An Intensive Survey of Fair Non-Repudiation Protocols

Steve Kremer

Université Libre de Bruxelles, Belgium

Olivier Markowitch

Université Libre de Bruxelles, Belgium

Jianying Zhou

Kent Ridge Digital Labs, Singapore

Abstract

With the phenomenal growth of the Internet and open networks in general, security services, such as non-repudiation, become crucial to many applications. 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 fair non-repudiation protocol has to generate non-repudiation of origin evidences intended to Bob, and non-repudiation of receipt evidences destined to Alice. In this paper, we clearly define the properties a fair non-repudiation protocol must respect, and give a survey of the most important non-repudiation protocols without and with trusted third party (TTP). For the later ones we discuss the evolution of the TTP's involvement and, between others, describe the most recent protocol using a transparent TTP. We also discuss some ad-hoc problems related to the management of non-repudiation evidences.

Key words: Network security, security protocols, exchange protocols, non-repudiation

1 Introduction

With the explosion of the Internet, electronic transactions have become more and more common. However the transactions' security is crucial to many applications, e.g. electronic commerce, digital contract signing, electronic voting, and so on. While issues such as confidentiality, authentication, access control, etc. have been studied intensively, most interest in non-repudiation protocol has only come in recent years. 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. With the birth of public-key cryptography [1] in general and digital signatures in particular, the primitives for providing non-repudiation were created. Irrefutable evidences can be based on digital signatures, supposing that an adequate public key infrastructure is used.

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 must 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 must be fair. We say that a non-repudiation protocol is *fair* if, at the end of the protocol, 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.

The challenging part of non-repudiation protocols is to avoid one of the implied entities to cheat. Consider for instance a naive protocol, where Alice sends a signed message to Bob, who replies with a signed receipt for the given message. If the two entities do not trust each other, this protocol is not applicable, as Bob may not send the second message. The protocol could be altered in the following way: Alice sends a commitment to the message to Bob, who replies with a receipt and, in a third step, Alice sends the message itself to Bob. Here, we have the dual problem, in the sense that this time it is Alice who is in an advantageous position, being the first to obtain her complete evidence, and hence could refuse to send the last message.

The importance of the fairness property has not always been well understood. Among the first non-repudiation protocols, we find the three protocols [2–4] proposed by the International Organization for Standardization. None of these protocols supports fairness.

Similar problems that also require the fairness property to be respected are fair exchange protocols, contract signing protocols and certified e-mail protocols. Historically first solutions providing fairness in exchange protocols were based on a *gradual exchange* of the expected information [5,6]. These protocols need the hypothesis that both involved parties have an equivalent, or related computing power. This is however rather unrealistic in practice. Moreover the protocols often need a great amount of transmissions. An amelioration came with the idea of *probabilistic protocols* [7,8]: the computing powers do not need to be related anymore, but the number of transmissions is still important to provide an adequate security level. Another approach to resolve the problem of fair non-repudiation is to use a *trusted third party* (TTP). First solutions using this approach are based on an *inline* TTP [9], i.e. a trusted third party acting as a delivery authority, intervening in each transmission. However, the heavy involvement of the TTP implies a communication and computation bottleneck. A first improvement to reduce the TTP involvement was the use of an *online* TTP: the TTP intervenes in each protocol run, but not in each transmission [10,11]. Protocols with a light-weight TTP have been proposed. Finally, a big step towards more efficient solutions was the introduction of *offline* TTPs. Independently, Micali and Asokan et al. [12,13], in the context of certified e-mail and fair exchange, designed a protocol where the TTP only intervenes in case of problem. This approach, using an off-line TTP, is also called the *optimistic* approach. The rationale is that in most cases the participating entities are honest and the network is well functioning, implying a protocol run without any involvement of the TTP. Only in case of a cheating entity or a network failure, the TTP intervenes to finish the protocol, either ending with no exchange taking place or forcing a successful exchange. Rapidly this approach has also been applied to non-repudiation protocols [14–16]. Recently, the notion of a *transparent* TTP has been introduced. When a transparent TTP is used, at the end of the protocol, it is impossible to decide, by only looking at the evidences, whether the TTP did intervene or not. This feature can be useful in electronic commerce. As a TTP may intervene due to a network failure, rather than a cheating entity, bad publicity can be avoided using transparent TTPs.

The aim of this paper is to survey existent two-party non-repudiation protocols, particularly the most recent evolutions of these protocols. The paper is structured as follows. We start giving definitions of all the properties a non-repudiation protocol must provide. Then we present classical protocols without TTP and also show some recently designed probabilistic protocols. The three following sections are devoted to non-repudiation protocols with trusted third party. We will study the existing protocols in chronological order and observe the evolution of the TTPs. Considered in a first time as an agent of synchronization, it will serve as a signatory in later protocols. It is also interesting to note the evolution of the needs. The first non-repudiation protocols permit that the TTP generates signatures in its proper name. These signatures, called *affidavits*, even if structurally different, have the same juridical value as the signatures which should have been produced by the entities themselves. The most recent protocols with transparent TTP ensure that the entities receive the awaited signature, i.e. the other entity's signature and not an affidavit produced by the TTP, in any circumstances. Then, we look at the key revocation problem related to non-repudiation evidences. Finally, we compare the different protocols, on criteria such as efficiency and TTP involvement. To end the paper we draw some conclusions.

2 Preliminary definitions and notations

In this section, we give some preliminary definitions, dealing with our communication model and the properties a non-repudiation protocol has to provide. We also present the notation that will be used through the remaining of the paper to present the protocols.

2.1 The communication model

Generally in literature, three classes of communication channels are considered: unreliable channels, resilient channels and operational channels. No assumptions have to be made about unreliable channels: data may be lost. A resilient channel (also called asynchronous network) delivers correct data after a finite, but unknown amount of time. Data may be delayed, but will eventually arrive. When using an operational channel (also called synchronous network) correct data arrive after a known, constant amount of time. Operational channels are however rather unrealistic in heterogeneous networks.

2.2 Properties

We suppose for the rest of the paper 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 main property a non-repudiation protocol has to respect is *non-repudiability.* A non-repudiation protocol has to offer both non-repudiation of origin and non-repudiation of receipt.

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.* □

In order for a non-repudiation protocol to be interesting in practice, we have to add the *fairness* property. Fairness insures that none of the participating entities can fool the other one (for example, if the protocol ends in a situation where Bob got a valid non-repudiation of origin evidence without Alice having got the corresponding non-repudiation of receipt evidence). Different flavors of fairness have been defined: weak, strong, true and probabilistic fairness. Weak fairness, ensures that if an entity, Alice for example, does not obtain its evidence, while the other entity, Bob, did, then Alice will receive a proof of this fact. Strong fairness is defined as follows.

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.* □

In a truely fair protocol, the generated evidences are independent of the fact whether the TTP did intervene in the protocol or not. It is impossible to decide, by only looking at the generated evidences, whether the TTP did intervene or not. As the intervention of a TTP can be due to a network failure, rather than a cheating behavior of a party, this property can be very important in a context of electronic commerce to avoid bad publicity. Achieving true fairness is equivalent to having a transparent TTP (cf definition 11). True fairness is defined as follows.

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.* □

Probabilistic fairness has been introduced for protocols without TTP where fairness is guaranteed with a given (generally high) probability.

Definition 5 (Probabilistic fairness). *A non-repudiation protocol is* ϵ-fair *if and only if the probability that 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, is $\geq 1 - \epsilon$.* □

Timeliness is a property, that is generally requested, in order for the protocol to be practical. It assures that the participating entities can always finish the protocol after a finite amount of time. It avoids situations where an entity does not know whether a protocol run is finished or not, and needs to keep an open protocol session for a potentially infinite amount of time to assure fairness.

Definition 6 (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.* □

2.3 TTP's involvement

Various types of TTP can be considered according to their involvement in the protocol.

Definition 7 (Inline TTP). *A TTP involved in each message's transmission during the protocol, is said to be* inline. □

Definition 8 (Online TTP). *A TTP involved during each session of the protocol but not during each message's transmission, is said to be* online. □

Definition 9 (Offline TTP). *A TTP involved in a protocol only in case of an incorrect behavior of a dishonest entity or in case of a network error, is said to be* offline. □

Definition 10 (Neutral TTP). *A TTP is known as* neutral *if the assistance that it brings to the successful realization of a protocol is not conditioned by its knowledge of the information to be exchanged.* □

Definition 11 (Transparent TTP). *An offline TTP producing evidences indistinguishable from the evidences Alice and Bob should have exchanged in a faultless case, is said to be* transparent. □

2.4 Notations

We now introduce the notation that will be used to describe the protocols.

- $X \to Y$: 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
- $\ell = h(m, k)$: a label that in conjunction with (A, B) uniquely identifies a protocol run
- f: a flag indicating the purpose of a message

3 Non-repudiation protocols without TTP

3.1 Introduction

Although (inefficient) protocols without TTP were the first protocols proposed in the framework of fair exchange of secrets and digital contract signing, non-repudiation protocols without TTP were initially presented at the end of the 1990s [8] (thus, curiously, much later than protocols with TTP, in contrary to fair exchange protocols).

In the middle of the 1980s, fair exchange protocols were developed in order to achieve the exchange of secrets (e.g. secret keys) between two entities. The basic idea was that each entity transmits in turn successive bits of the information to be exchanged. This process continues until the last bit of each information (both information to exchange are supposed to have the same size) was sent or until one of the two entities stops his participation in the protocol. The amount of computing necessary for each entity to retrieve the missing bits decreases at each step of the protocol. If the protocol is stopped before the information has completely been sent, and if the entities have the same computing power, there could be at most a difference of factor two in the time needed for each entity to retrieve the expected information. In order to reduce this difference of time needed to obtain the information, Tedrick [5,6] shows how to transmit the fraction of a bit: rather than sending in turn one bit of each information, the entities transmit in turn a binary string which is different from the corresponding string of the information to be transmitted.

Whatever method is used to exchange the bits of information, one has to be able to detect whether an entity attempts to cheat (by sending incorrect bits). Many methods were suggested (e.g. based on the square root problem [5,17]). A more generic method was proposed by Even et al. [18] using oblivious transfers.

However, all these previous methods require that the communicating entities have the same or an equivalent computing power. This is unrealistic in practice (e.g. individuals versus large organizations).

In 1990, in the context of digital contract signing protocols, Ben Or et al [7] proposed to exchange privileges rather than bit information. An entity is said to be more privileged than other ones when it has a greater ability (than the other entities) to convince an external judge that the contract is signed by all the participating entities. The presented two-party protocol is such that the entities are privileged in turn. During the protocol, each entity sends to the other one a message saying that with a probability λ the contract will be valid (signed by both parties) at the (previously agreed) moment D. The probability λ has to increase during each round of the protocol. The protocol ends when $\lambda = 1$ or when the deadline D is reached. After the moment D, each party can present to the judge the last received message. The judge, once, chooses randomly a value between 0 and 1 and compares this value with the probability λ extracted from the message. If λ is greater or equal than the chosen value, the judge declares the two parties linked to the contract, otherwise he states that λ is too small. The decision is provided to both parties.

No other methods without TTP were proposed until the late 1990s. In 1998, Syverson proposed protocols [19] where low value information to be exchanged are ciphered and sent with a commitment of the key used to cipher them. These commitments will, in practice, be breakable if enough time is invested. This amount of time is known on the base of a known computing power. Such a commitment is called temporally secret bit commitment which could be implemented via a time-lock puzzle [20]. In the rationale exchange protocol, fairness is based on the rather unrealistic assumption that one of the parties is trusted. In the generic fair exchange protocol, two parties exchange the ciphered information and then send in turn a temporarily secret bit commitment easier to break than the previous one. Obviously, there will be a moment where one party can break the commitment in time. In that sense fairness is not maintained.

In another work, Han [21] proposed a protocol without TTP but where Alice is in possession of a system, the *pub*, publicly accessible and which automatically records all the operations (access, modification, ...) on the data that it contains (the operation recordings are neither “erasable” nor “modifiable”). The protocol envisages the sending by Alice of the ciphered message and then the disclosure of the deciphering key via the pub. Bob and the judge have to be sure about the validity of the information recorded in the pub. Hence, this is equivalent to having as many online TTPs as entities sending messages (e.g. Alice). It is also necessary to rely on Alice who manages the pub and who can possibly simulate it. Finally the suggested protocol requires a synchronization between the entities.

The first non-repudiation protocol without TTP was proposed in 1999 [8] and is described in the following section.

3.2 Markowitch and Roggeman protocol

The goal of this protocol is to avoid the intervention of a TTP at the price of accepting the probabilistic version of fairness. The protocol has to be parametrized on the basis of the most powerful entity’s computing power. This iterative protocol is such that, except at the last iteration, no entity is more privileged than another one during the protocol.

Being freed from a TTP during exchanges not only makes it possible to avoid a bottleneck in the communications but also permits to relax the need of trust in a TTP. The honesty of a TTP is difficult to evaluate. In the protocol here, the risk is known and can be parametrized.

Suppose Alice wants to send a message m and a non-repudiation of origin evidence of this message to Bob in exchange against Bob's non-repudiation of receipt evidence. The protocol is such that Alice will not find it beneficial to stop the protocol before its end. In the same way, if Bob stops the protocol before the last couple of sendings, he will not gain any profit. The only way for Bob to cheat, i.e. obtain Alice's message and the corresponding non-repudiation of origin evidence, without acknowledging receipt (by the means of sending a non-repudiation of receipt evidence), is to guess the number of iterations in the protocol. This number of iterations is selected randomly and secretly by Alice. At each iteration, the probability that Bob obtains the message and the non-repudiation of origin evidence without sending the non-repudiation of receipt will be smaller or equal to a quantity noted ε.

In the protocol the following evidences are generated.

- the evidence of origin for the cipher c: $\mathsf{EOO} = \mathrm{Sig}_A\left(f_{\mathsf{EOO}}, B, \ell, c\right)$
- the evidence of receipt for the cipher c: $\mathsf{EOR} = \mathrm{Sig}_B\left(f_{\mathsf{EOR}}, A, \ell, c\right)$
- the evidence of origin for the the value v_i : $\mathsf{EOO}_{k,i} = \mathrm{Sig}_A\left(f_{\mathsf{EOO}_{k,i}}, B, \ell, i, v_i\right)$
- the evidence of receipt for the the value v_i: $\mathsf{EOR}_{k,i} = \mathrm{Sig}_B\left(f_{\mathsf{EOR}_{k,i}}, A, \ell, i, v_i\right)$
- the non-repudiation of origin evidence: $\mathsf{NRO} = \{\mathsf{EOO}, \mathsf{EOO}_{k,n}\}$
- the non-repudiation of receipt evidence: $\mathsf{NRR} = \{\mathsf{EOR}, \mathsf{EOR}_{k,n}\}$

During the setup phase, Alice, who wants to send the message m to Bob, starts by choosing randomly, according to a geometrical distribution [1] (for example), a number n which will determine the number of iterations of the protocol. This value n is kept secret by Alice and will not have to be deduced by Bob during the protocol. She chooses also $n-1$ random independent and equidistributed values r_i and a key k (the random values and the key must have the same size).

Alice initiates the protocol by sending to Bob the cipher $c = \mathrm{C}_k(m)$, as well as the corresponding non-repudiation of origin evidence. Bob acknowledges the reception of this cipher. Alice, then, sends the first of the $n-1$ random value r_1 as well as the non-repudiation of origin evidence for r_1. Bob confirms the reception of this value, and the same process continues. At the $2n-1^{\text{th}}$ sending, Alice transmits the last random value r_{n-1} and the corresponding non-repudiation of origin evidence. Bob sends to Alice the non-repudiation of receipt evidence of r_{n-1}. Alice, then, transmits the deciphering key k (related to the cipher c) and Bob acknowledges the reception of this key which is indistinguishable from the already received random values. After a known delay or after having received a notification from Alice, Bob calculates $m = \mathrm{D}_k(c)$.

[1] A geometrical distribution is proposed here as its non-aging property avoids the leak of any information about the chosen number of iterations to Bob.

Before the last sending of Alice, Bob did not receive anything usable. Moreover, the only way he can detect whether he received the key k is by deciphering c using the value he received from Alice. But this computation will be supposed too long compared to the time before which Alice stops the protocol, not having received the expected non-repudiation of receipt evidence for the given value. We choose a cryptosystem, whose performances are appropriate with regard to the message size and the time Alice waits before stopping the protocol.

Protocol 1 Markowitch-Roggeman probabilistic protocol without TTP

1. A → B: $f_{\mathsf{EOO}}, B, \ell, c, \mathsf{EOO}$
2. B → A: $f_{\mathsf{EOR}}, A, \ell, \mathsf{EOR}$
3. A → B: $f_{\mathsf{EOO}_{k,1}}, B, \ell, 1, r_1, \mathsf{EOO}_{k,1}$
4. B → A: $f_{\mathsf{EOR}_{k,1}}, A, \ell, \mathsf{EOR}_{k,1}$

$\vdots$

$2n-1$. A → B: $f_{\mathsf{EOO}_{k,n-1}}, B, \ell, n-1, r_{n-1}, \mathsf{EOO}_{k,n-1}$

$2n$. B → A: $f_{\mathsf{EOR}_{k,n-1}}, A, \ell, \mathsf{EOR}_{k,n-1}$

$2n+1$. A → B: $f_{\mathsf{EOO}_{k,n}}, B, \ell, n, k, \mathsf{EOO}_{k,n}$

$2n+2$. B → A: $f_{\mathsf{EOR}_{k,n}}, A, \ell, \mathsf{EOR}_{k,n}$

At any moment, if Alice or Bob receives an incorrect message, they stop taking part in the protocol. Moreover, if Bob does not directly answer Alice's messages by sending the corresponding EOR_i, Alice will suppose that Bob attempts to cheat and consequently she stops the protocol (by not sending the next value).

It is necessary to determine deadlines after which Alice and Bob decide to not take part in the protocol anymore. A publicly known deadline can be considered when an entity awaits a sending. When the deadline expires, it is supposed that the entity who should carry out the sending is either trying to cheat, or that the network is overloaded (or that the protocol is ended). The protocol is then stopped. Such a mechanism makes it possible to use an unreliable network [2] .

Bob does not know the number of iterations n and cannot determine, when he receives a message from Alice, whether he receives the last message containing the deciphering key (no clue about n can be deduced from Alice's sending). We will note ε the probability that Bob guesses the value of n and does not send the evidence EOR_i precisely when $i = n$, knowing that n was selected secretly according to a geometrical distribution. If Bob does not send EOR_n, Alice will have sent all information necessary to Bob to obtain m and the

[2] Another solution consists in using an operational communication channel between Alice and Bob. If Bob does not receive a new message within the time ensured by the channel, he understands that the protocol is finished. On the other hand, if Alice does not receive Bob's acknowledgment within the time allowed by the channel, she notes that Bob tries to find m on the basis of the last value obtained; Alice then stops the protocol.

non-repudiation of origin evidence of m, whereas she does not obtain the non-repudiation of receipt evidence of m.

For the protocol to work correctly, it is necessary to use a cipher, where it is impossible to partially decrypt the ciphertext, in order to quickly decide whether the obtained value is the correct key or not. If, for instance, a block cipher is used, Alice must cipher her message, whatever the cryptosystem used, by means of a mode where all the blocks of the ciphertext must be deciphered to be able to obtain any blocks of the plain text. Such a mechanism is described in [8]. Moreover, it is possible that an all-or-nothing ciphering mode may not generate sufficient delay for deciphering short messages. A delaying mechanism, as [20] for example, could be used.

It is supposed that n has been randomly chosen following a geometrical distribution. If Alice or Bob stops the protocol before the $(2n+1)^{\text{th}}$ step, no one will obtain the necessary evidences ($\mathsf{EOO}_{k,n}$ and $\mathsf{EOR}_{k,n}$) composing the final non-repudiation evidences. After the last sending, Alice received all of Bob's acknowledgments and is able to compose the non-repudiation of receipt evidence for the message m. Bob is also able to build the non-repudiation of origin evidence for the message m and can decipher c to retrieve m. If Bob does not realize the $(2n+2)^{\text{th}}$ step, the protocol is no longer fair since Bob obtained the message m and the non-repudiation of origin evidence for this message, whereas Alice did not received the last Bob's acknowledgment and cannot compose the non-repudiation of receipt evidence for m. The probability that Bob, unaware of n, decides at the right moment to stop the protocol at the $(2n+1)^{\text{th}}$ step is ε, the success parameter of the geometrical distribution used by Alice to choose randomly n. The protocol is then ε-fair.

By the means of an operational channel between Alice and Bob, or by making use of deadlines, Alice and Bob will be able, for each sending of the protocol, to decide, within a finite amount of time, whether the protocol is ended. The protocol finishes after an expected number of $\frac{1}{\varepsilon}$ iterations.

3.3 Mitsianis protocol

More recently, in [22], a similar protocol to [8] has been proposed. Classically, Alice begins by sending to Bob the cipher c of the message m under a secretly chosen session key K_σ. If Bob acknowledges the reception of this ciphered message, Alice adds a padding ω to the session key K_σ to compose the key K'_σ. The size of the padding is chosen secretly by Alice. Alice ciphers this new padded key with a session key K_λ shared with Bob in order to obtain the ciphered key K_ς. This last ciphering is obtained using a mode similar to the one described in the section 3.2 in order to require Bob to obtain all the bits of the ciphered key K_ς to be able to decipher it.

Alice then chooses randomly and secretly, as in the previous protocol, a value

n and splits K_ς in n parts κ_i of different random lengths. Then n 2-moves iterations begin. At each iteration, Alice sends to Bob a κ_i (in the order she splits K_ς) and Bob has to acknowledge having received it. When Bob acknowledges the reception of the last parts, κ_n, of K_ς, Alice possesses the non-repudiation of receipt of the message m and Bob could retrieve the session key needed to recover the message m (he composes K_ς by concatenating the different received κ_i, deciphers the ciphered key using K_λ, obtains K'_σ, knowing the size of the session key K_σ he can extract the padding ω and retrieves K_σ; with K_σ he deciphers c and retrieves the message m).

Bob has to obtain all of the κ_i in order to retrieve K_σ, even if he knows the size of K_σ because of the *all-or-nothing* mode used to cipher K'_σ and his unawareness of the length of ω.

Exactly as in the previous protocol, if Bob does not send the last acknowledgment (of κ_n), he obtains the message m without providing the NRR at a probability of $\frac{1}{n}$. Also, Alice waits until a fixed deadline for each acknowledgment of Bob for κ_i before deciding that Bob is trying to cheat and stopping the protocol. A major difference between the protocol of section 3.2 and the here described protocol by Mitsanis is that in the latter the value n, determining the number of protocol's rounds, is static whereas n is dynamic in the first one. At the beginning of Mitsianis' protocol, Alice chooses n which will be fixed for the remaining of the protocol. In the Markowitch-Roggeman protocol, the number of rounds is dynamic, as the decision to stop or continue could be taken after each completed round, by launching a $\frac{1}{\varepsilon}-$faced die.

4 Non-repudiation protocols with inline TTP

We will start giving a short overview of protocols using either an inline or an online TTP. Then we present in details protocols following the more recent evolutions, making use of offline or transparent TTPs.

An inline TTP was first used in the context of certified email protocols [23]. In 1996, Coffey and Saidha [9] proposed a non-repudiation protocol which illustrates well the use of an inline TTP. Coffey and Saidha used the TTP as a *non-repudiation server*. The TTP collects the non-repudiation evidences and transmits them to Alice and Bob thereafter. The protocol makes use of time-stamps produced by a time-stamping authority as well as resilient communication channels between each of the entities and the TTPs. The time-stamping authority does not consider the content of the received messages (contrarily to what is sometimes awaited from a TTP). It just adds a time-stamp to a message signed by the entity with whom it communicates. To ensure the exchange of the non-repudiation evidences, the protocol uses partial evidences. Such evidences are known as partial because they are part of the final non-repudiation evidences. It should also be noticed that the protocol ensures confidentiality of the message transmitted by Alice to Bob.

Alice initiates the protocol by submitting, to the time-stamping authority, the signed partial non-repudiation of origin evidence to be dated and signed by this authority. If Alice's signature is valid, the time-stamping authority replies with the non-repudiation of origin evidence, ciphered for Alice. If this non-repudiation of origin evidence is valid (including the time-stamp), Alice requests the TTP to initialize a non-repudiable communication and sends the "final" non-repudiation of origin evidence as well as the partial non-repudiation of receipt evidence she computed to the TTP (the TTP could compute this partial non-repudiation of receipt evidence, but Alice carries it out in order to decrease the load of the TTP). If both the message sent by Alice and her signature are valid, the TTP sends the partial non-repudiation of receipt evidence to Bob. Bob signs this partial evidence and sends it to the time-stamping authority which, if the signature is correct, responds to Bob with the "final" non-repudiation of receipt evidence. Bob submits this evidence to the TTP. The TTP, after checkings, sends the non-repudiation of origin evidence to Bob and the non-repudiation of receipt evidence to Alice.

To distinguish different protocol sessions using a same TTP, the authors considered the use of a randomly chosen value associated to each session of the protocol. These random values are produced by the TTP and transmitted, by the TTP, to both Alice and Bob. Alice and Bob must present the random value they received during each communication with the TTP.

This protocol ensures strong fairness, since the TTP collects all information necessary before forwarding them to the concerned entities. It should be noted that Alice and Bob never communicate directly. The TTP is used as an intermediary in each transmission, and thus is inline.

The authors did not propose the use of maximum times after which a message is considered as lost. As presented by the authors, the protocol does not respect the timeliness property. However, if a deadline is fixed for each sending, the protocol is timely finite.

In case of disputes, an external judge can be invoked. If Alice affirms to have successfully sent a message to Bob or if Bob affirms to have received a message from Alice, the judge requests to the complaining entity or asks the TTP to provide the non-repudiation of receipt and/or non-repudiation of origin evidence. If these evidences cannot be provided, the judge rejects the complaint. Otherwise, the judge checks the signatures and the time-stamps on the evidences. If all these checks succeed, Alice's or Bob's claim is accepted.

Protocols with inline TTP present several disadvantages. First, they require the management of large databases by the TTP. It must preserve the messages it forwards, as well as the moments of each event. The management of such a database of centralized sensitive information represents a significant security risk. It is advisable to take particularly care of the protection of the information managed by such a TTP. Lastly, the bottleneck produced by the flow of information forwarded by the TTP is maximum. On the other hand, in opposition to online or offline TTPs, an inline TTP can include information about the time a message is sent or received into the evidences. Such evidences can for instance be used for settling disputes of late submission.

Consequently, an inline TTP requires, a particularly significant confidence, as well as the management of a considerable quantity of centralized resources.

5 Non-repudiation protocols with online TTP

The protocols based on an online TTP are such that the TTP does not act anymore as a delivery authority (as an intermediary for each transmission between the entities). However, an online TTP intervenes during each session of the protocol.

Some protocols from related frameworks, i.e. certified e-mail protocols and electronic payment protocols, using also an online TTP have been proposed in [24] and [25]. We consider here three non-repudiation protocols which are representative of the use that can be made of such a TTP.

To introduce non-repudiation protocols with online TTP, we will exceptionally make a return in time and evoke a protocol suggested by Rabin [26] in 1983. The author proposed an original method, called method by beacons, making it possible to carry out the exchange of an information against an acknowledgment. In this approach, the use of the TTP extremely differs from the traditionally developed methods.

5.1 Rabin's beacons protocol

This protocol is the first exchange protocol making use of a TTP. The idea is to have a TTP, broadcasting at regular and fixed intervals of time a signed message called the beacon. This beacon is composed of n public keys, a deciphering key corresponding to one of the public keys sent during the previous broadcasting and a value j indicating which previously broadcasted public key is associated with the presently broadcasted deciphering key.

In the first time, Alice sends to Bob a message ciphered under a session key k. During the protocol, Alice and Bob communicate directly and have to realize a complete round of the protocol between two broadcastings of the TTP. During a round, Bob randomly chooses an integer i between 1 and n. Bob sends this integer to Alice with a non-repudiation of receipt evidence for the message Alice would like to send to Bob. Then, Alice sends a signed message containing the session key ciphered with the i^{th} public key broadcasted by the TTP in the beacon received just before initiating the current round. If, when the TTP broadcasts the beacon, the value j chosen by the TTP is the same as the value i chosen by Bob, then Bob can retrieve the session key k and Alice's message. The non-repudiation of origin and receipt evidences are composed of the signed messages sent by Alice and Bob during this round associated with the beacon.

This protocol is probabilisticly fair as Alice can decide to stop the protocol when she receives the signed message from Bob and before sending her signed message. Fairness is only broken if Bob and the TTP choose the same value ($i = j$). This situation can happen with a probability equal to $\frac{1}{n}$.

However, this protocol respects a probabilistic version of the timeliness property. Although the probability that $i \neq j$ decreases when the number of rounds increases, this probability never reaches zero (due to the asymptotic behavior of the geometrical distribution). The expected number of rounds equals n.

In order to prevent a man in the middle attack for this protocol, we suggest that, when Alice and Bob send their signed message to each other, they add the recipient's identity in the messages. Otherwise, an opponent could intercept and block the messages that Bob sends to Alice. If this happens when j, chosen by the TTP, is equal to i, chosen by Bob, the opponent obtains a complete and valid non-repudiation of receipt evidence (from Bob) for a message he never sent.

This protocol, called "confidential disclosure protocol" by the author is rather connected with our definition of a non-repudiation protocol.

The protocol is such that the message remains confidential (and is recoverable only by Bob).

This probabilistic approach of fairness is comparable to the non-repudiation protocol without TTP presented at the section 3.2.

It should also be noted that the need for synchronization between the entities is a heavy constraint to realize. Therefore the deadline between two broadcastings must be parameterized so that the entity having the smallest computing power can provide the necessary message within the interval.

5.2 Zhang and Shi protocol

In 1996, Zhang and Shi proposed [10] a protocol where Alice transmits to Bob the message ciphered with a session key (and ciphered again by the mean of Bob's public key) and where the TTP publishes, at the right moment (decided by Bob and agreed by Alice), in a publicly accessible way, e.g. a public board, the session keys (provided to the TTP by Alice) needed to retrieve the message that Alice wants to transmit to Bob. Moreover, the TTP manages a database containing the keys used during a protocol run and records the time when these keys are added in that database. In this protocol, in case of dispute, the judge, resolving the given dispute, has to contact the TTP in order to decide which entity is honest. For this reason, the TTP cannot delete any information stored in the database. Such a database grows indefinitely.

The protocol ensures the confidentiality of the transmitted message (even once the session key is revealed, since the ciphered message is ciphered again with Bob's public key). The protocol also ensures the confidentiality of the message with respect to the TTP.

5.3 Zhou and Gollmann protocol

Zhou and Gollmann presented a non-repudiation protocol with online TTP [11]. The idea of this protocol is to reduce the work of the TTP to a minimum.

During the protocol, if an incorrect message arrives or if an awaited message does not arrive, the potential recipient stops the protocol.

Alice initiates the protocol by sending the cipher to Bob, using session key k, of the message she wants to transmit to Bob, a label identifying the protocol session, a time-out value before which the the session key must be submitted to the TTP and after which it can be consulted, as well as the signed non-repudiation of origin evidence for the ciphered message. If Bob accepts the consultation time-out proposed by Alice, he sends his signed non-repudiation of receipt evidence for the ciphered message. Alice then sends to the TTP a signed copy of the session key. The TTP accepts during a session of a protocol only one submission from an entity and checks whether Alice's signature is valid and whether the time-out is not exceeded. After the time-out, Bob can get, the session key and the non-repudiation of origin evidence for this session key provided by the TTP. This evidence is necessary in order to build a complete non-repudiation of origin evidence for the message that Alice sends to him. In a similar way, Alice consults the TTP to complete her non-repudiation of receipt evidence for the message.

Both Alice and Bob will fetch the session key and the corresponding evidence for this key at the TTP. This evidence serves to Bob as an evidence of origin and to Alice as a proof that the key is accessible to Bob. The entities consult, at the proper time, a read-only public directory managed by the TTP. If one of entity can not get the evidence at the TTP, while the other entity does, he will lose a possible future dispute on this subject.

The work of the TTP is thus reduced by rejecting the responsibility for obtaining the information managed by the TTP on the entities. The protocol requires for its good functioning a resilient channel of communication between the TTP and each entities.

If the communication channels between the TTP and respectively Alice and Bob are resilient, the protocol is strongly fair and the protocol respects the timeliness property.

In case of dispute, if Alice claims to have successfully sent a message to Bob, the judge asks her to provide this message and the non-repudiation of receipt evidence for this message. The non-repudiation evidence is composed of the non-repudiation of receipt evidence for the ciphered message, provided by Bob, and the non-repudiation of origin evidence of the session key, provided by the TTP. If all the information provided by Alice to the judge is correct, the judge declares that the assertion of Alice is correct.

If Bob claims to have received a message from Alice, the judge asks him to provide this message as well as the non-repudiation of origin evidence for this message. This evidence is composed of the non-repudiation of origin evidence for the ciphered message, provided by Alice, and the non-repudiation of origin evidence of the session key, provided by the TTP. If all the information provided by Bob to the judge is correct, the judge declares that the assertion of Bob is correct.

The protocol does not propose mechanisms ensuring the confidentiality. The session key and the ciphered message are accessible to any observer.

6 Non-repudiation protocols with offline TTP

In this section we present some non-repudiation protocols with offline TTPs. A TTP is said offline if it does not intervene in the protocol while no problem occurs. A problem could be an incorrect behavior of a dishonest entity or a network error. When such a problem occurs, Alice and/or Bob invoke the TTP to help them to finish the protocol run in a fair way. Such protocols suppose that most of the time no problem will occur. This is the reason why protocols with offline TTP are also called optimistic.

In the framework of exchange protocols, the first protocols, a certified e-mail protocol and fair exchange protocols, making use of an offline TTP were presented in [12], [13] and [27].

The first series of non-repudiaiton protocols detailed hereunder are variants of the protocols presented in [14] and [16]. The last part of the section is devoted to a non-repudiation protocol with transparent TTP. In this kind of protocol the TTP produces evidences which are indistinguishable from the evidences Alice and Bob should have exchanged in a faultless case.

6.1 A fair non-repudiation protocol

Here is the first fair non-repudiation protocol. The protocol is divided into two sub-protocols, a main and a recovery protocol. The TTP does not intervene in the main protocol. In case of problems, Bob can launch the recovery protocol. It is supposed that the communication channels between the TTP and both

Alice and Bob are resilient. The communication channels between Alice and Bob may be unreliable.

In the protocol the following evidences are generated.

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

The main protocol consists of three messages, that are detailed in protocol 2. All messages include one or several purpose flags and are linked by a label ℓ. The label in conjunction with the protocol entities uniquely identifies a protocol run. In the first message Alice sends a signed commitment, the cipher c to Bob. Alice also includes the decryption key k, ciphered with the TTP's public key. This allows the TTP in case of a recovery protocol to extract k and send it to Bob. In the second transmission, Bob sends the non-repudiation of receipt for the message to Alice. Although he hasn't received the message yet, he is sure that he is able to receive it later on. Alice finishes the protocol by sending k to Bob. If the last message does not arrive, Bob can launch the recovery protocol.

Protocol 2 A fair protocol - Main protocol

1. A $\rightarrow$ B: $f_{\mathsf{EOO}}, f_{\mathsf{Sub}}, B, TTP, \ell, c, E_{TTP}(k), \mathsf{EOO}, \mathsf{Sub}$
2. B $\rightarrow$ A: $f_{\mathsf{NRR}}, A, TTP, \ell, \mathsf{NRR}$
3. A $\rightarrow$ B: $f_{\mathsf{EOO}_k}, B, \ell, k, \mathsf{EOO}_k$
   if B times out then recovery

To execute the recovery protocol (protocol 3), Bob sends a recovery request to the TTP. This request proves to the TTP that Alice started the protocol with Bob. The TTP recovers the decryption key k and sends it back to Bob with an evidence, asserting that the key originated from Alice. The TTP also sends the non-repudiation of receipt evidence, which is included in Bob's recovery request, to Alice. This is necessary, as Bob could launch the recovery protocol after having received the first message of the main protocol, without having sent the second message.

Protocol 3 A fair protocol - Recovery protocol

1. B $\rightarrow$ TTP: $f_{\mathsf{Rec}}, f_{\mathsf{Sub}}, Y, l, h(c), E_{TTP}(k), \mathsf{Rec}, \mathsf{Sub}, \mathsf{NRR}, \mathsf{EOO}$
2. TTP $\rightarrow$ A: $f_{\mathsf{Con}_k}, A, B, l, k, \mathsf{NRR}$
3. TTP $\rightarrow$ B: $f_{\mathsf{Con}_k}, A, B, l, k, \mathsf{Con}_k$

After the first message has been sent, Bob does not possess a complete non-repudiation evidence. Neither does Alice. Note that Bob has the ability to launch the recovery protocol. However in that case both Alice and Bob receive the respected evidences. If the second message has been sent, Alice receives a complete non-repudiation of receipt evidence. Bob will either receive the third message from Alice or launch a recovery protocol. Also note that if Alice provides a wrong encrypted key in the first message, i.e. $E_{TTP}(k')$ instead of $E_{TTP}(k)$, the produced evidences will be invalid. Hence the protocol provides strong fairness. One easily sees that true fairness is not provided: in case of a TTP intervention, the second part of the non-repudiation of origin evidence differs from the one provided in a faultless execution.

Although the protocol provided here is fair, it does not provide timeliness. Consider the scenario where Bob stops the protocol after having received the first message. Alice cannot stop this protocol session, as Bob may launch a recovery protocol at some later moment. Alice needs to keep an open protocol session for a potentially infinite amount of time, as at the moment Alice decides to stop the protocol Bob could launch a recovery, resulting in an unfair situation. Note that the lack of timeliness could be considered as unfair in the common sense of the word fair, as Bob has reached an advantageous position. However the protocol remains fair, with respect to our definitions.

6.2 A fair non-repudiation protocol respecting timeliness

To remedy the shortcomings of the previous protocol, i.e. the lack of timeliness, a more complete protocol is detailed, respecting both fairness and timeliness. The protocol, in addition to a main and a recovery protocol, requires an abort protocol. In this protocol both Alice and Bob can launch a recovery protocol. The abort protocol can be executed by Alice and, as we will see underneath, implies that the timeliness property holds. While a recovery protocol forces the exchange to take place, the abort protocol informs the TTP of Alice's intention to stop the protocol. The recovery protocol and the abort protocol are mutually exclusive. The mutual exclusion is guaranteed by the TTP. The channel qualities are the same as in the previous protocol. The evidences generated in this protocol are the following.

- the evidence of origin for the cipher: $\mathsf{EOO} = S_A(f_{\mathsf{EOO}}, B, TTP, l, h(c))$
- the evidence of receipt for the cipher: $\mathsf{EOR} = S_B(f_{\mathsf{EOR}}, A, TTP, l, h(c))$
- the submission evidence for 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 evidence: $\mathsf{Con}_a = S_{TTP}(f_{\mathsf{Con}_a}, A, B, l)$

The precise description of the main protocol is given in protocol 4. The main protocol can be divided in two parts. The first part is the exchange of the cipher under key k of message m, and the evidence of origin for the cipher against an evidence of receipt for this cipher. The second part consists of the exchange of the key k and the corresponding evidence of origin against the evidence of receipt for the key k. If the second message does not arrive to Alice before a reasonable amount of time[3], she executes an abort protocol. If the third or fourth messages do not arrive, Bob and Alice, respectively, can launch a recovery protocol.

Protocol 4 A fair protocol respecting timeliness - Main protocol

1. A $\rightarrow$ B: $f_{\mathsf{EOO}}, f_{\mathsf{Sub}}, B, TTP, \ell, c, E_{TTP}(k), \mathsf{EOO}, \mathsf{Sub}$
2. B $\rightarrow$ A: $f_{\mathsf{EOR}}, A, TTP, \ell, \mathsf{EOR}$
   if A times out then abort
3. A $\rightarrow$ B: $f_{\mathsf{EOO}_k}, B, \ell, k, \mathsf{EOO}_k$
   if B times out then recovery
4. B $\rightarrow$ A: $f_{\mathsf{EOR}_k}, A, \ell, \mathsf{EOR}_k$
   if A times out then recovery

The abort protocol, described in protocol 5, can be launched at any time by Alice. If a valid abort request arrives, the TTP first verifies if the current protocol run has not yet been recovered or aborted. The protocol run is uniquely identified by the label ℓ and the identities (A, B). If neither a recovery protocol nor an abort protocol has been executed, the TTP informs both Alice and Bob that the protocol has been aborted. It is important to see that an abort evidence does not mean that the exchange did not take place. It is possible to complete a faultless main protocol and execute the abort protocol later on. An abort evidence only informs Alice and Bob that no recovery will be accepted any more by the TTP, regarding this protocol run.

Protocol 5 A fair protocol respecting timeliness - Abort protocol

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: $f_{\mathsf{Con}_a}, A, B, l, \mathsf{Con}_a$

A detailed description of the recovery protocol is given in protocol 6. The recovery protocol is intended to be executed by either Alice or Bob. Bob can launch the protocol, as soon as the first message of the main protocol arrives. Alice can launch the protocol once the second message of the main protocol arrived. The aim of the recovery protocol is to provide to Alice the possibly missing evidence of receipt for the cipher (EOR), as well as a substitution (Con_k) for the evidence of receipt of the key k, and to Bob a substitution (Con_k) of the missing evidence of origin for the key k, as well as the key itself.

[3] Alice chooses herself how long she decides to wait for a given message before reacting.

Protocol 6 A fair protocol respecting timeliness - Recovery 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: $f_{\mathsf{Con}_k}, A, B, l, k, \mathsf{Con}_k$

Recovery protocol. After the first message has been sent, Bob can either stop the protocol, launch a recovery protocol or reply and continue the main protocol. If Bob stops the protocol no complete evidence has yet been obtained and no party will obtain a correct evidence anymore. If Bob launches a recovery protocol, both Alice and Bob will receive all expected evidences, and hence the protocol remains fair. Note that Alice is not going to stop the protocol just after having sent the first message. Such a behavior would harm herself, as Bob could launch a recovery protocol, and get the expected evidences. At any moment, Alice can launch an abort protocol. However she will not continue the protocol after having done so, even if Bob's reply arrives afterwords, as Bob could decide not to send the fourth message of the main protocol. In that case Alice would not have the possibility any more to execute the recovery protocol, as it is mutually exclusive with the abort protocol. The protocol would end up in an unfair situation. Hence, Alice either launches the abort protocol and stops afterwords, or continues after having received message 2 of the main protocol. When the second message arrives, both participants have the ability to recover the protocol. A recovery implies that both participants receive complete evidences. Hence, each party can force the successful exchange. Only, if Alice launches an abort protocol the recovery is not accepted anymore. However, doing this would harm Alice, as Bob has the advantageous position in the main protocol, i.e. Bob obtains his complete non-repudiation evidence before Alice. Also note, that as in the protocol described above, providing a wrong key, ciphered for the TTP, in the first message, results into invalid evidences.

Timeliness is provided by the fact that at each moment in the protocol, both Alice and Bob can take an action to force a fair termination. While in the previous protocol, Alice could not react if Bob stops the protocol after the first message, she can now abort the protocol. If the abort is accepted, Alice can stop the protocol, knowing that Bob is unable to launch a successful recovery protocol any more. On the other hand, the only reason an abort request could be refused is the previous execution of a recovery protocol. Hence, the evidences will arrive after a finite amount of time to both Alice and Bob, due to the resilience of the communication channels. Once the second message of the main protocol arrived, both entities are able to recover the protocol. We conclude that the protocol provides timeliness.

Although confidentiality is not required in bare non-repudiation protocols, many applications require the secrecy of the sent message. Confidentiality is however rather easy to provide. One could for instance cipher k or c with

Bob's public key, each time they are sent over the network. Another solution is to use ad-hoc mechanisms such as VPN or SSL to assure the confidentiality of the message.

6.3 Non-repudiation protocols with transparent TTP

In the previous non-repudiation protocols with offline TTP, when the TTP intervenes, in case of problems during the communication between Alice and Bob, it digitally signs some pieces of information which will be used as non-repudiation evidences. These evidences have the same effect to an adjudicator as those produced by Alice and Bob in a faultless case.

The aim of the following protocol [28] 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 commerce, in order to avoid bad publicity.

The use of an *invisible TTP* was first proposed by Micali [13] in the framework of certified e-mails. Asokan et al. [29] and Bao et al. [30], proposed fair exchange protocols allowing to recover, in case of problem, the original client's signature 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 been broken by Boyd and Foo [31]. In the same paper, Boyd and Foo [31] 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 [32]. 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 [33]. The protocol is based on a specific signature scheme (inspired by the Girault-Poupard-Stern signature scheme [34]). 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. 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 (the description of the message will only be known 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 hereafter is based on the Markowitch-Saaednia method [33].

In this protocol, the TTP produces, when a fault occurs during the main protocol execution, exactly the same evidences as 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.

The protocol uses a signature scheme based on the GPS signature scheme [34,35]. 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 also chooses 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 $y_u = \alpha^{x_u} \bmod n$. As usual in public key cryptography, a certificate for the public key has to be obtained and distributed.

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 [33,34].

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: $\mathsf{Con}_e = S_{TTP}(f_{\mathsf{Con}_e}, A, B, l)$

Alice starts the main 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.

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

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).

Protocol 7 A protocol with transparent TTP - Main protocol

1. A $\to$ B: $f_{\mathsf{EOO}}, f_{\mathsf{Sub}}, B, TTP, l, h(k), c, E_{TTP}(k)$, EOO, Sub
2. B $\to$ A: $f_{\mathsf{EOR}}, A, TTP, l$, EOR
   if A times out then abort
3. A $\to$ B: $f_{\mathsf{NRO}}, B, l, k$, NRO
   if B times out then recovery$[X := B, Y := A]$
4. B $\to$ A: f_{NRR}, A, l, NRR
   if A times out then recovery$[X := A, Y := B]$

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 to try 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.

Protocol 8 A protocol with transparent TTP - Recovery protocol

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

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 has not yet been recovered or aborted, the TTP sends to both Alice and Bob a signed abort confirmation.

Protocol 9 A protocol with transparent TTP - Abort protocol

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

The TTP runs the 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) [4]. The goal of this protocol is to warn Bob that Alice tried to cheat (and to inform Alice that this attempt has been detected) [5].

Protocol 10 A protocol with transparent TTP - Error protocol

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

If Bob stops the protocol after having received the first message, Alice can run the abort protocol to prevent Bob from initiating 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 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

[4] 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.

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

will get his non-repudiation evidence, while Alice does not get her evidence. Such a behavior would contradict the assumption 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.

7 Key revocation and non-repudiation evidences

The security of a non-repudiation protocol also depends on some ad-hoc problems. One of the most important issues is good management of the non-repudiation evidences and hence of the used digital signatures and the corresponding keys. It can happen that a secret signature generation key is compromised. It is then necessary to revoke the certificate of the corresponding public verification key ([36,37]). In the context of non-repudiation protocols, it is necessary to be able to identify whether a signature (present in a non-repudiation evidence) was generated before or after the revocation.

A traditional solution [38,39] consists in using a TTP acting as a time-stamping authority (as Coffey and Saidha did). The time-stamp present in an evidence is then compared with the date of revocation.

You et al. proposed [40] a mechanism where such an authority is not necessary any more. The approach consists in binding all the evidences, generated during a non-repudiation protocol, between them and validating all of them only at the end of the protocol. This idea could be applied to the non-repudiation protocol with online TTP of Zhou and Gollmann, where the TTP ends up the protocol by providing the session key. This session key has to be transmitted during the protocol to the TTP by Alice in a confidential (e.g. ciphered) way. Otherwise, Bob could intercept the key and then revoke his signature key certificate. The non-repudiation evidences of the Zhou and Gollmann protocol are then modified so that the non-repudiation of origin evidence of the ciphered message is included in the non-repudiation of receipt evidence of the ciphered message. This last evidence is incorporated in the non-repudiation of origin evidence of the session key produced by Alice and sent by her to the TTP.

The TTP will include the non-repudiation of receipt evidence of the ciphered message and the moment from which all the evidence become significant in the non-repudiation of origin evidence for the session key that the TTP produces. Bob will not be able to revoke his signature key until the moment when the TTP reveals this non-repudiation of origin evidence for the session key. The TTP will check the validity of the certificate of the public key of Bob before producing his evidence on the session key.

The previous approach works well in protocols where a TTP has to intervene in each protocol run. However, in probabilistic protocols, where no TTP is involved at all, or in optimistic protocols, which aim that the TTP does not intervene in most cases, the approach described above is not applicable. In [41], Zhou et al. present a method not requiring a TTP. The idea is based on using two kinds of signature keys: long term revocable signature keys and short term irrevocable signature keys. The first kind of keys are classical signature keys issued by a certification authority. The second kind are signature keys issued by the signatory itself. The certificate for these keys contains a time stamp and is signed using the long term key.

Each entity owns a classical, long term signature key. Before starting a non-repudiation protocol, the entity generates a short term key, signs a certificate for this key, using its long term signature key. The new certificate also contains the life time of this key. Then the entity adds a time stamp on the certificate by contacting a time stamp authority. The time stamp authority verifies the validity of the long term key and checks that the life time of the new key does not exceed the life time of the long term key.

During a non-repudiation protocol, the entities sign their evidences using their short term irrevocable evidences. The recipient of an evidence verifies the certificate and checks the lifetime of the key. As these keys cannot be revoked, the recipient is sure that the evidence is valid. Although a short term key may be compromised, it cannot be revoked. However, as the lifetime of these keys is very short, this risk is acceptable.

Another approach which does not need a trusted third party to be involved in order to maintain the validity of digital signatures, and hence applicable in an optimistic environment, is proposed in [42]. This method is useful in a context where an entity transmits, to a same recipient, several digital signatures during a communication (as it is the case in a non-repudiation protocol). When an entity issues a digital signature, he also signs the hash of the previous signature he produced. If the entity wants to revoke his signature key, he asks the recipients to countersign his last digital signature. Then the entity can deny other signed messages generated with the revoked key but not those produced before the revocation and being part of the signatures link leading to the countersignature.

8 Comparisons

In this section we are reviewing most of the published non-repudiation protocols. We give a comparison in the following table, where we summarize important information such as the degree of fairness that is reached, whether timeliness is respected or not, which kind of TTP is involved in the protocol and the channel requirements. It is rather difficult to make a direct performance comparison, as the performance depends heavily on facts such as the network load that could create a bottleneck at the TTP, when it is inline or online, the honesty of the entities in optimistic protocols, ... However we remark that some protocols, above all the older ones, do not respect timeliness and are less suitable in practice. Moreover the channel requirements may be of crucial importance. For instance, operational channels are rather unrealistic in heterogeneous networks.

Protocol	Fairness	Timeliness	TTP involvement	Channel requirements
Coffey-Saidha [9]	s	N	inline	r
Rabin [26]	p	p	online	o [6]
Zhang-Shi [10]	s	N	online	r
Zhou-Gollmann [11]	s	Y	online	r
Zhou-Gollmann [15]	s	Y	offline	o
Zhou et al.,Kremer-Markowitch [16,43]	s	Y	offline	r
Markowitch-Kremer [28]	t	Y	offline transparent	r
Markowitch-Roggeman [8]	p	p	none	u
Mitsianis [22]	p	p	none	u

s=strong, t=true, p=probabilistic, u=unreliable, r=resilient, o=operational

For inline protocols it is impossible to have a neutral TTP as the TTP itself transmits the message. In the other cited protocols with TTP, online and offline TTPs are also neutral.

[6] The TTP uses broadcasting to transmit the beacons.

9 Conclusion

The aim of this paper is to give a state-of-the-art of non-repudiation mechanisms. Throughout the paper we surveyed most of the existing non-repudiation protocols. In the beginning of the paper we clearly defined the properties a non-repudiation protocol is required to respect. Then we browsed through the different approaches without and with TTP and showed the evolution of the involvement of the TTP from protocols, using an inline TTP towards protocols, where the TTP is offline and transparent. We also discussed some ad-hoc problems related to a correct management of the evidences and the problems implied by signature key revocation. Finally we briefly described the existing methods for formally verifying non-repudiation protocols.

There have been several previous surveys covering the topic of non-repudiation protocols. In 1997, Zhou and Gollmann [44] wrote a first survey on the topic, where they defined the different services, and the related evidences, that non-repudiation mechanisms have to provide. However, probabilistic non-repudiation and recent techniques using offline TTPs are not covered in this early paper. More recently, Louridas [45] gave some informal guidelines for designing non-repudiation protocols. He emphasizes on several practical problems where special care is required. Only a first attempt to protocols using offline TTPs is given in this paper. In [46], Zhou gave a rather complete overview on the topic. However, due to the very fast evolution of this topic, the latest techniques are not covered in his book. Hence, this survey paper is the only of its kind covering the very recent techniques, such as transparent TTPs and the latest attempts of using formal method verification, giving a complete picture of the subject.

References

[1] W.Diffie, M.E.Hellman, New directions in cryptography, IEEE Transactions on Information Theory 22 (6) (1976) 644–654.

[2] ISO/IEC 13888-1, Information technology - Security techniques - Non-repudiation - Part 1: General (1997).

[3] ISO/IEC 13888-2, Information technology - Security techniques - Non-repudiation - Part 2: Mechanisms using symmetric techniques (1998).

[4] ISO/IEC 13888-3, Information technology - Security techniques - Non-repudiation - Part 3: Mechanisms using asymmetric techniques (1997).

[5] T. Tedrick, How to exchange half a bit, in: D. Chaum (Ed.), Advances in Cryptology: Proceedings of Crypto 83, Plenum Press, New York and London, 1984, 1983, pp. 147–151.

[6] T. Tedrick, Fair exchange of secrets, in: G. R. Blakley, D. C. Chaum (Eds.), Advances in Cryptology: Proceedings of Crypto 84, Vol. 196 of Lecture Notes in Computer Science, Springer-Verlag, 1985, pp. 434–438.

[7] M. Ben-Or, O. Goldreich, S. Micali, R. Rivest, A fair protocol for signing contracts, IEEE Transaction on Information Theory 36 (1) (1990) 40–46.

[8] O. Markowitch, Y. Roggeman, Probabilistic non-repudiation without trusted third party, in: Second Conference on Security in Communication Networks'99, Amalfi, Italy, 1999.

[9] T. Coffey, P. Saidha, Non-repudiation with mandatory proof of receipt, ACMCCR: Computer Communication Review 26.

[10] N. Zhang, Q. Shi, Achieving non-repudiation of receipt, The Computer Journal 39 (10) (1996) 844–853.

[11] J. Zhou, D. Gollmann, A fair non-repudiation protocol, in: IEEE Symposium on Security and Privacy, Research in Security and Privacy, IEEE Computer Society,Technical Committee on Security and Privacy, IEEE Computer Security Press, Oakland, CA, 1996, pp. 55–61.

[12] N. Asokan, M. Schunter, M. Waidner, Optimistic protocols for fair exchange, in: T. Matsumoto (Ed.), 4th ACM Conference on Computer and Communications Security, ACM Press, Zurich, Switzerland, 1997, pp. 6, 8–17.

[13] S. Micali, Certified E-mail with invisible post offices, Available from author; an invited presentation at the RSA '97 conference (1997).

[14] S. Kremer, O. Markowitch, Optimistic non-repudiable information exchange, in: J. Biemond (Ed.), 21st Symp. on Information Theory in the Benelux, Werkgemeenschap Informatie- en Communicatietheorie, Enschede (NL), Wassenaar (NL), 2000, pp. 139–146.

[15] J. Zhou, D. Gollmann, An efficient non-repudiation protocol, in: Proceedings of The 10th Computer Security Foundations Workshop, IEEE Computer Society Press, 1997, pp. 126–132.

[16] J. Zhou, R. Deng, F. Bao, Evolution of fair non-repudiation with TTP, in: ACISP: Information Security and Privacy: Australasian Conference, Vol. 1587 of Lecture Notes in Computer Science, Springer-Verlag, 1999, pp. 258–269.

[17] M. Blum, How to exchange (secret) keys, ACM Transactions on Computer Systems 1 (2) (1983) 175–193, previously published in ACM STOC 83 proceedings, pages 440–447.

[18] S. Even, O. Goldreich, A. Lempel, A randomized protocol for signing contracts, Communications of the ACM 28 (6) (1985) 637–647.

[19] P. Syverson, Weakly secret bit commitment: Applications to lotteries and fair exchange, in: Proceedings of the 1998 IEEE Computer Security Foundations Workshop (CSFW11), 1998.

[20] R. L. Rivest, A. Shamir, D. A. Wagner, Time-lock puzzles and timed-release crypto, Technical Report MIT/LCS/TR-684, Massachusetts Institute of Technology (Feb. 1996).

[21] Y. Han, Investigation of non-repudiation protocols, in: ACISP: Information Security and Privacy: Australasian Conference, Vol. 1172 of Lecture Notes in Computer Science, Springer-Verlag, 1996, pp. 38–47.

[22] J. Mitsianis, A new approach to enforcing non-repudiation of receipt, manuscript (2001).

[23] A. Bahreman, J. D. Tygar, Certified electronic mail, in: Symposium on Network and Distributed Systems Security, Internet Society, 1994, pp. 3–19.

[24] R. H. Deng, L. Gong, A. A. Lazar, W. Wang, Practical protocols for certified electronic mail, Journal of Network and System Management 4 (3) (1996) 279–297.

[25] B. Cox, J. D. Tygar, M. Sirbu, NetBill security and transaction protocol, in: USENIX Association (Ed.), Proceedings of the first USENIX Workshop of Electronic Commerce, USENIX, 1995, pp. 77–88.

[26] M. O. Rabin, Transaction protection by beacons, Journal of Computer and System Sciences 27 (2) (1983) 256–267.

[27] N. Asokan, V. Shoup, M. Waidner, Asynchronous protocols for optimistic fair exchange, in: Proceedings of the IEEE Symposium on Research in Security and Privacy, 1998, pp. 86–99.

[28] O. Markowitch, S. Kremer, An optimistic non-repudiation protocol with transparent trusted third party, in: Information Security Conference 2001, Lecture Notes in Computer Science, Springer-Verlag, 2001.

[29] N. Asokan, V. Shoup, M. Waidner, Optimistic fair exchange of digital signatures, in: Advances in Cryptology: Proceedings of Eurocrypt'98, Vol. 1403 of Lecture Notes in Computer Science, Springer-Verlag, 1998, pp. 591–606.

[30] F. Bao, R. H. Deng, W. Mao, Efficient and practical fair exchange protocols with off-line TTP, in: IEEE Symposium on Security and Privacy, 1998.

[31] C. Boyd, E. Foo, Off-line fair payment protocols using convertible signatures, Lecture Notes in Computer Science 1514 (1998) 271–285.

[32] D. Chaum, Designated confirmer signatures, in: A. D. Santis (Ed.), Advances in Cryptology: Proceedings of Eurocrypt'94, Vol. 950 of Lecture Notes in Computer Science, Springer-Verlag, 1995, 1994, pp. 86–91.

[33] O. Markowitch, S. Saeednia, Optimistic fair-exchange with transparent signature recovery, in: 5th International Conference, Financial Cryptography 2001, Lecture Notes in Computer Science, Springer-Verlag, 2001.

[34] G. Poupard, J. Stern, Security analysis of a practical "on the fly" authentication and signature generation, in: Advances in Cryptology: Proceedings of Eurocrypt'98, Vol. 1403 of Lecture Notes in Computer Science, Springer-Verlag, 1998, pp. 422–436.

[35] M. Girault, Self-certified public keys, in: Advances in Cryptology: Proceedings of EuroCrypt'91, Vol. 547 of Lecture Notes in Computer Science, Springer-Verlag, 1991, pp. 490–497.

[36] M. Naor, K. Nissim, Certificate revocation and certificate update, in: Proceedings of the 7th USENIX Security Symposium (SECURITY-98), Usenix Association, 1998, pp. 217–228.

[37] R. Rueppel, Revocation and revocation certificates, in: Proceedings of the Trusted Third Party Workshop, Barcelona, Spain, 1995, pp. 1–8.

[38] S. G. Akl, Digital Signatures: A Tutorial Survey, IEEE Computer 16 (2) (1983) 15–24.

[39] K. S. Booth, Authentication of signatures using public key encryption, Communications of the ACM 24 (11) (1981) 772–774.

[40] C. You, J. Zhou, K. Lam, On the efficient implementation of fair non-repudiation, Computer Communication Review 28 (5) (1998) 50–60.

[41] J. Zhou, K. Lam, Securing digital signatures for non-repudiation, Computer Communications 22 (8) (1999) 710–716.

[42] J. Zhou, R. Deng, On the validity of digital signatures, Computer Communication Review 30 (2) (2000) 29–34.

[43] S. Kremer, O. Markowitch, Optimistic non-repudiable information exchange, in: J. Biemond (Ed.), 21st Symp. on Information Theory in the Benelux, Werkgemeenschap Informatie- en Communicatietheorie, Enschede (NL), Wassenaar (NL), 2000, pp. 139–146.

[44] J. Zhou, D. Gollmann, Evidence and non-repudiation, Journal of Network and Computer Applications 20 (3) (1997) 267–281.

[45] P. Louridas, Some guidelines for non-repudiation protocols, Computer Communication Review 30 (5).

[46] J. Zhou, Non-repudiation in electronic commerce, Computer Security Series, Artech House, 2001.