Noname manuscript No.
(will be inserted by the editor)

Fair Multi-Party Non-Repudiation Protocols *

Steve Kremer, Olivier Markowitch
e-mail: {skremer,omarkow}@ulb.ac.be

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

The date of receipt and acceptance will be inserted by the editor

Abstract In this paper we introduce the concept of multi-party non-repudiation protocols. We define the aims of multi-party non-repudiation protocols, compare them to multi-party fair exchange and show some fundamental differences. We present two protocols, the first one using an online trusted third party and the second one using an offline trusted third party. We also show how to modify each of these protocols in order to provide confidentiality. The definitions and the resulting protocols are more general than the only comparable work, a multi-party certified e-mail protocol.

Key words Non-repudiation, fair exchange, group communications.

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, both Alice and Bob receive evidence showing that the other party has participated in a part or the whole of this communication.

Therefore a non-repudiation protocol, during which Alice sends a given message to Bob, has to generate non-repudiation of origin evidences intended for 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

* Partial short versions of this work have previously been published in [10] and in [12].

with respect to the protocol that generated them, the underlying public key infrastructure, ... and take a decision in favor of one of the parties.

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 evidences are based on digital signatures. A digital signature, that can be used as a non-repudiation evidence, provides a link between a message and a public verification key. A certification authority assures the correspondence between this public signature verification key and an identity. The problem of correctly maintaining non-repudiation evidences and methods to correctly manage signature keys and the related evidences are studied in [14, 16, 20]. In this paper we only focus on the *protocols* distributing the evidences.

The aim of a non-repudiation protocol is to provide non-repudiation of origin and non-repudiation of receipt, with respect to a given message. There are different ways to consider the exchange of the evidences. Either the recipient already knows the message before the exchange protocol starts, and he can thus refuse to run the protocol for this message, or the recipient has to send a non-repudiation of receipt evidence, as soon as he gets to know the message. In the latter case, the exchange of the message and the non-repudiation of origin evidence against a non-repudiation of receipt evidence has to be fair in many usual applications. We say that a non-repudiation protocol is fair if either Alice receives a non-repudiation of receipt evidence and Bob receives the message and the corresponding non-repudiation of origin evidence or none of them obtains any valid evidence. Fair non-repudiation protocols are the ones traditionally studied in literature. Throughout the remaining of the paper, we assume that non-repudiation protocols always refer to fair non-repudiation protocols.

First solutions to the problem of realizing exchanges in a fair way 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 [1,4]: 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 offline TTP. The most complete non-repudiation protocols with offline TTP have been presented in [17] and independently in [11]. 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 (or an abort protocol in the two-party case) when a problem arises.

All of the previously proposed non-repudiation protocols are two-party protocols. In the framework of fair exchange and contract signing protocols first efforts have been made to generalize the two-party protocols to the case of n participants [2,3,5,9]. Although these first works on fair exchange and contract signing protocols exist, the design of multi-party non-repudiation protocols remains interesting for several reasons. First of all, when we consider more than two entities, the differences between fair exchange protocols or digital contract signing protocols and non-repudiation protocols are accentuated. For instance, in a multi-party digital contract signing protocol, with n participants, each participant aims to obtain the $n-1$ signatures of all the other participants. In comparison, a multi-party non-repudiation protocol is similar to the needs of a multi-party certified e-mail protocol. Indeed, in a multi-party non-repudiation or certified e-mail protocol, all of the receivers await one message and one evidence of origin for this given message, while the sender awaits an acknowledgment from each of the receivers. A receiver does not await any evidence from the other receivers. Multi-party non-repudiation protocols also emphasize the difference between a mere digital signature and a non-repudiation evidence. As explained in the next section, the identity of the recipient of a non-repudiation evidence must make part of the evidence itself. Hence, it is not possible to forward evidence to a malicious participant who quit the protocol before its end. Although this also holds in two-party non repudiation protocols, the fact is more obvious in the multi-party protocols, as the message is initially destined to several recipients. Note that we only ensure that the non-repudiation evidence cannot be forwarded to another person. Of course, even in the protocols providing confidentiality, the message content can always be leaked by a coalition of an entity terminating the protocol correctly, forwarding the message to an entity having stopped the protocol before. Obviously it is impossible to design a protocol that prevents an entity knowing a secret from forwarding this secret to someone else. Hence we only aim for an exchange of the non-repudiation evidences in a fair way. The message, without a valid non-repudiation of origin evidence, could be received by entities not sending an acknowledgment. In the confidential version of our protocols, an entity not responding cannot learn the content of the message, unless someone having received the message is willing to share his knowledge. Therefore, we suppose that an entity having received the message and having sent an acknowledgment for the message, will never forward the message to an entity not having successfully finished the protocol.

In this paper we present two multi-party non-repudiation protocols. The first protocol is based on an online TTP. The TTP intervenes in each protocol run but only has to publish signed evidences. The second protocol uses an offline TTP. If no error occurs, the TTP does not intervene at all. 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 protocols are more general: as will be outlined later in more details, the certified mail protocol only continues if the whole set of participating entities is willing to do so. Our protocols permit a subset of participating entities, wishing to complete the protocols, to continue in such a way that they remain fair for all entities involved at the beginning of the protocols. Our protocols can also behave as their certified mail protocol does, i.e. stop if not all involved entities are willing to complete the protocol.

We start discussing two-party non-repudiation. We outline their goals, define the properties they have to respect and the different channel qualities considered in literature. Then, we define the multi-party non-repudiation problem, showing some fundamental differences with multi-party fair exchange. We generalize the properties required by two-party protocols to the case of multi-party protocols. We go on presenting two protocols. The first protocol uses an online TTP, while the second protocol only needs an offline TTP. For each of the protocols we discuss the generated evidence and the required properties. Before concluding, we show how to add confidentiality in our protocols and give an example of an existing group encryption scheme that provides an efficient implementation of our protocols.

2 Non-repudiation

2.1 Non-repudiation evidences

Many non-repudiation protocols have recently been designed to assure non-repudiation services in a two party communication [11,15,17,19]. The primary goal of a non-repudiation protocol is to produce the following evidences:

- non-repudiation of origin evidence
- non-repudiation of receipt evidence

The non-repudiation of origin evidence links a message, its originator and its recipient. It must be impossible to reuse the same evidence for different messages or recipients. Otherwise, the recipient Bob could transmit Alice's evidence to Charlie who could falsely prove having received the message from Alice. In the same way, a non-repudiation of receipt evidence links the recipient and the originator to the message.

It is crucial to distinguish the non-repudiation property provided by digital signatures and non-repudiation evidences. For instance, Alice's digital signature on a message m means that Alice is at the origin of the message. However, a non-repudiation evidence includes the identity of the person intended to receive the proof. For example, a non-repudiation of origin evidence proves that Alice sent a given message to Bob, while the mere signature on this message only proves that Alice is at the origin of the message without specifying any recipient.

With correct non-repudiation evidences, if Alice, denies having sent a message to Bob, or if Bob denies having received it, the non-repudiation of origin, respectively receipt evidences can be presented to an adjudicator, who can evaluate these evidences, in the context they have been generated in, and decide to either accept or reject Alice's and Bob's claims.

2.2 Properties of non-repudiation protocols

It should be noticed that the hereunder definitions suppose perfect cryptography. In order to be more close to real cryptographic primitives, one should introduce probabilities, and accept security flaws if they can only appear with negligible probability. For the sake of readability, we prefer to abstract these technical details.

Classically, two-party protocols have to respect three properties: viability, fairness and timeliness.

Definition 1 *A two-party non-repudiation protocol is said to be* viable *if, independently of the communication channels quality (cf. subsection 4), there exists an execution of the protocol, where the exchange succeeds.*

Viability discards protocols that could be fair and respect timeliness but do never achieve an exchange of the evidences, e.g. protocols which do nothing.

Definition 2 *A two-party non-repudiation protocol is said to be* fair *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.*

Fairness assures that neither Alice nor Bob can cheat the other person, and gain the ability to prove origin or receipt of a message without the other having his/her respective evidence.

Definition 3 *A two-party non-repudiation protocol provides* timeliness *if Alice and Bob always have the ability to reach, in a finite amount of time, a point in the protocol, where they can stop the protocol, while preserving fairness.*

Timeliness assures that an entity does not need to keep open protocol runs for an infinite amount of time. Such a situation could occur if an entity is not sure whether the other entity finished the protocol or not. It must always be possible for an entity to either successfully finish or to quit a protocol, without giving the possibility to the other entity to gain any advantage.

A fourth property which is not always required in non-repudiation protocols is confidentiality of the message. In plain non-repudiation protocols, the sent message itself does not have any value, if the corresponding non-repudiation evidence cannot be obtained. However, in some applications confidentiality of the message can be a desirable property.

Definition 4 *A two-party non-repudiation protocol provides* confidentiality *of the message sent by Alice to Bob, if an intruder, to obtain the message, has to directly receive it from Alice or Bob or has to break the underlying cryptographic primitives used in the protocol.*

Of course, Alice and Bob are always able to directly transfer the message to anyone. This cannot be prevented by any protocol. Hence, we only consider confidentiality inside the protocol. If there is a risk that entities inside of the protocol terminate the protocol correctly and transfer the message to other entities, not having correctly finished the protocol, we can configure our protocol so that either all the participants present at the beginning of the protocol complete the exchange, or none of them receives anything. With such an "all-or-nothing" behavior, the confidentiality is ensured with respect to all entities participating in the protocol.

3 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 distinguish between single-unit and multi-unit exchanges. Moreover different topologies are possible: [5] and [9] 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 [2] and [3].

A fundamental difference between non-repudiation and fair exchange protocols is the following. In a fair non-repudiation protocol, 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 protocol each entity offers an a priori known item (i.e. something is known about the item a priori but not its precise content) 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. Although other possibilities for generalization exist (many-to-one, many-to-many), they seem to be less natural.

The expectations we have towards multi-party non-repudiation protocols are rather similar to the properties required in two-party non-repudiation. Particularly, we have to redefine fairness.

Definition 5 *A multi-party non-repudiation protocol is said to be* viable *if, independantly of the communication channels quality (cf. subsection 4), there exists an execution of the protocol, where the exchange between all entities succeeds.*

Definition 6 *A multi-party non-repudiation protocol is said to be* fair *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.*

Here we can clearly see the difference with the certified e-mail protocol proposed by Asokan et al [2]. 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. This means that a single entity has the power to stop the protocol for all other entities. Our definition of fairness only enables a recipient to stop the protocol between himself and the originator. However we will show that our protocols can be easily modified to provide the same service as the proposed certified e-mail protocol.

Definition 7 *A multi-party non-repudiation protocol provides* timeliness *if* each *participating entity always has the ability to reach, in a finite amount of time, a point in the protocol, where it can stop the protocol, while preserving fairness.*

Definition 8 *A multi-party non-repudiation protocol provides* confidentiality *of the message sent by Alice to a set of recipients, if an intruder, to obtain the message, has to directly receive it from Alice or any of the recipients or has to break the underlying cryptographic primitives used in the protocol.*

4 Cryptographic primitives and communication channels

The non-repudiation protocols described hereafter use some cryptographic primitives which we consider as black boxes, i.e. we suppose perfect cryptography. We now give some definitions of these primitives as we use them in the paper. Note that these definitions are not the most general ones and are contextual definitions.

Definition 9 *An* encryption scheme *consists*[1] *of:*

- *a ciphering function* $E : \mathcal{M} \times \mathcal{K} \to \mathcal{C}$ *where* $\mathcal{M}$ *is the finite set of plaintexts,* $\mathcal{K}$ *is the finite set of keys and* $\mathcal{C}$ *is the finite set of ciphertexts;*
- *a deciphering function* $D : \mathcal{C} \times \mathcal{K} \to \mathcal{M}$*;*

If $E_k(m)$ *is the result of applying the ciphering function on the cleartext* m *using the key* k*, and if* k' *is the corresponding deciphering key, then* $D_{k'}(E_k(m))$ *has to be equal to* m*.*

Definition 10 *A* symmetric encryption scheme *is an encryption scheme where the deciphering key can be derived from the ciphering key using a polynomial time (with respect to the key length) algorithm. This implies that both the ciphering and the deciphering keys have to remain secret.*

Definition 11 *An* asymmetric encryption scheme *is an encryption scheme where no polynomial time (with respect to the key length) algorithm able to derive the deciphering key from the ciphering key exists. The ciphering key can be public and the deciphering key has to remain secret.*

Definition 12 *A* digital signature scheme*, as used in this paper, consists of:*

- *a signature generation algorithm* $S : \mathcal{M} \times \mathcal{K} \to \mathcal{S}$ *where* $\mathcal{S}$ *is the finite set of valid signatures;*
- *a verification algorithm* $V : \mathcal{S} \times \mathcal{M} \times \mathcal{K} \to \{true, false\}$

The signature algorithm is based on the signer's secret signature key and the verification algorithm uses the corresponding signer's public verification key.

The digital signature scheme we use here is a scheme with appendix. We suppose that when a signed message is sent during a protocol, this corresponds to the sending of the elements needed to verify the signature (the signature, the message and the signer's public key certificate).

Definition 13 *A cryptographic* hash function *is a function taking an input of arbitrary length and producing an output of fixed length. This function has to be computable in polynomial time (in the input's length), one-way and collision resistant.*

In the remaining of the paper we always consider *cryptographic* hash functions, when using hashing.

Definition 14 *A* group encryption scheme *consists of:*

[1] For the sake of readability we omit the key generation algorithms.

- *a ciphering function* $E : \mathcal{M} \times \mathcal{K}^n \to \mathcal{C}$ *where* $\mathcal{M}$ *is the finite set of plaintexts,* $\mathcal{K}$ *is the finite set of keys and* $\mathcal{C}$ *is the finite set of ciphertexts;*
- *a deciphering function* $D : \mathcal{C} \times \mathcal{K} \to \mathcal{M}$*;*

If $E_{k_1,\ldots,k_n}(m)$ *is the result of applying the ciphering function on the cleartext* m *using the keys* $k_1, \ldots, k_n$*, corresponding to* n *entities and if* k' *is the deciphering key of one of these entities, then* $D_{k'}(E_{k_1,\ldots,k_n}(m))$ *has to be equal to* m*.*

Definition 15 *A* multicast *consists in a sending from one entity to a set of entities. This kind of sending corresponds to a one-to-many communication and not to a one-to-any communication (i.e. broadcasting).*

In the framework of non-repudiation and fair exchange protocols, we can distinguish three classes of communication channels: unreliable channels, resilient channels and operational channels.

Definition 16 *An* unreliable *channel is a channel on which data may be delayed and even lost.*

Definition 17 *A* resilient *channel is a channel that delivers data correctly after a finite, but unknown amount of time.*

On a resilient channel, data may be delayed, but will eventually arrive.

Definition 18 *An operational channel delivers data correctly before a known and constant amount of time.*

Operational channels make very strong assumptions on the network services and are rather unrealistic in heterogeneous networks.

5 A multi-party protocol with online TTP

Now we give a description of a multi-party non-repudiation protocol with an online trusted third party. Although the trusted third party intervenes in each protocol execution, it is only used to publish evidences. The here presented protocol is a generalization of the two-party protocol presented by Zhou and Gollmann in [18].

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 Notation

The following notation is used to present the protocol.

- $X \to Y$: transmission from entity X to entity Y
- $X \Rightarrow \mathcal{E}$: multicast from entity X to the set of entities $\mathcal{E}$.

- $X \leftrightarrow Y$: ftp get operation performed by X at Y.
- $h()$: a cryptographic 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
- c: the result of the symmetric-key encryption function applied on the message m under the key k
- k: the key used by A to cipher m
- $\mathcal{B}$: the set of receiving entities.
- $\mathcal{B}'$: 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
- t: a time-out, chosen by Alice, before which the TTP has to publish some information[2]
- f: a flag indicating the purpose of a message ($f_{\mathsf{EOO}}, f_{\mathsf{EOR}}, f_{\mathsf{Sub}}$ and f_{Con} being flags indicating respectively a message from the originator, the answer from the recipient, the submission's message to the TTP and the TTP's confirmation).

The protocol generates the following evidences.

- $\mathsf{EOO} = S_A(f_{\mathsf{EOO}}, \mathcal{B}, l, t, h(c))$: the evidence of origin for the cipher c.
- $\mathsf{EOR}_i = S_{B_i}(f_{\mathsf{EOR}}, A, l, t, c)$: the evidence of receipt from the recipient B_i for the cipher c.
- $\mathsf{Sub}_k = S_A(f_{\mathsf{Sub}}, \mathcal{B}', l, t, k)$: the evidence of submission of the key k.
- $\mathsf{Con}_k = S_{TTP}(f_{\mathsf{Con}}, A, \mathcal{B}', l, t, k)$: the evidence of confirmation of the key k.

5.2 Protocol description

Protocol 1 A multi-party non-repudiation protocol with online TTP

1. A $\Rightarrow \mathcal{B}$: $f_{\mathsf{EOO}}, \mathcal{B}, l, t, c, \mathsf{EOO}$
2. $\mathsf{B_i} \rightarrow$ A: $f_{\mathsf{EOR}}, A, B_i, l, \mathsf{EOR}_i$
   where $\mathsf{B_i} \in \mathcal{B}$ and $i \in \{1, \ldots, |\mathcal{B}|\}$
3. A $\rightarrow$ TTP: $f_{\mathsf{Sub}}, \mathcal{B}', l, t, k, \mathsf{Sub}_k$
4. $\mathsf{B}'_\mathsf{i} \leftrightarrow$ TTP: $f_{\mathsf{Con}}, A, \mathcal{B}', l, k, \mathsf{Con}_k$
   where $B'_i \in \mathcal{B}'$ $\forall i : 1 \leq i \leq |\mathcal{B}'|$
5. A $\leftrightarrow$ TTP: $f_{\mathsf{Con}}, A, \mathcal{B}', l, k, \mathsf{Con}_k$

[2] Note that, in this protocol, there is no need to synchronize clocks between the participant. Potential differences between clocks cannot threaten the security or fairness of the protocol.

The idea of the protocol is closely related to the Zhou-Gollmann protocol. However in comparison to n parallel executions of the Zhou-Gollmann protocol our multi-party protocol is more efficient, where $n = |\mathcal{B}|$ is the number of recipients. On one hand we use one multi-casting in the first message, instead of n transmissions. On the other hand we dramatically decrease the number of digital signatures that have to be produced by Alice, respectively the TTP. Instead of generating n, respectively $n' = |\mathcal{B}'|$, signatures merely one signature is produced. Signatures may require a lot of computing power and hence the workload of the TTP is considerably reduced, which is of great important to decrease the risk of a computational bottleneck at the TTP.

A detailed description of our protocol is given in protocol 1. At each step of the protocol the entities verify the validity of the received signature(s) and information before deciding to complete the next step. The protocol can be divided into two phases. During the first phase (messages 1 and 2), Alice and a subset of the recipients commit to the exchange. During the second phase (messages 3 to 5), the exchange of the message and the evidences is finalized. In the first phase Alice sends the cipher c of the message and the evidence of origin of the cipher EOO to each of the receivers. The message also contains a label l, used in conjunction with the identity of Alice to identify the protocol run, and the time-out t. The time-out is used to guarantee protocol termination. Some or each of the receivers respond with an evidence of receipt for the cipher EOR_i. The set of receivers having done so is denoted as $\mathcal{B}'$. Note that Alice decides herself a suitable time to start the second phase. Every receipt arriving after the second phase has been initiated, is not considered any more and will, as we shall see, bring no valuable information to Alice. The second phase consists in sending the decryption key to the TTP together with the signed submission evidence Sub_k. Alice can always decide to stop the protocol before starting the second phase if not all of the recipients have answered. In that case it provides the same service as the certified e-mail protocol in [2]. When the TTP receives the key, before the time-out t, the TTP publishes the key with a signed confirmation of the key in a publicly accessible read-only directory. If message 3 arrives after t, the TTP ignores it. The evidence generated by the TTP proves that the key is available to each $B_i \in \mathcal{B}'$, as well as the key's origin.

It is however possible for Alice to contact the TTP with a Sub'_k, which is different from the one transmitted at the first step of the protocol. However this would generate inconsistent evidences for Alice.

Note also that a recipient $\widetilde{B} \in \mathcal{B} \setminus \mathcal{B}'$ could get the key (and then the message) and the confirmation evidence at the TTP. However, this is not a problem as the evidence is not valid for him (the set $\mathcal{B}$ is specified in the evidence), and we do not worry about confidentiality of the message in this first version of the protocol.

5.3 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 and receipt evidence, respectively, for the message m:

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

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 an originator A, who denies having sent it. Repudiation of receipt arises when the originator A claims having sent a message m to a recipient B_i who denies having received it.

5.3.1 Repudiation of origin. When Alice denies being at the origin of a message, B_i has to provide EOO, Con_k, $\mathcal{B}$, $\mathcal{B}'$ and (m, c, k, t, l) to the judge.

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

- A's signature on EOO,
- the TTP's signature on Con_k,
- that $B_i \in \mathcal{B}'$,
- that $l = h(m, k)$,
- that c corresponds to the cipher of m under key k.

5.3.2 Repudiation of receipt. When B_i denies the receipt of message m, Alice can prove the receipt of the message by presenting EOR_i, Con_k, $\mathcal{B}'$ and (m, c, k, t, l) to a judge, who verifies

- B_i's signature on EOR_i,
- the TTP's signature on Con_k,
- that $B_i \in \mathcal{B}'$,
- that $l = h(m, k)$,
- that c corresponds to the cipher of m under key k.

5.4 Viability, fairness and timeliness

We now show that the above presented protocol is viable, fair and respects the timeliness property.

It is trivial to see that the above protocol is viable. Consider the following execution, where Alice sends the first message and all recipients reply before Alice transmits the key to the TTP. Now Alice, as well as the recipients get their remaining evidences and the key at the TTP. Hence, the here described execution succeeds in exchanging the message and the evidences and so the protocol is viable.

Fairness is achieved by the use of a TTP. Before the submission of k to the TTP by A, none of the involved parties has got any complete evidence of origin or receipt. This is due to the fact that non-repudiation of origin (receipt) evidences are composed of the EOO (EOR_i) evidence and of the confirmation of submission evidence issued by the TTP. As the TTP makes this last evidence available simultaneously to the originator and the recipients before time t, neither the originator nor any of the recipients can get a complete evidence without the other also doing so.

There is a delicate case that should be discussed. Consider a scenario where at the second step several receivers send the evidence of receipt. Alice will start the second phase and contact the TTP. If a group of late receipts arrive at Alice, she will possess an evidence of receipt for the cipher c from these receivers. However the evidence published by the TTP will contain the set of receivers able to decipher the key. As the non-repudiation of receipt evidence is composed by both the receipt of the cipher and the receipt of the key, provided by the TTP, it is only complete if the recipient originating the cipher receipt is included in the confirmation of the key. Thus the evidences of receipt arriving too late do not provide valuable information to the originator.

When a different key k' is submitted to the TTP during the second phase of the protocol, the generated evidences, both for Alice and the recipients, will be invalid, as $l = h(m, k) \neq h(D_{k'}(c), k')$[3].

We assure timeliness for the protocol by using a time-out (t). As the TTP will not accept to publish the key after the time-out, both parties are sure that after the time-out the protocol is finished. Moreover the time-out t is proposed in the first message by A. By responding to the first message B_i accepts the proposed time-out. As the channels between the TTP and respectively A and B_i are resilient the protocol will always be finished before a finite amount of time.

Note that, in this protocol, there is no need to synchronize clocks between the participants. Potential clock shifts cannot threaten the fairness of the protocol. Indeed, if due to synchronization problems, the TTP considers that the third message arrives after t (contrary to Alice's clock) this message is ignored and nobody obtains its expected evidences. With a variation in the opposite direction, it is necessary to wait to obtain information from the TTP, but information will be available for everyone before a finite amount of time, i.e. when the TTP's clock reaches t. Therefore, neither fairness nor timeliness are threatened.

6 A multi-party protocol with offline TTP

We propose a generalization of the optimistic two-party non-repudiation protocol presented in [11]. As in the previous section, the here presented

[3] This also prevents a situation where $c = E_{k_1}(m_1) = E_{k_2}(m_2)$, since the label is different and depends on the message as well as the key.

protocol is more efficient in terms of transmission and above all computation than the execution of several two-party protocols. 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.

In this protocol the TTP does not intervene, unless a problem is arising. We say that the protocol uses an offline TTP. The protocol is composed of a main protocol and a recovery protocol. In the faultless scenario, only the main protocol is executed. When one of the participating entities thinks that a problem occurs, he contacts the TTP, launching the recovery protocol. At each step of the protocols the entities verify the validity of the received signature(s) and information before deciding to realize the next step or to initiate a recovery.

6.1 Notations

We use the same notation as in the previous section to present the protocol. We also note:

- $E_X()$: an asymmetric-key encryption function with the public key of the entity X
- $f_{\mathsf{EOO}_k,\mathsf{EOR}_k,\mathsf{Con}_k,\mathsf{Early},\mathsf{Set}}$: purpose flags indicating respectively a message with an evidence of origin for the key, a message with an evidence of receipt for the key, a message with the TTP's signature on the key, a message from the TTP saying that it is too early to contact him and a message signed by Alice containing a set of entities with which Alice desires to continue the protocol.
- Δ: a time-out, chosen by Alice, before which a recovery cannot be realized

The generated evidences are the following.

- $\mathsf{EOO} = S_A(f_{\mathsf{EOO}}, \mathcal{B}, l, h(c))$: the evidence of origin of the cipher c
- $\mathsf{EOR}_i = S_{B_i}(f_{\mathsf{EOR}}, A, l, h(c))$: the evidence of receipt from the recipient B_i for the cipher c
- $\mathsf{Sub} = S_A(f_{\mathsf{Sub}}, \mathcal{B}, l, \Delta, E_{TTP}(A, l, k))$: the evidence of origin of the key k ciphered for the TTP, needed by the entities to ask for a recovery
- $\mathsf{EOO}_k = S_A(f_{\mathsf{EOO}_k}, \mathcal{B}', l, h(k))$: the evidence of origin of the key k
- $\mathsf{EOR}_{i,k} = S_{B_i}(f_{\mathsf{EOR}_k}, A, l, h(k))$: the evidence of receipt from the recipient B_i for the key k
- $\mathsf{Rec}_X = S_X(f_{\mathsf{Rec}_X}, A, \mathcal{B}, l)$: the evidence of a recovery request
- $\mathsf{Con}_k = S_{TTP}(f_{\mathsf{Con}_k}, A, \mathcal{B}', l, h(k))$: the evidence of confirmation of the key k
- $\mathsf{Early} = S_{TTP}(f_{\mathsf{Early}}, l)$: the evidence that a recovery request was asked too early
- $\mathsf{Set} = S_A(f_{\mathsf{Set}}, \mathcal{B}', l)$: the evidence of origin of the set $\mathcal{B}'$

6.2 Main Protocol

In the main protocol (protocol 2) 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.

Protocol 2 Main protocol

1. $\mathsf{A} \Rightarrow \mathcal{B}$: $f_{\mathsf{EOO}}, f_{\mathsf{Sub}}, \mathcal{B}, l, \Delta, c, E_{TTP}(A, l, k), \mathsf{EOO}, \mathsf{Sub}$
2. $\mathsf{B_i} \rightarrow \mathsf{A}$: $f_{\mathsf{EOR}}, A, l, \mathsf{EOR}_i$
   where $\mathsf{B_i} \in \mathcal{B}$ and $i \in \{1, \ldots, |\mathcal{B}|\}$
3. $\mathsf{A} \Rightarrow \mathcal{B}'$: $f_{\mathsf{EOO}_k}, \mathcal{B}', l, k, \mathsf{EOO}_k$
4. $\mathsf{B}'_{\mathsf{j}} \rightarrow \mathsf{A}$: $f_{\mathsf{EOR}_k}, A, l, \mathsf{EOR}_{j,k}$
   where $\mathsf{B}'_{\mathsf{j}} \in \mathcal{B}'$ and $j \in \{1, \ldots, |\mathcal{B}'|\}$

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). The identity of A and the label l are ciphered together with the key for the TTP in order to avoid the reuse of this cipher during a different protocol run. This becomes above all important when confidentiality should be provided, as it prevents using the TTP as a decryption oracle. Alice also sends a time-out Δ: a recovery can always be initiated after Δ, with respect to the TTP's clock, (hence, the here used time-out Δ and the time-out used in the previously described online protocol have different semantics). 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 e-mail presented in [2], Alice may stop the protocol if not all of the receivers in $\mathcal{B}$ have answered.

Afterwards, Alice multicasts the deciphering key to all recipients having sent message 2. Then the recipients have to answer by sending an evidence of receipt for this key. If a recipient does not receive message 3, he can initiate the recovery protocol. If Alice does not receive message 4, from all expected recipients, she can also launch the recovery protocol.

6.3 Recovery Protocol

At each moment during the main protocol Alice and the recipients have the possibility to launch the recovery protocol (protocol 3) with the TTP. The recipients in $\mathcal{B}'$ launch a recovery if Alice does not multicast the key or if the key is incoherent with the label; Alice initiates the recovery protocol, if not all recipients in $\mathcal{B}'$ send a receipt for the ciphered key. Note that the recovery protocol, when launched by Alice, does not contain any proof that a recipient is engaged in the protocol. This can be explained by the fact that the recovery protocol can also act as an abort protocol, as known in two-party protocols, in the case none of the recipients responds at the second step. However, as we will see, this does not threaten fairness for any of the participants.

Protocol 3 Recovery protocol

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

Before Δ (Δ is specified by Alice during the first message in the main protocol and is resent together with Alice's signature during the recovery request) the recovery cannot be initiated. Note that Alice could generate a different evidence Sub', containing $\Delta' < \Delta$, and launch the recovery protocol before Δ. This however gives not an advantage to Alice, as possibly fewer recipients have responded. Moreover, it does not affect the security of the protocol, as any recipient can launch the recovery at moment Δ[4].

If someone tries to execute a recovery before Δ, 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 Δ, the TTP connects to a public directory maintained by Alice and fetches (using, for example, a ftp get operation) the set $\mathcal{B}'$. The directory is a read-only publicly accessible directory containing the set of users in $\mathcal{B}'$ and Alice's signature on this set. She must maintain this directory accessible. As the communication channels are resilient, we are sure that the TTP will eventually succeed to access it. If the directory is empty the TTP supposes $\mathcal{B}' = \emptyset$.

As soon as the TTP made the ftp get on Alice's public directory, Alice considers that a recovery is in execution and stops the 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. If Alice would not

[4] Observe that the fact that a recipient is in $\mathcal{B}$, but not in $\mathcal{B}'$, does not inevitably indicate that this recipient is cheating, as a heavily loaded network could also be a source of his absence in $\mathcal{B}'$.

stop taking part in the main protocol as soon as the TTP consults its public directory, the members of $\mathcal{B}$ could take advantage of a possible shift between the TTP's clock and Alice's clock, associated to a race in the transmissions. If the TTP's clock indicates that the moment Δ arrived whereas it is not yet the case on Alice's clock, a member $\tilde{B}$ of $\mathcal{B}$ that didn't send the the second message of the main protocol, can initiate the recovery protocol. The TTP will consult Alice's public directory and will get a set $\mathcal{B}'$ not containing $\tilde{B}$. If $\tilde{B}$ sends the second message of the main protocol before the moment Δ on Alice's clock and before the third message of the recovery protocol, sent by the TTP, arrives to Alice, Alice will include $\tilde{B}$ in $\mathcal{B}'$. $\mathcal{B}'$ will be different from the one obtained by the TTP. If Alice sends the third message of the main protocol, before receiving the message sent by the TTP, $\tilde{B}$ possesses the message m and the non-repudiation of origin evidence. If $\tilde{B}$ doesn't send the last message of the main protocol, Alice will receive from the TTP a non-repudiation of receipt evidence concerning a set of recipients that does not contain $\tilde{B}$. Fairness would be broken.

Therefore, if Alice stops the execution of the main protocol as soon as the TTP made a ftp get on the public directory, this problem cannot happen. Stopping the main protocol can be seen as a kind of synchronization with the TTP, as Alice notices that the TTP considers the time-out Δ as being triggered. The semantics of the time-out Δ is rather distinct from the time-out t used in the online protocol where clocks shifts between the entities could not lead to a security problem.

When we use the ftp get operation, this should be seen as an analogy, giving the intuition of what happens. As the ftp get operation could easily be performed by a malicious entity this could lead to a denial of service attack, that minimizes the number of recipients included in the directory, without however harming the fairness of the protocol. If denial of service is an issue, a protocol with an identification phase, that informs the directory owner of the access is necessary.

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.

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 the evidence is only valid for the entities in $\mathcal{B}'$. This message only informs a recipient in $\mathcal{B}' \backslash \mathcal{B}$, that he is not a member of $\mathcal{B}'$.

Note that k has been ciphered for the set $\mathcal{B}'$ and only informs the recipient that he does not belong to this set. Moreover, k is signed by the TTP in Con_k in order, to enable the recipient to construct the non-repudiation evidence (as explained in section 6.4).

In comparison with n ($n = |\mathcal{B}|$) parallel executions of an offline two-party non-repudiation protocol, such as for instance the protocols proposed in [17,11], this multi-party protocol is more efficient. The protocol gains efficiency in terms of communication, i.e. the number of messages that is sent, as well as computational power. The main protocol only sends $2 \times n' + 2$ messages, while n parallel executions require $n + 3 \times n'$ messages. This also means that Alice only needs to compute 2 signatures, instead of $n + n'$ signatures. During the recovery protocol, the gain in terms of messages is of $n' - 1$ messages and hence also $n' - 1$ signatures. This gain of efficiency is above all important, with respect to the TTP, to prevent a potential bottleneck. Although comparing the main and recovery protocols separately cannot directly apply to real executions, as in practice the main protocol is not executed completely each time the recovery protocol is launched, and similarly, the recovery protocol is not executed during each run, these partial comparisons show that there is a significant improvement in the efficiency, in comparison to n parallel protocols.

6.4 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 $\mathsf{NRR}_i = (\mathsf{EOR}_i, \mathsf{Con}_k)$

6.4.1 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, $\mathcal{B}$, $\mathcal{B}'$ and (m, c, k, l). 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

- A's signature on EOO, after having computed $h(c)$,
- A's signature on EOO_k or the TTP's signature on Con_k,
- that $B_i \in \mathcal{B}'$,
- that $l = h(m, k)$,
- that c corresponds to the cipher of m under key k.

6.4.2 Repudiation of Receipt. When B_i denies the receipt of m, A can prove his receipt of the message by presenting EOR_i, $\mathsf{EOR}_{i,k}$ or Con_k, $\mathcal{B}'$ and (m, c, k, l) to a judge.

To accept Alice's claim, the judge verifies

- B_i's signature on EOR_i after having computed $h(c)$,
- B_i's signature on $\mathsf{EOR}_{i,k}$ or the TTP's signature on Con_k,
- that $B_i \in \mathcal{B}'$,
- that $l = h(m, k)$,
- that c corresponds to the cipher of m under key k.

6.5 Viability, fairness and Timeliness

Viability is easily shown for this protocol. It is sufficient to consider the following execution of the main protocol: Alice sends message 1, and all recipients respond, before message 3 is sent. Then Alice goes on sending message 3 to all recipients, who respond with message 4. Hence, the exchange took place between all the participants and the protocol is viable.

The protocol provides fairness. 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 each of the receivers can initiate the recovery protocol at each moment following the transmission of message 1 and the deadline Δ. As outlined in the previous section, Alice maintains a read-only publicly 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 $l = h(D_{k'}(c), k')$, 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. 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 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 Δ) and he has to relaunch the recovery after Δ (Δ has been known when the recipient agreed to continue the protocol). Otherwise, if 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.

Let us recall here that the optimistic approach considers a framework where the entities are supposed to behave, most of the time, honestly. Thus, the TTP would not intervene during the protocol. However, in spite of the potential honesty of the entities, a sent message could not arrive, which implies the execution of the recovery protocol, if this message corresponds to the fourth message of the main protocol. If the number of recipients considered by Alice is big, it can be frequent that a dishonest recipient or a failing connection requires the realization of the recovery. It can appear that the TTP is then not really offline, since the risk to refer to it can, in certain circumstances, be non negligible. Notice that the recovery protocol doesn't require the involvement of all recipients belonging to $\mathcal{B}'$ (even if all members of $\mathcal{B}'$ receive, by multicasting, a message of the TTP at the end of the recovery protocol, those that finished with success the main protocol must not necessarily wait for this message). Only Alice, the TTP and some members of $\mathcal{B}'$ will be concerned by the recovery, if recovery takes place.

7 Adding confidentiality to the protocols

In this section, we show how to alter the protocols presented in the previous sections in order to provide confidentiality. Confidentiality is not a basic requirement of non-repudiation protocols, but is of major importance for some applications. We first show how to modify the protocol with online TTP, then the protocol with offline TTP. Both protocols require a group encryption scheme, and therefore, we continue giving an example of a group encryption scheme, that is adequate for our usage.

7.1 A confidential protocol with online TTP

We add the following notation :

- $E_{\mathcal{E}}()$: a group encryption scheme E, that can be deciphered by each party $P \in \mathcal{E}$.

and alter the definition of some of the evidences in the following way :

- $\mathsf{Sub}_k = S_A(f_{\mathsf{Sub}}, \mathcal{B}', l, t, E_{\mathcal{B}'}(k))$: the evidence of submission
- $\mathsf{Con}_k = S_{TTP}(f_{\mathsf{Con}}, A, \mathcal{B}', l, t, E_{\mathcal{B}'}(k))$: the evidence of confirmation of the key k.

All other evidences are defined as in section 5.

The protocol is rather close to the original one. The only difference is that we use a group encryption scheme when Alice transmits the key in message 3 to the TTP. This means that only entities in $\mathcal{B}'$ are able to decipher the key k, and hence extract message m from the cipher c. Of course, confidentiality is only provided inside the protocol. If two entities collude, one in $\mathcal{B}'$ and the other in $\mathcal{B} \backslash \mathcal{B}'$ it is impossible to prevent the fact that

Protocol 4 A confidential multi-party non-repudiation protocol with online TTP

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

the entity having successfully completed the protocol transmits the message to the other entity. Also note, that even if the message is transmitted, this does not give a valid non-repudiation message to this entity, as the non-repudiation message contains the set of recipients $\mathcal{B}'$.

Moreover, in case of dispute, when transmitting the evidences to an adjudicator, m and k should be transmitted using a secure channel. Otherwise an eavesdropper could benefit of a dispute resolution to obtain the message m. When looking at evidences, in order to accept or reject and entity's claim, in addition to the checks indicated in section 5.3.

Note that in comparison to n parallel executions of two-party protocols, that provide confidentiality, the group encryption does not decrease the efficiency, as it is equivalent to the n encryptions needed in the confidential two-party protocols. So, we still have the advantages in terms of messages and signatures without paying additional cost for confidentiality, with respect to n two-party protocols.

7.2 A confidential protocol with offline TTP

In order to provide confidentiality, we change the protocol presented in the previous section. The evidences used in this protocol are the same as in the previous one.

Protocol 5 Main protocol

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

In the main protocol we only alter message 3. In message 3 Alice ciphers the key, so that only entities in $\mathcal{B}'$ can access the key. Hence, entities in $\mathcal{B} \backslash \mathcal{B}'$

cannot get the key k, and thus cannot read message m. If Alice ciphers a different key k', the recipients detect the difference with the label l and launch a recovery protocol.

Protocol 6 Recovery protocol

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

As the main protocol, the recovery protocol is also rather close to the original protocol. The only altered messages are messages 2 and 6, that are actually identical. In these messages, the TTP has to send a ciphered version of the key to the members in $\mathcal{B}'$. As the TTP ciphers the key k, this assures that the key is identical to the key to which Alice committed during the first message.

As for the confidential protocol with online TTP, in case of dispute a confidential channel must be used to contact the adjudicator. The remaining of the dispute protocol is similar to the non confidential version.

When comparing the main protocol and the recovery protocol separately to the performance of n two-party main, respectively recovery, protocols, there is no additional cost for adding confidentiality. However, one could argue, that in the case of n executions of two-party protocols, there could be less cases of recovery protocols, and hence the TTP would do less ciphering operations. Nevertheless the gain offered by the number of signatures and transmissions compensates this potential overhead.

7.3 Group encryption scheme

Although we use cryptography primitives as black boxes, we briefly discuss here a group encryption scheme suitable for our protocols.

Several public-key group oriented cryptosystems exist, but the right choice is crucial to the security of the protocol. Boyd presented a generalization of RSA in [7]. In a group of n entities n keys are needed: each entity i knows $n-1$ keys; he knows all keys except the i^{th} one. Encryption is possible for each subset of entities. The problem with this scheme is that it is 1-resilient: when two entities collude they can decipher every message as together they possess all keys. Hence this cryptosystem is not suitable.

We shall now present Chiou and Chen's scheme proposed in [8] and show how it can be used to instantiate our protocols. It is based on a public-key encryption scheme and on the chinese remainder theorem. This method is generic as it can use any secure public-key cryptosystem. Let n be the number of recipients ($n = |\mathcal{B}|$) and m be the number of recipients having sent a receipt for the initial cipher ($m = |\mathcal{B}'|$). We denote the public-key encryption operation as $E_{e_i}()$ and the decryption operation as $D_{d_i}()$, where e_i and d_i respectively denote the public and the private key of recipient i ($1 \leq i \leq n$). Each recipient also chooses a large integer N_i such that all N_i are pairwise relatively prime (when choosing randomly large primes or multiplications of distinct primes for example, the probability of obtaining two numbers that are not relatively prime is negligible) and $N_i > E_{e_i}(k)$. The public values for each recipient are e_i and N_i; the private value is d_i.

To cipher the key k for the m recipients (indexed form j_1 to j_m) having sent an evidence of receipt for the cipher, the originator computes

$$C_{j_i} = E_{e_{j_i}}(k) \ (1 \leq i \leq m).$$

Then he uses the chinese remainder theorem to find a solution to the following system:

$$X \equiv C_{j_i} \pmod{N_{j_i}} \ (1 \leq i \leq m).$$

As all N_i are pairwise relatively prime we are sure that a unique solution exists. We now have that $E_{\mathcal{B}'}(k) = X$.
Each recipient j_i ($1 \leq i \leq m$) can decrypt $E_{\mathcal{B}'}(k)$ by computing

$$C_{j_i} = E_{\mathcal{B}'}(k) \mod N_{j_i}$$

and

$$D_{d_{j_i}}(C_{j_i}) = D_{d_{j_i}}(E_{e_{j_i}}(k)) = k.$$

Conclusion

In this paper we have defined multi-party non-repudiation, a generalization of two-party non-repudiation. We have shown the differences between multi-party non-repudiation and multi-party fair exchange. Above all, the communication topologies of multi-party fair exchange are different to those used in multi-party non-repudiation. The most closely related work that has been realized, is a multi-party certified mail protocol. However our definitions are more general.

We presented two protocols: a protocol using an online TTP, as well as an optimistic protocol using an offline TTP. Both protocols generate evidences that can be presented to an adjudicator in case of dispute, respect fairness as well as timeliness. The presented protocols are much more efficient, than the execution of several two-party non-repudiation protocols, both in terms of communication and computation. We also show how to add confidentiality to these protocols, by using a public-key group encryption scheme.

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. 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.
11. 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).
12. O. Markowitch and S. Kremer. A multi-party optimistic non-repudiation protocol. In D. Won, editor, *Proceedings of The 3rd International Conference on Information Security and Cryptology (ICISC 2000)*, volume 2015 of *Lecture Notes in Computer Science*, pages 109–122, Seoul, Korea, 2000. Springer-Verlag.
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. C. You, J. Zhou, and K. Lam. On the efficient implementation of fair non-repudiation. *Computer Communication Review*, 28(5):50–60, Oct. 1998.
15. J. Zhou. *Non-repudiation.* PhD thesis, University of London, Dec. 1996.
16. J. Zhou and R. Deng. On the validity of digital signatures. *Computer Communication Review*, 30(2):29–34, Apr. 2000.

17. 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.
18. J. Zhou and D. Gollmann. A fair non-repudiation protocol. In *Proceedings of the IEEE Symposium on Research in Security and Privacy*, Research in Security and Privacy, pages 55–61. IEEE Computer Society,Technical Committee on Security and Privacy, IEEE Computer Security Press, May 1996.
19. 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.
20. J. Zhou and K. Lam. Securing digital signatures for non-repudiation. *Computer Communications*, 22(8):710–716, May 1999.