An Optimistic Non-Repudiation Protocol with Transparent Trusted Third Party

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

Université Libre de Bruxelles
Département d'Informatique
CP 212 Boulevard du Triomphe
1050 Bruxelles
Belgium

Abstract. In this paper we consider a new and efficient optimistic non-repudiation protocol. In a non-repudiation protocol, during which Alice wants to transmit a message to Bob, Alice has to send a non-repudiation of origin evidence to Bob (attesting that Alice is at the origin of the transmitted message), and Bob has to send a non-repudiation of receipt evidence to Alice (attesting Bob's receipt of the message). Classical solutions propose to use a trusted third party to help realizing the exchange without giving any significant advantage to one of the two parties. In an optimistic protocol, the trusted third party intervenes only in case of problems during the communication between Alice and Bob. Classically, in a situation where an error occurs, evidences that have been digitally signed by the TTP are issued. Although these evidences are distinct from those produced by Alice and Bob in a faultless case, they have the same value in case of a dispute. In this paper we propose a protocol where the TTP produces the same evidences that Alice and Bob should have produced in a faultless protocol execution (this prevents, after a succesful protocol execution, to determine whether the TTP was involved or not).

1 Introduction

Due to the overwhelming importance the Internet gained nowadays, more and more sophisticated security services are requested. Non-repudiation services are one of these new security requirements. 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. Non-repudiation of receipt is more difficult to provide: therefore Alice and Bob have to follow a protocol that assures both services. To ensure the correct exchange of the non-repudiation evidences, the protocol has to be fair. Informally, we say that a protocol is fair if at the end of the protocol execution either Alice receives a non-repudiation of receipt evidence and Bob receives a non-repudiation of origin evidence or none of them receives any valid evidence.

First complete protocols providing both non-repudiation of origin and non-repudiation of receipt have been presented by Zhou and Gollmann and by Zhang and Shi [14, 16]. Their proposals relies on a trusted third party (TTP) that has to intervene during each protocol run. Such a TTP is said to be online. In Zhou and Gollmann's protocol, the TTP plays the role of a low-weight notary: it only has to publish an evidence in a read-only public directory, accessible to both Alice and Bob. Although the TTP is low-weight it may create a communication bottleneck. Therefore Zhou and Gollmann presented a second protocol [17] based on the optimistic idea introduced for fair exchanges in [2]: he assumes that in general Alice and Bob are honest, i.e. they correctly follow the protocol, and that the TTP only intervenes, by the mean of a recovery protocol, when a problem arises. A TTP that does not necessarily intervene in each protocol run is said to be offline. Zhou makes the assumption that the channels between Alice and Bob are unreliable and that the channels between the two parties and the TTP are resilient, i.e. all messages arrive after a finite, but unknown amount of time. In [10, 15] it has been shown that with these assumptions the optimistic protocol does not provide fairness. For the protocol to be fair the channel between Alice and the TTP needs to be operational, i.e. each message arrives after a known, constant amount of time. This is however a very strong and unrealistic assumption. Solutions for optimistic non-repudiation protocols have been proposed in [10, 15] where the last one suffers of a little design problem [6].

In this paper we present an optimistic non-repudiation protocol with a *transparent* TTP. In an optimistic protocol, the trusted third party intervenes only in case of problems during the communication between Alice and Bob. Classically, in such a situation the TTP digitally signs some pieces of information which will be used as non-repudiation evidences. These evidences have the same value to an adjudicator than those produced by Alice and Bob in a faultless case. Our aim is to design a protocol, where the TTP is *transparent*. This means that at the end of the protocol, by only looking at the produced evidences, it is impossible to decide whether the TTP did intervene in the protocol execution or not. As the intervention of the TTP can be due to a network failure, rather than a cheating party, transparent TTPs can be very useful in the context of electronic com-

merce, in order to avoid bad publicity. First related works have been realized by Chen [8], Asokan et al. [3], Bao et al. [4], Boyd and Foo [5] and Markowitch and Saaednia [11]. In the context of purchase of digital goods, Liqn Chen [8] proposed a protocol using discrete logarithm based signatures, in which the client commits his signature in a verifiable way for the provider. If the client does not send his final signature after having received the item, the TTP transmits information which have the same properties as a client's final signature when combined with the earlier committed signature. However, the signature generated by the TTP in case of a problem, can be distinguished from the signature generated in a faultless case. Hence, the TTP is not transparent. The use of an *invisible TTP* was first proposed by Micali [12] in the framework of certified e-mails. Asokan et al. [3] and Bao et al. [4], proposed fair exchange protocols allowing to recover, in case of problem, the original client's signature committed earlier in the protocol rather than affidavits produced and signed by the TTP. Asokan et al.'s protocol is based on verifiable encryption, which however is computationally inefficient. Bao et al. proposed two protocols, from which the first one is inefficient, while the second one, though more efficient, has recently been broken by Boyd and Foo [5]. In the same paper, Boyd and Foo [5] proposed a fair exchange protocol for electronic payment. Their method allows to recover the original client's signature from the committed one, using designated convertible signatures [7]. They also proposed a concrete protocol based on the RSA signature scheme. However, their scheme requires an additional interactive protocol and hence is rather inefficient. The most efficient protocol for fair exchange with transparent TTP has recently been proposed by Markowitch and Saaednia [11]. The protocol is based on a specific signature scheme (inspired by the Girault-Poupard-Stern signature scheme [13]). It does not need an additional interactive protocol and is efficient considering both communication and computation. All of the here discussed proposals apply to fair exchange protocols. In this paper we aim to present the first non-repudiation protocol using a transparent TTP. Although a non-repudiation protocol could be seen as a special instance of a fair exchange protocol—an exchange of a message and a non-repudiation of origin evidence against a non-repudiation of receipt evidence—there exist several inherent differences. While in a fair exchange protocol, the description of the items to exchange is known a priori, in a non-repudiation protocol the recipient of a message does not expect a particular message (it does not know the description of the message he will obtain at the end of the protocol). Moreover Bob does not exchange an item, but only an evidence of receipt, which is generally required in fair exchanges in addition to the expected item. These differences are rather subtle, but imply more efficient solutions for non-repudiation protocols than instanciations of fair exchange protocols. The protocol, presented in this paper is based on the Markowitch-Saaednia method [11].

The paper is organized as follows. We start giving some basic definitions of the different classes of communication channels considered in this paper, and define the requirements and properties that non-repudiation protocols must respect. We go on presenting an optimistic non-repudiation protocol [10]. Then we present

our novel optimistic non-repudiation protocol with transparent TTP, emphasizing the differences with the previous one. Finally we conclude the paper.

2 Basic definitions and properties

2.1 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 correct data after a finite, but unknown amount of time. Data may be delayed, but will eventually arrive. When using an operational channel correct data arrive after a known, constant amount of time. Operational channels are however rather unrealistic in heterogeneous networks.

2.2 Requirements on non-repudiation protocols

In the rest of this paper we suppose that no party acts against its own interests. This assumption is rather natural and avoids us to deal with situations where a dishonest party, i.e. a party not following the protocol, breaks some of the underneath defined properties by adopting a behavior harming itself.

The primary property a non-repudiation protocol between Alice and Bob must provide is *non-repudiability*. To offer complete non-repudiation services, a protocol must provide non-repudiation of receipt and non-repudiation of origin.

Definition 1 (Non-repudiation of receipt). *A non-repudiation protocol provides non-repudiation of receipt, if and only if it generates a non-repudiation of receipt evidence, destined to Alice, that can be presented to an adjudicator, who can unambiguously decide whether Bob received a given message or not.*

Definition 2 (Non-repudiation of origin). *A non-repudiation protocol provides non-repudiation of origin, if and only if it generates a non-repudiation of origin evidence, destined to Bob, that can be presented to an adjudicator, who can unambiguously decide whether Alice is the author of a given message or not.*

However, the non-repudiability property is not sufficient in most cases. For a non-repudiation protocol to be practical we require that none of the involved parties will have an advantage at any given moment. The property reflecting this requirement is *fairness*. Fairness comes in three flavors: *weak* fairness, *strong* fairness and *true* fairness. Weak fairness has been introduced by Asokan et al. [1]. It guarantees that if Bob received his expected item, Alice will either receive her item or an evidence showing that Bob can have access to Alice's item. We will not enter into details of this property as it is not of great importance in non-repudiation protocols.

Definition 3 (Strong fairness). *A non-repudiation protocol provides* strong fairness *if and only if at the end of a protocol execution either Alice got the non-repudiation of receipt evidence for the message m, and Bob got the corresponding message m as well as the non-repudiation of origin evidence for this message, or none of them got any valuable information.*

Note that the non-repudiation of receipt and/or origin could consist of an evidence generated by the TTP.

Definition 4 (True fairness). *A non-repudiation protocol provides* true fairness *if and only if it provides strong fairness and, if the exchange is successful, the non-repudiation evidences produced during the protocol are independent of how the protocol is executed.*

When a non-repudiation protocol respects the true fairness property, it is impossible, by looking only at the evidences, to decide whether the TTP did intervene or not in the protocol. The TTP, if solicited, has to produce evidences indistinguishable from those issued by Alice and Bob in faultless cases. Such a TTP is transparent.

Timeliness is another property, needed in order for a protocol to be practical.

Definition 5 (Timeliness). *A non-repudiation protocol provides* timeliness *if and only if all honest parties 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 avoids situations where a party does not know whether it can stop the protocol without loosing fairness or not.

3 A non-repudiation protocol with offline TTP

3.1 Introduction

We will now present a two-party non-repudiation protocol with an off-line TTP [10]. 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-Gollmann optimistic protocol [17] 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 [15]. However Zhou et al.'s protocol suffers from a small design error, pointed out in [6].

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 and evidences

We use the following notation to describe the protocol.

- $X \rightarrow Y$: transmission from entity X to entity 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
- $E_X()$: a public-key encryption function under X's public key
- $D_X()$: a public-key decryption function under X's private key
- $S_X()$: the signature function of entity X
- m: the message sent from A to B
- k: the session key A uses to cipher m
- $c = E_k(m)$: the cipher of m under the session 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

During the protocol the following evidences are generated.

- the evidence of origin for the cipher c: $\mathsf{EOO} = S_A(f_{\mathsf{EOO}}, B, TTP, l, h(c))$
- the evidence of receipt for the cipher c: $\mathsf{EOR} = S_B(f_{\mathsf{EOR}}, A, TTP, l, h(c))$
- the evidence of submission of key k: $\mathsf{Sub} = S_A(f_{\mathsf{Sub}}, B, l, E_{TTP}(k))$
- the evidence of origin for key k: $\mathsf{EOO}_k = S_A(f_{\mathsf{EOO}_k}, B, l, k)$
- the evidence of receipt for key k: $\mathsf{EOR}_k = S_B(f_{\mathsf{EOR}_k}, A, l, k)$
- the recovery request: $\mathsf{Rec}_X = S_X(f_{\mathsf{Rec}_X}, Y, l)$
- the confirmation evidence for key k: $\mathsf{Con}_k = S_{TTP}(f_{\mathsf{Con}_k}, A, B, l, k)$
- the abort request: $\mathsf{Abort} = S_A(f_{\mathsf{Abort}}, B, l)$
- the abort confirmation: $\mathsf{Con}_a = S_{TTP}(f_{\mathsf{Con}_a}, A, B, l)$

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 that the transmission will not arrive any more and initiates either a recovery or an abort protocol.

1. $A \rightarrow B$: $f_{\mathsf{EOO}}, f_{\mathsf{Sub}}, B, TTP, l, c, E_{TTP}(k), \mathsf{EOO}, \mathsf{Sub}$
2. $B \rightarrow A$: $f_{\mathsf{EOR}}, A, TTP, l, \mathsf{EOR}$

if A times out then abort

3. $A \rightarrow B$: $f_{\mathsf{EOO}_k}, B, l, k, \mathsf{EOO}_k$

if B times out then recovery$[X := B, Y := A]$

4. $B \rightarrow A$: $f_{\mathsf{EOR}_k}, A, l, \mathsf{EOR}_k$

if A times out then 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. 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. Alice goes on sending 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. Bob replies with 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. If at some moment in the protocol, one of the messages does not arrive in a reasonable amount of time, i.e. before the local time-out, Alice or Bob execute an abort or a recovery protocol as indicated in the protocol description. For a more detailed description of this protocol, we refer the reader to [10].

3.4 Recovery protocol

To launch the recovery protocol Alice or Bob has to send to the TTP several evidences that prove to the TTP that Alice sent the cipher c to Bob and that Bob really received it, i.e. that the protocol has been started between Alice and Bob. Note that the recovery protocol can only be executed once per protocol run and is mutually exclusive with the abort protocol.

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 whether an abort protocol or a recovery protocol has already been accepted 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. If the

TTP accepts to perform a recovery protocol, the TTP sends to Alice as well as to Bob all possibly missing evidences. If the recovery protocol is executed, the key confirmation evidence Con_k signed by the TTP 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.

1. $A \rightarrow TTP:\ f_{\mathsf{Abort}}, l, B, \mathsf{Abort}$
   if *aborted* or *recovered* then stop
   else *aborted*=true
2. $TTP \rightarrow A:\ f_{\mathsf{Con}_a}, A, B, l, \mathsf{Con}_a$
3. $TTP \rightarrow B:\ f_{\mathsf{Con}_a}, A, B, l, \mathsf{Con}_a$

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, TTP, l, c, m and k. The judge verifies that:

- $\mathsf{EOO} = S_A(f_{\mathsf{EOO}}, B, TTP, l, c)$,
- $\mathsf{EOO}_k = S_A(f_{\mathsf{EOO}_k}, B, TTP, l, k)$ 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, TTP, l, c, m and k to a judge. The judge verifies that:

– $\mathsf{EOR} = S_B(f_{\mathsf{EOR}}, A, TTP, l, h(c))$,
– $\mathsf{EOR}_k = S_B(f_{\mathsf{EOR}_k}, A, l, k)$ 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

We do not discuss fairness and timeliness in detail, but will only give a short idea, why the protocol does provide those properties. A more complete discussion can be found in [10]. If Bob stops the protocol after having received the first message, Alice may perform the abort protocol, in order to avoid that Bob initiates a recovery later. As neither Bob nor Alice received complete evidences the protocol remains fair. If Bob had already initiated the recovery protocol (in that case the abort request is refused by the TTP), 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. Once the second message of the main protocol has been received by Alice, both Alice and Bob can launch a recovery protocol and force the successful exchange of the message and the non-repudiation evidences. Note that the protocol achieves strong fairness, but not true fairness. At the end of the protocol, when looking at the evidences, we can easily decide whether the TTP did intervene or not. In the case the TTP intervened in the protocol, the evidences EOR_k and possibly EOO_k have been substituted by the evidence Con_k, generated by the TTP.

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 to both 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 A non-repudiation protocol with offline transparent TTP

We will now describe the variant protocol where the offline TTP produces, when a fault occurs during the main protocol execution, exactly the same evidences than those produced by Alice and Bob in a faultless case. The protocol described underneath supposes a resilient channel between the TTP and Alice and between the TTP and Bob. The communication channel between Alice and Bob may be unreliable.

4.1 The signature scheme

The protocol uses a signature scheme based on the GPS signature scheme [9, 13]. The GPS signature of a message m is realized on one hand by choosing a random value r and computing $t = \alpha^r \bmod n$ where n is a composite modulus and α is a basis of order $\lambda(n)$, and on the other hand by computing $z = r + x \cdot h(t, m)$ where x is a secret value associated to $y \equiv \alpha^{-x} \bmod n$ the corresponding public value. The verification is achieved by comparing t and $\alpha^z \cdot y^{h(t,m)} \bmod n$.

The signature used in this protocol is issued in two phases. First the signer produces a committed signature. Then this committed signature is turned into a final signature either by the signer or by the TTP. The recipient of a committed signature is able to check whether the TTP has the ability to transform the committed signature into the signer's final signature.

During an initialization phase, the TTP chooses an integer $n = pq$, where p and q are large random strong primes (of almost the same size). The TTP chooses also a base α of order $\lambda(n)$ and a small integer c such that $\gcd(\lambda(n), c) = 1$. The TTP computes d such that $cd \equiv 1 \pmod{\lambda(n)}$ and $\beta = \alpha^c \bmod n$. Finally, the TTP makes n, β, c and α public, keeps d secret and discards p and q.

A signer u chooses a random integer x_u as secret key and computes the relative public key $y_u = \alpha^{x_u} \bmod n$.

To produce the committed signature on a message m the signer u chooses a random r_u and computes $t_u = \beta^{r_u} \bmod n$ and $z_u = c \cdot r_u + h(t_u, m) \cdot x_u$. The pair (t_u, z_u) forms the committed signature of signer u and will also be noted $\mathsf{ComSig}_u(m)$.

A verifier v can check the committed signature by comparing $\alpha^{z_u} \bmod n$ and $t_u \cdot {y_u}^{h(t_u,m)} \bmod n$.

The final signature of signer u can be computed independently by the signer u by computing $t'_u = \alpha^{r_u} \bmod n$, or by the TTP by computing $t'_u = {t_u}^d \bmod n$. The pair (t'_u, z_u) forms the final signature of signer u and will also be noted $\mathsf{FinalSig}_u(m)$.

The verifier checks the validity of the final signature by comparing $\alpha^{z_u} \bmod n$ to ${t'_u}^c \cdot {y_u}^{h({t'_u}^c \bmod n, m)} \bmod n$ (in practice, it is sufficient to verify that $t = {t'_u}^c \bmod n$).

The security of this signature scheme has been studied in [11, 13].

4.2 Evidences and notations

The notation used to describe the protocol is the same as in the previous section.

The evidences generated during the protocol are the following.

- the evidence of origin: $\mathsf{EOO} = \mathsf{ComSig}_A(f_{\mathsf{NRO}}, B, TTP, l, h(c), h(k))$
- the evidence of receipt: $\mathsf{EOR} = \mathsf{ComSig}_B(f_{\mathsf{NRR}}, A, TTP, l, h(c), h(k))$

– the non-repudiation of origin evidence:
  $\mathsf{NRO} = \mathsf{FinalSig}_A(f_{\mathsf{NRO}}, B, TTP, l, h(c), h(k))$
– the non-repudiation of receipt evidence:
  $\mathsf{NRR} = \mathsf{FinalSig}_B(f_{\mathsf{NRR}}, A, TTP, l, h(c), h(k))$
– the evidence of submission for key k: $\mathsf{Sub} = S_A(f_{\mathsf{Sub}}, B, l, E_{TTP}(k))$
– the abort request: $\mathsf{Abort} = S_A(f_{\mathsf{Abort}}, B, l)$
– the recovery request: $\mathsf{Rec}_X = S_X(f_{\mathsf{Rec}_X}, Y, l)$
– the abort confirmation: $\mathsf{Con}_a = S_{TTP}(f_{\mathsf{Con}_a}, A, B, l)$
– the error confirmation: $\mathsf{Con}_e = S_{TTP}(f_{\mathsf{Con}_e}, A, B, l)$

4.3 Main protocol

1. $A \to B$: $f_{\mathsf{EOO}}, f_{\mathsf{Sub}}, B, TTP, l, h(k), c, E_{TTP}(k), \mathsf{EOO}, \mathsf{Sub}$
2. $B \to A$: $f_{\mathsf{EOR}}, A, TTP, l, \mathsf{EOR}$

if A times out then abort

3. $A \to B$: $f_{\mathsf{NRO}}, B, l, k, \mathsf{NRO}$

if B times out then recovery$[X := B, Y := A]$

4. $B \to A$: $f_{\mathsf{NRR}}, A, l, \mathsf{NRR}$

if A times out then recovery$[X := A, Y := B]$

Alice starts the protocol by transmitting to Bob the cipher c of the message m under the session key k, the hash of this session key, the session key ciphered with the TTP's ciphering public key, the committed signature EOO (which is the committed non-repudiation of origin evidence) and the evidence of origin of the session key ciphered for the TTP.

Bob verifies the received message and checks the committed signature EOO as indicated previously and Sub. If the verification holds, Bob sends to Alice his committed signature EOR (which is the committed non-repudiation of receipt evidence).

If Alice does not receive the protocol's second message (from Bob) before a local time-out (chosen by herself), or if the received information are incorrect (the message is not well formed or Bob's committed signature is invalid) she realizes the abort protocol described below. Otherwise, she sends to Bob the session key k and the non-repudiation of origin evidence (her final signature NRO).

If the information received by Bob are correct (well formed message and valid NRO with regard to the session key k he just received), he sends the non-repudiation of receipt evidence (his final signature NRR). Otherwise, he initiates the recovery protocol, described underneath

Eventually, if Alice does not receive a correct final sending from Bob she initiates the recovery protocol.

4.4 Recovery protocol

We here consider that X is the party initiating the recovery, Y being the other party.

1. $X \rightarrow TTP$: $f_{\mathsf{Rec}_X}, f_{\mathsf{Sub}}, Y, l, h(c), h(k), E_{TTP}(k), \mathsf{Rec}_X, \mathsf{Sub}, \mathsf{EOR}, \mathsf{EOO}$
if $h(k) \neq h(D_{TTP}(E_{TTP}(k)))$ then error
if *aborted* or *recovered* then stop
else *recovered*=true
2. $TTP \rightarrow A$: $f_{\mathsf{NRR}}, A, l, \mathsf{NRR}$
3. $TTP \rightarrow B$: $f_{\mathsf{NRO}}, B, l, k, \mathsf{NRO}$

At any time after having received the first message of the main protocol, Bob can initiate the recovery protocol. Alice, after having received the second message of the main protocol, can also initiate the recovery protocol (for example if Bob does not send the fourth message of the main protocol).

The initiator of the recovery protocol sends to the TTP the hash of the ciphered message, the hash of the session key k, the session key ciphered for the TTP and the signatures Rec_X, Sub, EOR and EOO. The TTP first verifies that all the signatures are correct. If at least one signature is incorrect, the request is ignored. These checks also make it impossible for Bob of trying to recover the protocol with a wrong session key k, as the submission evidence Sub has been signed by Alice. Then the TTP verifies that the hash of the key committed in the first message of the main protocol corresponds to the key ciphered under the TTP's public key. If the keys are different, the error protocol described below is launched to inform Bob, that Alice is trying to cheat. Otherwise the TTP checks whether neither the abort nor the recovery protocol have yet been performed. If not, the TTP uses its private key d to convert the committed signatures into final ones. Then the TTP forwards the non-repudiation of receipt evidence (Bob's final signature) to Alice and the non-repudiation of origin evidence (Alice's final signature) to Bob.

4.5 Abort protocol

1. $A \rightarrow TTP$: $f_{\mathsf{Abort}}, l, B, abort$
if *aborted* or *recovered* then stop
else *aborted*=true
2. $TTP \rightarrow A$: $f_{\mathsf{Con}_a}, A, B, l, \mathsf{Con}_a$
3. $TTP \rightarrow B$: $f_{\mathsf{Con}_a}, A, B, l, \mathsf{Con}_a$

If Alice does not receive the second message of the main protocol, she initiates the abort protocol, by sending an abort request to the TTP. If the protocol have not yet been recovered or aborted, the TTP sends to both Alice and Bob a signed abort confirmation.

4.6 Error protocol

aborted=true
1. $TTP \rightarrow A$: $f_{\mathsf{Con}_e}, A, B, l, \mathsf{Con}_e$

2. $TTP \rightarrow B :$ $\quad f_{\mathsf{Con}_e}, A, B, l, \mathsf{Con}_e$

The TTP runs this error protocol if during a recovery protocol it appears that Alice provided a session key to be recovered (thanks to $E_{TTP}(k)$) different from the initially committed session key (the hash of the key is included in EOO)[1]. The goal of this protocol is to warn Bob that Alice tried to cheat (and to inform Alice that this attempt has been detected detected)[2].

4.7 Disputes resolutions

The two final signatures NRO and NRR form respectively the non-repudiation of origin and receipt evidences for message m.

Repudiation of origin. If Alice denies being the author of the message m, Bob presents NRO, TTP, l, c, m and k to an adjudicator. The judge verifies whether the NRO is valid with regard to the session label and the ciphered message provided, $l = h(m, k)$ and $c = E_k(m)$. If Bob provides the required information and all the checks hold, the adjudicator accepts the claim that Alice is at the origin of the message.

Repudiation of receipt. If Bob denies having received the message m, Alice presents to the adjudicator NRR, TTP, l, c, m and k. The judge verifies whether the NRR is valid with regard to the session label and the ciphered message provided, $l = h(m, k)$ and $c = E_k(m)$. If Alice presents all needed information and the checks hold, the adjudicator concludes that Bob received the message.

4.8 Fairness and timeliness

We start showing that our protocol provides fairness. If Bob stops the protocol after having received the first message, Alice can run the abort protocol to prevent Bob to initiate a recovery later. As neither Bob nor Alice received the non-repudiation evidences (neither NRO nor NRR), the protocol remains fair.

If Bob had already previously initiated the recovery protocol, the TTP forwards to both Alice and Bob all the possibly missing non-repudiation evidences and the protocol stays fair. If Bob is unable to run the recovery protocol, because Alice provided, at the beginning of the main protocol, a session key ciphered for the TTP which differs from the session key hashed (and signed in the EOO), the

[1] If Bob is the initiator of the recovery protocol, he cannot send a wrong ciphered session key because he has to provide a correct key submission evidence Sub, signed by Alice, at the beginning of the recovery protocol.

[2] Of course, the first message of this error protocol is optional.

TTP will launch the error protocol in order to inform Bob that Alice tried to cheat. The protocol will also end in a fair way with no evidences exchanged.

If Alice does not send the third message during the main protocol, Alice and Bob may initiate the recovery protocol. Again the protocol will end in a fair way with either all the non-repudiation evidences forwarded to Alice and Bob by the TTP, or with an error message, issued by the error protocol, and no exchanged evidences.

If Alice realizes the third step, Bob receives the non-repudiation of origin evidence. Bob can then send the fourth message of the main protocol and Alice receives the non-repudiation of receipt evidence. If Bob does not send the last message of the main protocol, Alice runs the recovery protocol, and thanks to the resilience of the channels between the TTP and both Alice and Bob, all data sent by the TTP to Alice and Bob eventually arrive. In those cases all entities receive valid evidences and the protocol finishes in a fair way.

Still consider the following scenario, where Alice tries to cheat by signing in the EOO a session key that differs from the one ciphered for the TTP. In that case Alice will never send the third message of the main protocol, in order not to harm herself. Suppose Alice sends the third message and Bob does not reply by sending the fourth message. Alice cannot perform a recovery protocol (since she is unable to provide coherent information to the TTP), and Bob will get his non-repudiation evidence, while Alice does not get her evidence. Such a behavior would contradict the assumption we made in section 2, that says that no entity acts against its own interests. Hence message 3 will not be sent in the main protocol, and the evidences will not be exchanged. Thus the protocol remains fair.

The protocol provides strong and true fairness: when looking at the evidences, no one can determine whether the TTP did intervene or not.

When looking at the timeliness, we have to consider three situations which 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 either the non-repudiation evidences or an error information (via the error protocol) after a finite amount of time because of the resilience of the channels.

5 Conclusion

We have considered a new and efficient fair optimistic non-repudiation protocol. A non-repudiation protocol between Alice and Bob, is such that at the end of the protocol, Alice has sent to Bob a message and a non-repudiation of origin evidence for this message, while Bob has sent to Alice a non-repudiation of receipt evidence for the message.

In previous optimistic non-repudiation protocols, the TTP, used during the protocol in case of problem, produces non-repudiation evidence which are distinct from the non-repudiation evidences exchanged by Alice and Bob in a faultless case. The protocol we propose is the first non-repudiation protocol which features a transparent TTP, i.e. a TTP who produces, in case of problem, the same evidences that Alice and Bob should have produced in a faultless protocol execution. In consequence, at the end of the protocol, by only looking at the exchanged non-repudiation evidences, it is impossible to decide whether the TTP did intervene in the protocol execution or not.

As it is difficult to determine whether the TTP was required during the non-repudiation protocol because of a dishonest party or because of a network problem, such as transparent TTP may be particularly relevant, for example, in an electronic commerce environment.

References

1. N. Asokan. *Fairness in Electronic Commerce*. PhD thesis, University of Waterloo, May 1998.
2. 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.
3. N. Asokan, V. Shoup, and M. Waidner. Optimistic fair exchange of digital signatures. In *Advances in Cryptology: Proceedings of Eurocrypt'98*, volume 1403 of *Lecture Notes in Computer Science*, pages 591–606. Springer-Verlag, 1998.
4. F. Bao, R. H. Deng, and W. Mao. Efficient and practical fair exchange protocols with off-line TTP. In *IEEE Symposium on Security and Privacy*, May 1998.
5. C. Boyd and E. Foo. Off-line fair payment protocols using convertible signatures. *Lecture Notes in Computer Science*, 1514:271–285, 1998.
6. C. Boyd and P. Kearney. Exploring fair exchange protocols using specification animation. In *The Third International Workshop on Information Security - ISW2000*, Lecture Notes in Computer Science, Australia, Dec. 2000. Springer-Verlag.
7. D. Chaum. Designated confirmer signatures. In A. D. Santis, editor, *Advances in Cryptology: Proceedings of Eurocrypt'94*, volume 950 of *Lecture Notes in Computer Science*, pages 86–91. Springer-Verlag, 1995, 9–12 May 1994.
8. L. Chen. Efficient fair exchange with verifiable confirmation of signatures. *Lecture Notes in Computer Science*, 1514:286–299, 1998.
9. M. Girault. Self-certified public keys. In *Advances in Cryptology: Proceedings of EuroCrypt'91*, volume 547 of *Lecture Notes in Computer Science*, pages 490–497. Springer-Verlag, 1991.
10. 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).
11. O. Markowitch and S. Saeednia. Optimistic fair-exchange with transparent signature recovery. In *5th International Conference, Financial Cryptography 2001*, Lecture Notes in Computer Science. Springer-Verlag, 2001.
12. S. Micali. Certified E-mail with invisible post offices. Available from author; an invited presentation at the RSA '97 conference, 1997.

13. G. Poupard and J. Stern. Security analysis of a practical "on the fly" authentication and signature generation. In *Advances in Cryptology: Proceedings of Eurocrypt'98*, volume 1403 of *Lecture Notes in Computer Science*, pages 422–436. Springer-Verlag, 1998.
14. N. Zhang and Q. Shi. Achieving non-repudiation of receipt. *The Computer Journal*, 39(10):844–853, 1996.
15. 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.
16. J. Zhou and D. Gollmann. A fair non-repudiation protocol. In *IEEE Symposium on Security and Privacy*, Research in Security and Privacy, pages 55–61, Oakland, CA, May 1996. IEEE Computer Society,Technical Committee on Security and Privacy, IEEE Computer Security Press.
17. 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.