at this moment, as the recovery protocol requires EOR, the evidence of receipt for the cipher.

If message 2 arrives to Alice before the time-out, she sends to Bob the decryption key k, as well as her signature on this key. This signature is used as an evidence of origin for the key. The evidence of origin for the cipher c, together with the evidence of origin for the key k, form together the non-repudiation of origin evidence of the message m.

Message 3 has to arrive to Bob before a given time-out. Otherwise Bob initiates the recovery protocol with the TTP.

If message 3 arrives in time, Bob sends a receipt for the key to Alice: his signature on the key k. The signature serves as the evidence of receipt for the key. Together with the evidence of receipt for the cipher c, they form the non-repudiation of receipt evidence of the message m.

Alice may also initiate the recovery protocol with the TTP if this last message does not arrive in time.

3.4 Recovery Protocol

To launch the recovery protocol Alice or Bob has to send to the TTP the hash of c, the key k ciphered for the TTP, the evidence of origin for the cipher c, EOO, the evidence of origin for the encrypted key, Sub, the evidence of receipt for the cipher c, EOR, as well as the evidence of origin for the recovery request, Rec_X (where X may take the values A or B). Note that the recovery protocol can only be executed once per protocol run and is mutually exclusive with the abort protocol.

By the mean of these evidences the TTP can be sure that Alice sent the cipher c to Bob and that Bob really received it.

1. $X \rightarrow TTP$: $f_{\mathsf{Rec}_X}, f_{\mathsf{Sub}}, Y, l, h(c), E_{TTP}(k), \mathsf{Rec}_X, \mathsf{Sub}, \mathsf{EOR}, \mathsf{EOO}$

if aborted or recovered then stop
else recovered=true

2. $TTP \rightarrow A$: $f_{\mathsf{Con}_k}, A, B, l, k, \mathsf{Con}_k, \mathsf{EOR}$
3. $TTP \rightarrow B$: $f_{\mathsf{Con}_k}, A, B, l, k, \mathsf{Con}_k$

When the first message arrives, the TTP checks wether an abort protocol or a recovery protocol has already been started for this protocol run: a protocol run is uniquely identified by the label $l = h(m, k)$ and the identities (A, B). If either an abort or a recovery protocol has already been initiated the TTP halts. The resilient channels assure that some message ending the protocol will eventually arrive at X. If the TTP accepts to perform a recovery protocol, the TTP sends to Alice the confirmation of submission of the key, as well as the evidence of receipt for the cipher EOR. It is important to include EOR, as Bob can initiate the recovery protocol after having received the cipher, without having sent a receipt for it. The TTP sends to Bob the key k, as well as the confirmation of the submission of the key, serving to Bob as an evidence of origin for k.

If the recovery protocol is executed, the key confirmation evidence Con_k will make part of the non-repudiation evidences for the message m. It is used to replace both the evidence of origin for the key as well as the evidence of receipt for the key.

3.5 Abort Protocol

Alice has the possibility to run an abort protocol. If she decides to do so she sends a signed abort request, including label l, to the TTP.

If the TTP accepts the request (neither a recovery nor an abort has yet been initiated), the TTP sends to both Alice and Bob a signed abort confirmation.

if recovered or aborted then stop
aborted=true

1. $A \rightarrow TTP$: $f_{\mathsf{Abort}}, l, B, \mathsf{Abort}$
2. $TTP \rightarrow A$: $f_{\mathsf{Con}_a}, A, B, l, \mathsf{Con}_a$
3. $TTP \rightarrow B$: $f_{\mathsf{Con}_a}, A, B, l, \mathsf{Con}_a$

Note that Alice could specify a wrong B in her abort request. However this would refer to a different protocol run—a protocol run is identified by l and (A, B)—and would enable Bob to launch a recovery protocol.

3.6 Dispute Resolution

The non-repudiation of origin and receipt evidences for message m are the following :

– $\mathsf{NRO} = (\mathsf{EOO}, \mathsf{EOO}_k)$ or $\mathsf{NRO} = (\mathsf{EOO}, \mathsf{Con}_k)$
– $\mathsf{NRR} = (\mathsf{EOR}, \mathsf{EOR}_k)$ or $\mathsf{NRR} = (\mathsf{EOR}, \mathsf{Con}_k)$

Repudiation of Origin. When Alice denies the origin of the message, Bob has to present to the judge EOO, EOO_k or Con_k, l, c, m and k. The judge verifies that

- $\mathsf{EOO} = S_A(f_{\mathsf{EOO}}, B, l, c)$,
- $\mathsf{EOO}_k = S_A(f_{\mathsf{EOO}_k}, B, l, k)$ or $\mathsf{Con}_k = S_{TTP}(f_{\mathsf{Con}_k}, A, B, l, k)$,
- $l = h(m, k)$,
- $c = E_k(m)$.

If Bob can provide all the required items and all the checks hold, the adjudicator claims that Alice is at the origin of the message.

Repudiation of Receipt. When Bob denies receipt of m, Alice can prove his receipt of the message by presenting EOR, EOR_k or Con_k, l, c, m and k to a judge. The judge verifies that

- $\mathsf{EOR} = S_B(f_{\mathsf{EOR}}, A, l, h(c))$,
- $\mathsf{EOR}_k = S_B(f_{\mathsf{EOR}_k}, A, l, k)$ or $\mathsf{Con}_k = S_{TTP}(f_{\mathsf{Con}_k}, A, B, l, k)$,
- $l = h(m, k)$,
- $c = E_k(m)$.

If Alice can present all of the items and all the checks hold, the adjudicator concludes that Bob received the message.

3.7 Fairness and Timeliness

If Bob stops the protocol after having received the first message, Alice may perform the abort protocol, in order to avoid Bob to initiate a recovery later. As neither Bob nor Alice received complete evidences the protocol remains fair. If Bob had already initiated the recovery protocol, the TTP sends all the missing evidences to Alice and Bob. Note that the TTP also sends the EOR to Alice, as she has not received it yet. Thus the protocol stays fair.

If Alice does perform step 3, Bob receives a complete non-repudiation of origin evidence. There are two ways to finish the protocol: either Bob sends message 4 of the main protocol and Alice receives a complete non-repudiation of receipt evidence or Alice performs the recovery protocol. As the channels between the TTP and both Alice and Bob are resilient, all data sent by the TTP to Alice and Bob eventually arrive. In both cases all entities receive valid evidences and the protocol finishes providing fairness.

If Alice does not send message 3 during the main protocol, Alice and Bob may initiate the recovery protocol. Fairness is still guaranteed, as during the recovery protocol, Alice and Bob receive all expected evidences.

In the case where Alice sends a wrong key k' either to Bob or to the TTP ($E_{TTP}(k')$) the evidences will not be valid as $l \neq h(m, k')$. If Alice also transmits the label $h(m, k')$, the generated evidence will correspond to the message $m' = D_{k'}(c)$. As m' is the message Bob actually received, fairness is still provided.

Note that the protocol achieves strong fairness.

When looking at the timeliness, three situations may arrive: the main protocol ends up successfully (without any time-out); Alice aborts the protocol and the abort confirmation signed by the TTP arrives at Alice and Bob after a finite amount of time, as the channels between the TTP and both Alice and Bob are resilient; a recovery protocol is performed and Alice and Bob receive the evidences after a finite amount of time because of the resilience of the channels.

4 Multi-party Non-repudiation

In literature, different kinds of multi-party fair exchange have been considered. In [9] a classification has been proposed. One can differ between single-unit and multi-unit exchanges. Moreover different topologies are possible: [9] and [5] concentrated on a ring topology. Each entity e_i $(0 \leq i \leq n-1)$ desires an item (or a set of items) from entity $e_{i \boxminus 1}$ and offers an item (or a set of items) to entity $e_{i \boxplus 1}$, where $\boxplus$ and $\boxminus$ respectively denote addition and subtraction modulo n. Another topology is the more general matrix topology, where each entity may desire items from a set of entities and offer items to a set of entities. Such protocols have been proposed by Asokan et al. in [3] and [2].

A fundamental difference between non-repudiation and fair exchange is the following. In non-repudiation, the originator sends some data with a non-repudiation of origin evidence to a recipient, who has to respond with a non-repudiation of receipt evidence. The sent data is generally not known to the recipient a priori. In a fair exchange each entity offers an a priori known item and receives another item, also known a priori. In a multi-party fair exchange protocol one can imagine sending an item to one entity and receiving an item from a different one. In non-repudiation it does not make sense that one entity receives some data and a distinct entity sends the corresponding receipt. Thus a ring topology is not sound. The most natural and here considered generalization seems to be a one-to-many protocol, where one entity sends a message to $n-1$ receiving entities who respond to the sender. However other possibilities for generalization exist (many-to-one, many-to-many) although they seem to be less natural.

The expectations we have towards such a protocol are rather similar to the properties required in two-party non-repudiation. A multi-party non-repudiation protocol is said to provide *strong fairness* if at the end of the protocol the sender has got a complete non-repudiation of receipt evidence for a given recipient if and only if this recipient has got the message with a complete corresponding non-repudiation of origin evidence.
A multi-party non-repudiation protocol is said to provide *weak fairness* if either the protocol provides strong fairness, or the protocol provides evidences that can prove to an adjudicator until which state the protocol has been executed with each of the receivers.

Here we can clearly see the difference with the certified mail protocol proposed by Asokan et al. Their protocol requires that at the end of the protocol *all* receivers have got the message with corresponding origin evidence and that the sender has got a receipt for *every* receiver, or none of them gained any valuable information. A last required property is the timeliness property, defined as for two-party protocols.

5 A Multi-party Optimistic Non-repudiation Protocol

We propose a generalization of the presented two-party non-repudiation protocol, using an off-line TTP. We suppose that the channels between the TTP and both Alice and all possible receivers are resilient. The channels between Alice and any receiver may be unreliable.

5.1 Notations

The following notation will be used:

- $\mathcal{B}$: the set of receiving entities
- $A \Rightarrow \mathcal{B}$: multicast from Alice to the set of entities $\mathcal{B}$
- $E_{\mathcal{E}}()$: a group encryption scheme E, that can be deciphered by each party $P \in \mathcal{E}$
- $\mathcal{B}'$: the set of receiving entities having sent an evidence of receipt for the cipher to Alice
- $l = h(m,k)$: a label that in conjunction with the identity A uniquely identifies a protocol run
- $\mathsf{EOO} = S_A(f_{\mathsf{EOO}}, \mathcal{B}, l, t, h(c))$
- $\mathsf{EOR}_i = S_{B_i}(f_{\mathsf{EOR}}, A, l, h(c))$
- $\mathsf{Sub} = S_A(f_{\mathsf{Sub}}, \mathcal{B}, l, E_{TTP}(k))$
- $\mathsf{EOO}_k = S_A(f_{\mathsf{EOO}_k}, \mathcal{B}', l, h(k))$
- $\mathsf{EOR}_{i,k} = S_{B_i}(f_{\mathsf{EOR}_k}, A, l, h(k))$
- $\mathsf{Rec}_X = S_X(f_{\mathsf{Rec}_X}, A, \mathcal{B}, l)$
- $\mathsf{Con}_k = S_{TTP}(f_{\mathsf{Con}_k}, A, \mathcal{B}', l, h(k))$
- $\mathsf{Early} = S_{TTP}(f_{\mathsf{Early}}, l)$
- $\mathsf{Set} = S_A(f_{\mathsf{Set}}, \mathcal{B}', l)$

5.2 Main Protocol

In the main protocol Alice sends a cipher of the message to all potential receivers $\mathcal{B}$. Several of these receivers ($\mathcal{B}'$) are sending a receipt for this cipher. Alice continues the protocol with the receivers in $\mathcal{B}'$ by sending them the key corresponding to the cipher of message 1.

1. $A \Rightarrow \mathcal{B}$:	$f_{\mathsf{EOO}}, f_{\mathsf{Sub}}, \mathcal{B}, l, t, c, E_{TTP}(k), \mathsf{EOO}, \mathsf{Sub}$
2. $B_i \rightarrow A$:	$f_{\mathsf{EOR}}, A, l, \mathsf{EOR}_i$
	where $B_i \in \mathcal{B}$ and $i \in \{1, \ldots, \lvert\mathcal{B}\rvert\}$
3. $A \Rightarrow \mathcal{B}'$:	$f_{\mathsf{EOO}_k}, \mathcal{B}', l, E_{\mathcal{B}'}(k), \mathsf{EOO}_k$
4. $B_i' \rightarrow A$:	$f_{\mathsf{EOR}_k}, A, l, \mathsf{EOR}_{i,k}$
	where $B_i' \in \mathcal{B}'$ and $i \in \{1, \ldots, \lvert\mathcal{B}'\rvert\}$

The first message destined to all receivers in $\mathcal{B}$ includes the label, the cipher, as well as the key k ciphered using the public key of the TTP (this information is used by the TTP in the case of a recovery). Alice also sends a time-out t: a recovery may only be performed after t. If one of the receivers does not accept the time-out he may stop the protocol.

After having sent the cipher in message 1, Alice decides of the moment to continue the protocol. All receipts arriving after this moment are not considered any more, without any risk for the receivers of losing fairness. To realize a service similar to the certified mail presented in [2], Alice may stop the protocol if not all of the receivers in $\mathcal{B}$ have answered.

Afterwards, when Alice sends the deciphering key it is crucial that only the recipients in $\mathcal{B}'$ receive it. Therefore we need to use a cipher (this is not needed in a two-party protocol). In order to cipher only once and to use multicasting, Alice uses a group encryption scheme. The idea is that the key can be ciphered in a way such that only recipients $B'_i \in \mathcal{B}'$ can decipher it. Examples of such ciphering methods can be found in [8] and [10].

5.3 Recovery Protocol

At each moment during the main protocol Alice and Bob have the possibility to launch the recovery protocol with the TTP. The recipients in $\mathcal{B}'$ launch a recovery if Alice does not multicast the ciphered key; Alice initiates the recovery protocol, if not all recipients in $\mathcal{B}'$ send a receipt for the ciphered key.

```
    1. $X \rightarrow TTP:$   $f_{\mathsf{Rec}_X}, f_{\mathsf{Sub}}, A, \mathcal{B}, l, t, h(c), E_{TTP}(k), \mathsf{Rec}_X, \mathsf{Sub}, \mathsf{EOO}$
if recovered then
    $TTP \rightarrow X:$   $f_{\mathsf{Con}_k}, A, \mathcal{B}', l, E_{\mathcal{B}'}(k, S_{TTP}(k)), \mathsf{Con}_k$
else if before t then
    $TTP \rightarrow X:$   $f_{\mathsf{Early}}, \mathsf{Early}$
else
    recovered=true
    2. $TTP \leftrightarrow A:$   $f_{\mathsf{Set}}, \mathcal{B}', l, \mathsf{Set}$
    3. $TTP \rightarrow A:$   $f_{\mathsf{Con}_k}, A, \mathcal{B}', l, \mathsf{Con}_k$
    4. $TTP \Rightarrow \mathcal{B}' \cup \{X\} \backslash \{A\}:$   $f_{\mathsf{Con}_k}, A, \mathcal{B}', l, E_{\mathcal{B}'}(k, S_{TTP}(k)), \mathsf{Con}_k$
```

Before t (t is specified by Alice during the first message in the main protocol and it is resent together with Alice's signature during the recovery request) the recovery cannot be initiated.

If someone tries to execute a recovery before t, which is resent together with Alice's signature during the recovery request, the TTP sends a message to notify that the recovery cannot yet be initiated.

When the recovery protocol is initiated the first time after t, the TTP uses ftp get to fetch the set $\mathcal{B}'$ at Alice. For this purpose Alice maintains a read-only accessible directory containing the set of users and her signature on this set. If the directory is not accessible the TTP supposes $\mathcal{B}' = \emptyset$.

As soon as the TTP made the ftp get on Alices's public directory, Alice considers that a recovery is in execution and stops execution of the main protocol. Otherwise a malicious B_i could be inserted in $\mathcal{B}'$ after the ftp get, and B_i could benefit of a race condition to cheat Alice.

Now the TTP sends to Alice the confirmation of receipt of the key that may be used to substitute $\mathsf{EOR}_{i,k}$ for all i such that $B_i \in \mathcal{B}'$. The TTP sends to all receiver $B_i \in \mathcal{B}'$ the confirmation of the key, that substitutes EOO_k, as well as the signed key ciphered for $\mathcal{B}'$ using a group encryption scheme.

If a receiver $B_i \notin \mathcal{B}'$ wants to perform a recovery, after the recovery has already been performed for the first time (the TTP uniquely identifies each protocol run by l and A), the TTP sends to this entity the same message as to each $B_i \in \mathcal{B}'$. This message is however useless as k has been ciphered for the set $\mathcal{B}'$ and only informs the recipient that he does not belong to this set.

5.4 Dispute Resolution

At the end of a successful protocol execution, each recipient $B_i \in \mathcal{B}'$ and Alice receive the following non-repudiation of origin respectively receipt evidence for message m:

- $\mathsf{NRO} = (\mathsf{EOO}, \mathsf{EOO}_k)$ or $\mathsf{NRO} = (\mathsf{EOO}, \mathsf{Con}_k)$
- $\mathsf{NRR}_i = (\mathsf{EOR}_i, \mathsf{EOR}_{i,k})$ or $\mathsf{NRR} = (\mathsf{EOR}_i, \mathsf{Con}_k)$

Two kinds of disputes can arise: repudiation of origin and repudiation of receipt. Repudiation of origin arises when a recipient B_i claims having received a message m from Alice, who denies having sent it. Repudiation of receipt arises when Alice claims having sent a message m to a recipient B_i who denies having received it.

Repudiation of Origin. When Alice denies the origin of the message, B_i has to present to the judge EOO, EOO_k or Con_k, l, m, k, $\mathcal{B}$ and $\mathcal{B}'$.

The message m and the key k have to be sent to the judge via a secure channel, for example using encryption. Otherwise a recipient $B_i \in \mathcal{B} \backslash \mathcal{B}'$ can recover the transmission. If any of these informations cannot be provided the recipient's claim is rejected.

The judge validates the recipient's claim if he can successfully verify that:

- $\mathsf{EOO} = S_A(f_{\mathsf{EOO}}, \mathcal{B}, l, h(c))$ after having computed $c = E_k(m)$ and $h(c)$,
- $\mathsf{EOO}_k = S_A(f_{\mathsf{EOO}_k}, \mathcal{B}', l, h(k))$ or $\mathsf{Con}_k = S_{TTP}(f_{\mathsf{Con}_k}, A, \mathcal{B}', l, k)$,
- $l = h(m, k)$.

Repudiation of Receipt. When B_i denies receipt of m, A can prove his receipt of the message by presenting EOR_i, $\mathsf{EOR}_{i,k}$ or Con_k, l, m, k and $\mathcal{B}'$ to a judge. As above a secure channel is needed to transmit m and k.

To accept Alice's claim, the judge verifies that

- $\mathsf{EOR}_i = S_{B_i}(f_{\mathsf{EOR}}, A, l, h(c))$ after having computed $c = E_k(m)$ and $h(c)$,
- $\mathsf{EOR}_k = S_{B_i}(f_{\mathsf{EOR}_k}, A, l, h(k))$ or $\mathsf{Con}_k = S_{TTP}(f_{\mathsf{Con}_k}, A, \mathcal{B}', l, h(k))$,
- $l = h(m, k)$.

5.5 Fairness and Timeliness

Our generalized protocol provides strong fairness. In fact when the main protocol ends without problems, the non-repudiation evidences have been exchanged and all receivers in $\mathcal{B}'$ got the message m. When a problem arises both Alice and Bob can initiate the recovery protocol at each moment following the transmission of message 1. As outlined in the previous section, Alice maintains a read-only accessible directory that is consulted by the TTP during the recovery to get the description of the set $\mathcal{B}'$. The TTP then sends the missing evidences to both Alice and the entities in $\mathcal{B}'$.

If Alice tries to cheat by submitting a wrong key k', the generated evidences will not be valid as $l \neq h(m, k')$. If Alice also transmits $l = h(m, k')$, the evidences will be generated for the message $m' = D_{k'}(c)$ and Alice can only prove that B_i received m'. Alice could also try to cheat by publishing a wrong set $\widetilde{\mathcal{B}'}$. Publishing a set smaller than $\mathcal{B}'$ does not provide an advantage as Alice would not receive the confirmation of receipt of the key for the entities in $\mathcal{B}' \backslash \widetilde{\mathcal{B}'}$. If $\widetilde{\mathcal{B}'}$ is bigger than $\mathcal{B}'$, Alice would harm herself as all the entities in $\widetilde{\mathcal{B}'} \backslash \mathcal{B}'$ receive k with a confirmation of origin for k, while Alice does not have an evidence of receipt for the cipher of those entities. Hence Alice does not have any interest in publishing a different set.

Now consider a scenario where at the second step several receivers send the evidence of receipt. After a fixed amount of time, Alice continues the protocol. Now, if a group of late receipts arrive at Alice, she will possess an evidence of receipt for the cipher c from these receivers. However fairness is not threatened by this scenario as Alice will not receive a confirmation of the receipt for the key from those entities. If Alice would include these receivers in $\mathcal{B}'$, they would also receive k and the corresponding evidences of origin. So strong fairness is still provided.

We shall now show that timeliness is also respected. Alice has two possibilities: either she finishes the main protocol or she has to initiate a recovery protocol. The timeliness in the later case is assured by the resilient channels. A recipient B_i can either finish the main protocol or launch a recovery. If he launches a recovery several cases may arise. He may be too early (before t) and he has to relaunch the recovery after t (t has been known a priori when the recipient agreed to continue the protocol). When B_i successfully initiates the protocol, the TTP will send him the ciphered key with the corresponding confirmation of origin. As the channels between the TTP and $B_i \in \mathcal{B}$ are resilient the protocol finishes after a finite time preserving timeliness.

6 Conclusion

We have defined multi-party non-repudiation and presented a generalization of an optimistic two-party non-repudiation protocol to an n-party non-repudiation protocol. To the best of our knowledge this is the first *optimistic* multy-party non-repudiation protocol. We have shown that the generalized protocol provides strong fairness and respects timeliness.

Acknowledgments. The authors would like to thank the anonymous referees for their helpful comments on the draft version of this paper.

References

1. N. Asokan. *Fairness in Electronic Commerce.* PhD thesis, University of Waterloo, May 1998.
2. N. Asokan, B. Baum-Waidner, M. Schunter, and M. Waidner. Optimistic synchronous multi-party contract signing. Research Report RZ 3089, IBM Research Division, Dec. 1998.
3. N. Asokan, M. Schunter, and M. Waidner. Optimistic protocols for multi-party fair exchange. Research Report RZ 2892 (# 90840), IBM Research, Dec. 1996.
4. N. Asokan, M. Schunter, and M. Waidner. Optimistic protocols for fair exchange. In T. Matsumoto, editor, *4th ACM Conference on Computer and Communications Security*, pages 6, 8–17, Zurich, Switzerland, Apr. 1997. ACM Press.
5. F. Bao, R. Deng, K. Q. Nguyen, and V. Vardharajan. Multi-party fair exchange with an off-line trusted neutral party. In *DEXA'99 Workshop on Electronic Commerce and Security*, Florence, Italy, Sept. 1999.
6. M. Ben-Or, O. Goldreich, S. Micali, and R. L. Rivest. A fair protocol for signing contracts. *IEEE Transactions on Information Theory*, 36(1):40–46, 1990.
7. C. Boyd. Some applications of multiple key ciphers. In C. G. Günther, editor, *Advances in Cryptology—EUROCRYPT 88*, volume 330 of *Lecture Notes in Computer Science*, pages 455–467, Davos, Switzerland, May 1988. Springer-Verlag.
8. G. Chiou and W. Chen. Secure broadcasting using the secure lock. *IEEE Transactions on Software Engineering*, 15(8):929–934, Aug. 1989.
9. M. Franklin and G. Tsudik. Secure group barter: Multi-party fair exchange with semi-trusted neutral parties. *Lecture Notes in Computer Science*, 1465, 1998.
10. T. Hwang. Cryptosystem for group oriented cryptography. In I. B. Damgård, editor, *Advances in Cryptology—EUROCRYPT 90*, volume 473 of *Lecture Notes in Computer Science*, pages 352–360, Aarhus, Denmark, May 1990. Springer-Verlag, 1991.
11. S. Kremer and O. Markowitch. A multi-party non-repudiation protocol. In *SEC 2000: 15th International Conference on Information Security*, IFIP World Computer Congress, pages 271–280, Beijing, China, Aug. 2000. Kluwer Academic.

12. S. Kremer and O. Markowitch. Optimistic non-repudiable information exchange. In J. Biemond, editor, *21st Symp. on Information Theory in the Benelux*, pages 139–146, Wassenaar (NL), May25-26 2000. Werkgemeenschap Informatie- en Communicatietheorie, Enschede (NL).
13. O. Markowitch and Y. Roggeman. Probabilistic non-repudiation without trusted third party. In *Second Conference on Security in Communication Networks'99*, Amalfi, Italy, Sept. 1999.
14. A. J. Menezes, P. C. van Oorschot, and S. A. Vanstone. *Handbook of applied cryptography.* CRC Press series on discrete mathematics and its applications. CRC Press, 1996. ISBN 0-8493-8523-7.
15. J. Zhou. *Non-repudiation.* PhD thesis, University of London, Dec. 1996.
16. J. Zhou, R. Deng, and F. Bao. Evolution of fair non-repudiation with TTP. In *ACISP: Information Security and Privacy: Australasian Conference*, volume 1587 of *Lecture Notes in Computer Science*, pages 258–269. Springer-Verlag, 1999.
17. J. Zhou and D. Gollmann. Observations on non-repudiation. In K. Kim and T. Matsumoto, editors, *Advances in Cryptology—ASIACRYPT '96*, volume 1163 of *Lecture Notes in Computer Science*, pages 133–144, Kyongju, Korea, 3–7 Nov. 1996. Springer-Verlag.
18. J. Zhou and D. Gollmann. An efficient non-repudiation protocol. In *PCSFW: Proceedings of The 10th Computer Security Foundations Workshop.* IEEE Computer Society Press, 1997.