J. A. Onieva, J. Zhou, and J. Lopez, "Non-repudiation Protocols for Multiple Entities", *Computer Communications*, vol. 27, pp. 1608-1616, 2004.
NICS Lab. Publications: https://www.nics.uma.es/publications

Non-repudiation protocols for multiple entities [1]

Jose A. Onieva [a,2] Jianying Zhou [b,3] Javier Lopez [a]

[a] *Computer Science Department, E.T.S. Ingenieria Informatica*
University of Malaga, 29071 - Malaga, Spain

[b] *Institute for Infocomm Research*
21 Heng Mui Keng Terrace, Singapore 119613

Abstract

Non-repudiation is a security service that provides cryptographic evidence to support the settlement of disputes. In this paper, we introduce the state-of-the-art of non-repudiation protocols for multiple entities. We extend an existing *multi-party non-repudiation* (MPNR) protocol to allow an originator to send different messages to many recipients in a single transaction. We further propose an optimistic multi-party non-repudiation protocol for exchange of different messages. The performance of our protocols with enhanced functionalities is still promising in comparison with existing multi-party non-repudiation protocols.

Key words: multi-party non-repudiation, fair exchange, security protocol, electronic commerce

Email addresses: onieva@lcc.uma.es (Jose A. Onieva), jyzhou@i2r.a-star.edu.sg (Jianying Zhou), jlm@lcc.uma.es (Javier Lopez).

[1] Part of the work has been published in [14].

[2] The first author's work was done during his attachment to Institute for Infocomm Research under its sponsorship.

[3] Contact author.

1 Introduction

Electronic commerce could be defined in several ways and it is difficult to find out which of them is the most appropriate. New concepts like e-payment, e-banking, e-lottery, e-voting, etc., belong to this area. Nevertheless, we can define electronic commerce as "commercial transactions carried out with the assistance of telecommunications" [17]. What seems to be clear is that e-commerce helps businesses to expand their strategy and market, and for that, most of them are being shifted to the Internet or taking advantages of other digital sources. That means traditional paper-based transactions must be transformed into digital procedures.

But during the last years the impressive growth of the Internet and more generally of open networks has created several security-related problems. Repudiation is one of them. *Non-repudiation* must ensure that no party involved in a protocol can deny having participated in a part or the whole of the protocol. Therefore, a non-repudiation protocol must generate cryptographic evidence to support dispute resolution. In a typical non-repudiation protocol, a *trusted third party* (TTP) helps entities to accomplish their goals. Non-repudiation is especially important in electronic commerce to protect customers and merchants. It must not be possible for the merchant to claim that he sent the electronic goods when he did not. In the same way, it must not be possible for the customer to deny having received the goods. Other users such as online tax payers or administration users for a secure paperless office would also need a non-repudiation service.

In order to achieve a non-repudiation service, some common phases have to appear in the protocol:

Service request - One or more parties involved must somehow agree, prior to its origination and delivery, to utilize non-repudiation services and to generate the necessary evidence for a non-repudiation service.

Evidence generation - Depending on the non-repudiation service being provided and the non-repudiation protocol being used, evidence could be generated by the originator, the recipient, or the trusted third party. The elements of non-repudiation evidence and the algorithms used for evidence generation are determined by the non-repudiation policy in effect and service request phase. Namely, evidence can be generated using secure envelopes or digital signatures. The latter is more widely employed. A digital signature basically links a message with its originator, and also maintains the integrity of the message.

Evidence transfer - The evidence generator must transfer the evidence to the party who may ultimately need to use it. The principal participants may utilize trusted third parties to receive evidence.

Evidence verification and storage - Newly received evidence should be verified to gain confidence that the supplied evidence will indeed be adequate in the event of a dispute arising. The verification procedure is closely related to the mechanism of evidence generation. As the loss of evidence could result in the loss of future possible dispute resolution, the verified evidence needs to be stored safely. The duration of storage will be defined in the non-repudiation policy in effect.

Dispute resolution - This phase will not be activated unless disputes related to a transaction arise. When a dispute arises, an adjudicator will be invoked to settle the dispute according to the non-repudiation evidence provided by the disputing parties. The evidence required for dispute resolution and the means which the adjudicator will use to resolve a dispute are determined by the non-repudiation policy in effect.

A non-repudiation protocol generates at least the following important evidence for the participating entities:

Evidence of origin. This evidence is generated by the originator (perhaps with the assistance of a TTP) for a particular message and intended to the recipient, such that the originator cannot deny having sent that message.

Evidence of receipt. This evidence is generated by the recipient (perhaps with the assistance of a TTP) for a received message and intended to the originator, such that the recipient cannot deny having received that particular message from the originator.

In a typical two-party non-repudiation service, we identify several requirements, some of which could be optional, depending on the application the non-repudiation service is running over:

Fairness. A non-repudiation protocol provides fairness if neither party can gain an advantage by quitting prematurely or otherwise misbehaving during a protocol. At the end of the protocol either the sender gets evidence of receipt and the recipient receives a message as well as evidence of origin for that message or none of them gets any valuable item.

Timeliness. A non-repudiation protocol provides timeliness if any of the participating entities has the ability to reach the end of the protocol in a finite amount of time without loss of fairness.

Confidentiality. A non-repudiation protocol provides confidentiality if none but the intended parties can get access to the (plaintext) message sent during the non-repudiation protocol.

Several solutions to fair non-repudiation have been developed [12]. Some of them use a TTP which plays the role of a delivery agent between the participating entities. The major disadvantage of this approach is the communication bottleneck created at the TTP. Nevertheless, Zhou and Gollmann presented

a protocol [20] where the TTP intervenes during each execution as a "low weight notary" rather than as a delivery agent. Other solutions use an off-line TTP, assuming that participating entities have no malicious intentions and the TTP does not need to be involved unless there is an error in the protocol execution. This is called an *optimistic approach*. There are also solutions that completely eliminate the TTP's involvement. However, they need a strong requirement: all involved parties must have the same computational power in *gradual exchange* protocols, or fairness depends on the number of protocol rounds in *probabilistic* protocols.

Previous work on non-repudiation in the literature was mostly focused on the two-party scenario. There has been some work on the multi-party scenario in the related topics like *fair exchange*, where multiple entities exchange items among themselves without loss of fairness [5,8–10]. However, the research towards a generalization of non-repudiation with multi-party involvement has not been sufficiently undertaken. Markowitch and Kremer extended the two-party non-repudiation scenario to allow one originator to send the same message to multiple recipients in a single protocol run [11,13] [4]. In this paper, we further generalize their *multi-party non-repudiation* (MPNR) protocol such that the originator is able to send different messages to multiple recipients, and more importantly, in an optimistic approach. We also analyze the performance of our protocols, and discuss the cryptographic primitives for group encryption as well as possible optimization based on VPN.

The following basic notation is used throughout the paper.

- x, y : concatenation of messages x and y
- u_P : the public key of user P
- $S_P(X)$: digital signature of user P over message X
- $E_K(X)$: encryption of message X with key K
- $h(X)$: one-way hash function
- f_p : a flag indicating the *purpose* of a message
- $O \rightarrow P$: entity O sends a message to entity P
- $O \leftrightarrow P$: entity O fetches a message from entity P
- $O \Rightarrow R$: entity O broadcasts a message to a set of entities R

[4] The use of a semi-trusted intermediary for multi-party non-repudiation was discussed in [15], which helps final entities to collect, verify, and store evidence in electronic transactions.

2 A Fair MPNR Protocol for Exchange of Different Messages

Typically, non-repudiation protocols, as well as fair exchange protocols, have been studied as a two-party problem, in which Alice and Bob play the roles of originator and recipient, respectively. Although research has been conducted toward a multi-party scenario in fair exchange protocols [2,5,8,9] and contract signing protocols [1,3], non-repudiation protocols have not received such attention. Furthermore, some properties studied in multi-party fair exchange protocols such as exclusion freeness [5] are not applicable for non-repudiation protocols.

In a typical fair exchange protocol, each entity possesses an item, usually known a priori, and is willing to exchange for an item belonging to another entity. Hence, matrix or ring [10] topologies appear to provide solutions to these scenarios. In a multi-party fair exchange protocol, one can imagine sending an item to one entity and receiving an item from a different one. However, it does not make sense in a non-repudiation protocol that one entity receives some data and a distinct entity sends the corresponding receipt. Thus, in multi-party non-repudiation protocols, one of the entities plays the role of originator, and the others behave as recipients. Although other topologies could exist, this seems to be the most intuitive. As an example we could think in a practical certified e-mail application in which the sender wants to send a message (or different messages) to multiple recipients in only one transaction. However, other possibilities for generalization (e.g., many-to-one and many-to-many) may also exist.

An extension by Kremer et al. [11] of a low weight notary protocol for two entities [20] is the first non-repudiation protocol appeared in the literature dealing with multiple entities. This protocol supports a one-to-many topology in which the originator aims to send the same message to multiple recipients. This protocol broadcasts a message among several entities and provides evidence only to those entities who behave honestly during the protocol run, using the same key k for encryption. Nevertheless, it is not possible to send different messages to different recipients. In that way no personal and confidential messages can be sent to these parties without loss of privacy.

For such a reason we propose an extension in which customized messages can be sent to different parties in a confidential way. Sending (same or different) messages to several recipients could mean a single transaction in a specific application. Therefore, it would be better to store the same key and evidence in the TTP record for every protocol run. In those types of applications, the storage and computation requirements of the TTP are reduced and it will be easy to distinguish between different transactions, regardless of how many entities are involved.

Some useful notation in the protocol description is as follows.

- O : an originator
- R : set of intended recipients
- R' : subset of R that replied to O with the evidence of receipt
- M_i : message being sent from O to a recipient $R_i \in R$
- n_i : random value generated by O
- $v_i = E_{u_{R_i}}(n_i)$: encryption of n_i with R_i's public key
- k : key being selected by O
- $k_i = k \; xor \; n_i$: a key for R_i
- $c_i = E_{k_i}(M_i)$: encrypted message for R_i with key k_i
- $l_i = h(O, R_i, TTP, h(c_i), h(k))$: label [5] of message M_i
- L' : labels of all the messages being sent to R'
- t : a timeout chosen by O, before which the TTP has to publish some information
- $E_{R'}(k)$: a group encryption scheme that encrypts k for the group R' (see section 4.1 for further details)
- $EOO_i = S_O(f_{eoo}, R_i, TTP, l_i, t, v_i, u_{R_i}, c_i)$: evidence of origin for R_i
- $EOR_i = S_{R_i}(f_{eor}, O, TTP, l_i, t, v_i, u_{R_i}, c_i)$: evidence of receipt from R_i
- $Sub_k = S_O(f_{sub}, R', L', t, E_{R'}(k))$: evidence of submission of the key to the TTP
- $Con_k = S_{TTP}(f_{con}, O, R', L', t, E_{R'}(k))$: evidence of confirmation of the key by the TTP

In this extension, the use of the same key for all users creates a new problem that did not appear in Kremer's protocol. As messages are different, when the same key k is used for encryption, and after the key k is published, any recipient will be able to read the messages destined to the other recipients (by eavesdropping the messages that are transmitted between O and R). More importantly, R_i could get c_i in the initial steps of the protocol and quit. Then, colluding with any other party and getting the unique key k, it could decrypt c_i without providing any evidence of receipt. These problems are solved in our extended multi-party non-repudiation protocol, introducing some extra cost for the extended functionality over [11].

2.1 The Protocol

Here, we describe the protocol (see figure 1, where a dotted line indicates a *fetch* operation).

[5] There might be a potential attack [16] when the label l is constructed as $h(m, k)$ in the early literature, so we make the label unique in each run and verifiable by any party.

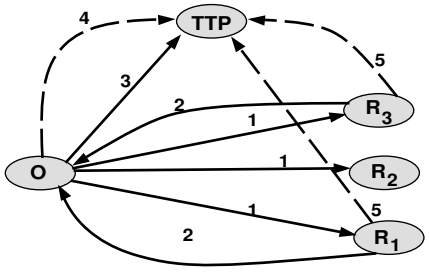

Fig. 1. Protocol with different messages

1. $O \rightarrow R_i$: $f_{eoo}, R_i, TTP, l_i, h(k), t, u_{R_i}, v_i, c_i, EOO_i$ for each $R_i \in R$
2. $R_i \rightarrow O$: f_{eor}, O, l_i, EOR_i where $R_i \in R$
3. $O \rightarrow TTP$: $f_{sub}, R', L', t, E_{R'}(k), Sub_k$
4. $O \leftrightarrow TTP$: $f_{con}, O, R', L', E_{R'}(k), Con_k$
5. $R'_i \leftrightarrow TTP$: $f_{con}, O, R', L', E_{R'}(k), Con_k$ where $R'_i \in R'$

The protocol works in the following way.

Step 1: O sends to every R_i the evidence of origin corresponding to the encrypted message c_i, together with v_i. In this way, O distributes $|R|$ messages in a batch operation and each R_i gets the encrypted message as well as n_i. O selects the intended public key u_{R_i} being used in the encryption of n_i. If R_i disagrees (i.e., its digital certificate has expired or been revoked), it should stop the protocol at this step. There is no breach of fairness if the protocol stops at step 1 because c_i cannot be obtained without key k.

Step 2: Some entities (or all of them) send evidence of receipt of c_i back to O after checking evidence and labels. Again, there is no breach of fairness if the protocol stops.

Step 3: O sends k and Sub_k to the TTP in exchange for Con_k. The key k is encrypted using a group encryption scheme where the group of users is R'. Hence, only those entities belonging to R' will be able to decrypt and extract the key. Before confirming the key, the TTP checks that $|R'| = |L'|$ holds and the current time is earlier than t.

Step 4: O fetches $E_{R'}(k)$ and Con_k from the TTP and saves it as the evidence to prove that k is available to R'.

Step 5: Each R_i fetches $E_{R'}(k)$ and Con_k from the TTP. They will obtain k_i by computing k xor n_i. Also, they save Con_k as the evidence to prove that k originated from O.

We assume that the communication channels between the TTP and O as well as each R_i are not permanently broken. Therefore, O and R'_i will eventually be able to retrieve the messages from the TTP at steps 4 and 5, respectively.

2.2 Dispute Resolution

Two kinds of disputes can arise: repudiation of origin and repudiation of receipt. Repudiation of origin arises when a recipient R_i claims having received a message M_i from an originator O who denies having sent it. Repudiation of receipt arises when the originator O claims having sent a message M_i to a recipient R_i who denies having received it.

Repudiation of Origin. If O denies sending M_i, R_i can present evidence EOO_i and Con_k plus $(t, u_{R_i}, v_i, c_i, n_i, k, E_{R'}(k), M_i, R', L')$ to the arbitrator. The arbitrator will check

- $v_i = E_{u_{R_i}}(n_i)$
- $k_i = k$ xor n_i
- $c_i = E_{k_i}(M_i)$
- $l_i = h(O, R_i, TTP, h(c_i), h(k))$
- O's signature EOO_i
- TTP's signature Con_k

Repudiation of Receipt. If R_i denies receiving M_i, O can present evidence EOR_i and Con_k plus $(t, u_{R_i}, v_i, c_i, n_i, k, E_{R'}(k), M_i, R', L')$ to the arbitrator. The arbitrator will check

- $R_i \in R'$
- $v_i = E_{u_{R_i}}(n_i)$
- $k_i = k$ xor n_i
- $c_i = E_{k_i}(M_i)$
- $l_i = h(O, R_i, TTP, h(c_i), h(k))$
- R_i's signature EOR_i
- TTP's signature Con_k

It is important to note that the verification of $v_i = E_{u_{R_i}}(n_i)$ can be carried out by the arbitrator alone only if a deterministic asymmetric public encryption algorithm is applied. Otherwise, if a non-deterministic algorithm is used (e.g., ElGamal cryptosystem [6]), either the recipient should prove $v_i = E_{u_{R_i}}(n_i)$ to the arbitrator, or the originator should provide the arbitrator with the random seed used in encryption.

2.3 Efficiency

We compare our approach with the one where an n-instance of a two-party protocol [20] is used in order to send messages to the intended parties. The efficiency of the three principal entities participating in the protocol is analyzed,

using an operation comparison. For this comparison we will use the following basic operations:

- signature generation and verification
- generation of random numbers
- asymmetric encryption and decryption
- modular equation computation
- store and fetch operation

Depending on which algorithm is chosen for each of these operations, the bit complexity (as well as the bandwidth requirements) of each of the participating entity will change, although the relation going between them remains.

We denote:

$|R| = N$
$|R'| = N'$ (with $N' \leq N$)
$\approx$ roughly equal
$>$ or $<$ greater or smaller
$\gg$ or $\ll$ much greater or smaller

Table 1
O's Computation Complexity

n-instanced two-party		**Our approach**
Evidence of origin EOO_i	$=$	**EOO_i**
N signatures.		N signatures.
Generation of k_i	$\approx$	**Generation of n_i plus k**
Evidence of submission Sub_{k_i}	$\gg$	**Sub_k**
N' signatures.		1 signature.
Encrypted key $E_{u_{R_i}}(k_i)$	$\ll$	**Encrypted key $E_{R'}(k)$ plus $E_{u_{R_i}}(n_i)$**
N' asymmetric encryptions.		N'+N asymmetric encryptions.
N fetches operations of Con_{k_i}	$\gg$	**One fetch operation of Con_k**

Table 2
$R_i's$ Computation Complexity

n-instanced two-party		**Our approach**
Evidence of receipt EOR_i	$=$	**EOR_i**
Fetch k_i and Con_{k_i}	$=$	**Fetch k and Con_k**
Obtain k_i	$<$	**Obtain k plus n_i**
Decrypts $E_{u_{R_i}}(k_i)$.		Decrypt $E_{u_{R_i}}(k)$.
		Decrypt $E_{u_{R_i}}(n_i)$.

Hence we can see in table 3 the TTP's efficiency is improved when it is generalized to multiple entities. Since communicating entities will usually pay for the TTP services, we achieve a more efficient and cheaper TTP service. In addition, we can see in tables 1 and 2 that O's efficiency is improved too, while

Table 3
TTP's Computation Complexity

n-instanced two-party		Our approach
Store N' keys	$\gg$	**Store only one key**
Generation of N' evidences Con_{k_i}	$\gg$	**Generation of only one evidence** Con_k

R_i's is slightly increased. However, if the originator and the recipients have any kind of previous relation between them and they all share a secret, then the encryption of n_i could be avoided in each protocol run though it should be still included in evidence.

3 An Optimistic MPNR Protocol for Exchange of Different Messages

As we noted in section 1, there is an optimistic approach in non-repudiation protocols where the entities are likely to behave honestly, thus giving priority to the main protocol and running sub-protocols only in case that an exception arises. Here we present an optimistic multi-party non-repudiation protocol based on [7] [6], and use the same solution described in the previous section for the privacy of different messages.

As defined in [19], new properties for fair exchange (and also desirable in non-repudiation protocols) are

Effectiveness. If two parties behave correctly, they will receive the expected evidence without any involvement of the TTP.

Verifiability of Third Party. If the third party misbehaves, resulting in the loss of fairness for an entity, the victim can prove the fact in a dispute.

In a comparable work [13], Markowitch et al. proposed a protocol for distribution of the same message to several parties with a non-transparent TTP. In their protocol, four steps are required in the main exchange, which is not optimized. (As we will see, the main exchange can be reduced to only three steps in our protocol.) Their protocol also makes use of a pre-defined time constraint thus does not achieve asynchronous timeliness. In addition, their protocol employs an inefficient ftp operation with the originator acting as a server.

We assume that the communication channels to and from the TTP are not

[6] The protocol [7] has some problems as being identified in [18]. Those problems have been corrected in the design of our new protocol.

permanently broken, which means that messages are delivered after an arbitrary but finite amount of time. Some additional notation in the protocol description is as follows.

- $R'' = R - R'$: a subset of R (in plaintext) with which O wants to cancel the exchange
- $R''\_finished$: a subset of R'' that have finished the exchange with the finish sub-protocol
- $R''\_cancelled = R'' - R''\_finished$: a subset of R'' with which the exchange has been cancelled by the TTP
- $l = h(c_1, c_2, .., k)$: label [7] that identifies the protocol run computed as the hash outcome of the concatenation of every encrypted message plus the key k
- $k_T = E_{u_{TTP}}(k)$: key k encrypted with the TTP's public key [8]
- $EOO_i = S_O(f_{eoo}, R_i, TTP, k_T, l, v_i, u_{R_i}, h(c_i), h(k))$: evidence of origin for R_i
- $EOR_i = S_{R_i}(f_{eor}, O, TTP, k_T, l, v_i, u_{R_i}, h(c_i), h(k))$: evidence of receipt from each R_i
- $Sub_k = S_O(f_{sub}, l, k)$: evidence of submission of the key to recipients
- $Cancel_{req} = S_O(TTP, R'', l)$: evidence of request of cancellation issued by the originator to the TTP
- $Cancel_O = S_{TTP}(O, l, R'', R''\_cancelled, Cancel_{req})$: evidence of cancellation issued by the TTP to the originator
- $Cancel_{R_i} = S_{TTP}(R_i, l, EOR_i, Cancel_{req})$: evidence of cancellation issued by the TTP to R_i
- $Con_k = S_{TTP}(R_i, l, k)$: confirmation evidence of k issued by the TTP

3.1 The Protocol

The protocol consists of a *main* protocol (which will be the only one executed by the entities in the normal situation) and two sub-protocols: *cancel* and *finish* (see figure 2). The TTP is only involved in the sub-protocols in case of any participant's misbehavior or channel failure between the originator and the recipients. Any participant can initiate the corresponding sub-protocols to terminate a protocol run at any time without loss of fairness.

The *main* protocol executed by the final entities is

[7] Note that with the reduction to 3 steps in the main protocol, the attack proposed in [16] does not work in our approach.

[8] To prevent attacks on such a ciphertext, the encryption scheme should provide non-malleability, i.e., it should be impossible to modify this ciphertext to construct a different meaningful related ciphertext.

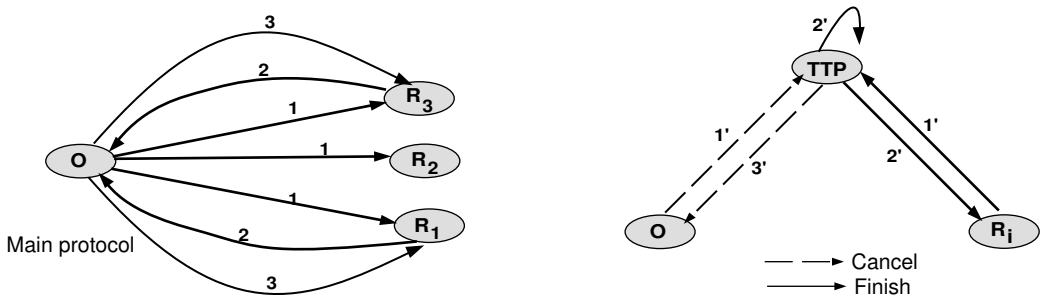

Fig. 2. Optimistic protocol with different messages

1. $O \rightarrow R_i : f_{eoo}, R_i, TTP, k_T, l, v_i, u_{R_i}, c_i, h(k), EOO_i$ for each $R_i \in R$
2. $R_i \rightarrow O : f_{eor}, O, l, EOR_i$ where $R_i \in R$
3. $O \Rightarrow R' : f_{sub}, l, E_{R'}(k), Sub_k$

In step 1, the originator sends to each recipient its message encrypted with k_i. A recipient can derive k_i from k and a random number n_i which are also sent in this step (in a confidential way). Note that the TTP is included in this step, thus there is no confusion about which TTP to use in case they have to launch any of the sub-protocols. The originator picks each receiver's public key. If any recipient does not want to use such a key (e.g., the correspondent public key certificate has been revoked), then it stops the protocol. Otherwise, after verifying the data obtained, the recipient sends to the originator evidence of receipt at step 2 and the originator sends to the set of recipients who replied after a reasonable amount of time at step 3, the key and evidence of submission of that key, as a second part of evidence of origin. If O did not receive a correct message 2 from some of the recipients R'', O may initiate the following *cancel* sub-protocol:

$1'. O \rightarrow TTP : TTP, R'', l, Cancel_{req}$
$2'. TTP$ FOR (all $R_i \in R''$)
IF ($R_i \in R''\_finished$) THEN retrieves EOR_i
ELSE appends R_i into $R''\_cancelled$
$3'. TTP \rightarrow O$: all retrieved EOR_i, $R''\_cancelled$, $Cancel_O$

In this case the originator communicates the TTP its intention of revoking the protocol with entities contained in R'' and for the protocol run labelled l. After verifying O's cancel request, the TTP checks which entities previously resolved the protocol and gets their proofs of receipt. The TTP generates an evidence of cancellation for the rest of entities and includes everything in a message destined to the originator. We do not send any information about the message, keys and EOO as [7] does because we use a well-defined label for indexing purposes on the TTP side.

Note that the originator does not have any interest in sending a subset $\bar{R}'' \neq R''$

when requesting to the TTP for cancellation. If $\bar{R''} \supset R''$, then O will cancel the exchange with those $R_i \subset \bar{R''} - R''$ that have replied with EOR_i. If $\bar{R''} \subset R''$, then $R_i \subset R'' - \bar{R''}$ may invoke the *finish* sub-protocol below to get the key k but O will not obtain EOR_i from the TTP at the above *cancel* sub-protocol.

If some recipient R_i did not receive message 3, R_i may initiate the following *finish* sub-protocol:

$1'.\ R_i \rightarrow TTP : TTP, k_T, l, v_i, u_{R_i}, h(c_i), h(k), EOO_i, EOR_i$
$2'.\ TTP \rightarrow R_i :$ IF $(R_i \in R''\_cancelled)$ THEN $R_i, l, R'', Cancel_{req}, Cancel_{R_i}$
ELSE $\{R_i, l, E_{u_{R_i}}(k), Con_k$
appends R_i into $R''\_finished$ and stores $EOR_i\}$

The recipient sends to the TTP all the information that it has already got from the originator along with its evidence of receipt. If this entity does not belong to the group of entities with which the originator has cancelled the exchange, the TTP verifies all the information (digital signatures) and decrypts k_T, getting the key for the recipient. It also stores EOR_i. Note that if the protocol has been cancelled, then it should be impossible for the recipient to cheat the TTP in a way that the TTP reveals the key k for that protocol run. For such a reason, the TTP must verify O's signature in the first step and check that l and k_T provided by the recipient fits with the information contained in EOO_i.

Otherwise, the TTP sends a cancellation evidence to the recipient such that the latter can easily demonstrate to an arbitrator that the exchange was cancelled in case a dispute arises. This evidence includes the request of cancellation, such that the TTP's behavior is verifiable while the TTP need not store all the request evidences from the originator.

3.2 Dispute Resolution

As we have mentioned, two kinds of disputes can arise. Here we further discuss the rules for their resolution.

Repudiation of Origin. If O denies sending M_i, R_i can present evidence EOO_i and Sub_k (or Con_k) plus $(TTP, l, u_{R_i}, v_i, c_i, n_i, k, k_T, M_i)$ to the arbitrator. The arbitrator will check

- $v_i = E_{u_{R_i}}(n_i)$
- $k_i = k$ xor n_i
- $c_i = E_{k_i}(M_i)$
- O's signature on EOO_i
- O's signature on Sub_k, or TTP's signature on Con_k

- k certified in Sub_k (or Con_k) matches $h(k)$ certified in EOO_i

Repudiation of Receipt. If R_i denies receiving M_i, O can present evidence EOR_i plus $(TTP, l, u_{R_i}, v_i, c_i, n_i, k, k_T, M_i)$ and $(R'', R''\_cancelled, Cancel_{req}, Cancel_O)$ if it has. The arbitrator will check

- $v_i = E_{u_{R_i}}(n_i)$
- $k_i = k$ xor n_i
- $c_i = E_{k_i}(M_i)$
- R_i's signature on EOR_i
- k matches $h(k)$ certified in EOR_i
- TTP's signature on $Cancel_O$ and $R_i \notin R''\_cancelled$

O will win the dispute if all the above checks are positive. If all the checks, but the last, are positive and O cannot present evidence $Cancel_O$, the arbitrator must further interrogate R_i. If the latter cannot present $Cancel_{R_i}$ (or this token is not properly constructed including $Cancel_{req}$ from O), O also wins the dispute. Otherwise, R_i can repudiate having received the message M_i.

We can also see that evidence provided by the TTP is *self-contained*, that is, the TTP need not to be contacted in case a dispute arises regarding the occurrence or not of the *cancel* sub-protocol launched by O. Thus, the TTP is efficiently verifiable.

3.3 Protocol Extensions

Transparent TTP. Our protocol can be modified such that the recipient can obtain the same evidence of origin even in case it needs to launch the *finish* sub-protocol. For this, we only have to make possible for the TTP to send O's signature Sub_k whenever the recipients try to fetch the key. In this way, external parties will not be able to distinguish if the recipient launched the *finish* sub-protocol, since the evidence obtained is the same. It helps to preserve O's reputation in case of channel failures. We simply redefine k_T as follows.

$$k_T = E_{u_{TTP}}(k, Sub_k)$$

In the *finish* sub-protocol, the TTP decrypts k_T and additionally checks that Sub_k is O's signature on (f_{sub}, l, k). If the TTP succeeds in all the checks, then it provides Sub_k to the recipient instead of Con_k.

Message Confidentiality. Our protocol has already fulfilled the confidentiality requirement in respect of external attackers. However, if we also want to keep the message confidential to the TTP as well, the originator needs to

transmit c_i to the recipients in a private way (e.g., via private channel such as SSL).

3.4 Efficiency

This protocol is very efficient in case of a good behavior of the participating entities. (In fact, 3 steps are the minimum number of steps we could reach without breaking fairness in non-repudiation protocols.) Even with multiple recipients for exchange of different messages, it manages to use only one key for evidence distribution, thus decreasing the computation and verification requirements for the originator and the TTP. For this new feature, public key encryption and decryption of temporal random numbers are the main extra cost added.

It is straightforward to see that this protocol is more efficient than any combination of two-party protocols, since it permits to send different messages in a confidential way to multiple entities as well as to cancel the protocol for a group of entities R'' in only one run of the *cancel* sub-protocol. In addition, this protocol achieves asynchronous timeliness, as each entity can terminate, if needed, the protocol at any time at their own discretion while maintaining fairness.

4 Further Discussions

The proposed approaches for multi-party non-repudiation protocols considerably improve the number of messages exchanged as well as the amount of evidence collected by the final entities involved. However, they are the first effort to generalize the non-repudiation service to multiple entities and might be reviewed to further improve their efficiency.

4.1 Group Encryption

Along the description of the protocols presented in previous sections, we have been using the notation $E_{R'}(k)$ to define an encryption operation over the key k intended for a group R'. In the multi-party non-repudiation protocol proposed in [11], a group encryption scheme [4] is used. It is based on a public key encryption scheme and on the Chinese Remainder Theorem (CRT). As the authors explained, it is efficient only when the number of users is small, since the time to compute the CRT and its length (hence transmission time)

is proportional to the number of users. This method is generic as it can use any public key cryptosystem:

- Let u_{R_i} and $\overline{u}_{R_i}$ be the public and private keys of R_i, respectively (where i corresponds to all parties that belong to R').
- Each recipient of R' receives a random integer $P_i > E_{u_{R_i}}(k)$ such that all P_i are pair-wise relatively prime. (When choosing randomly large primes or multiplications of distinct primes for example, the probability of obtaining two numbers that are not relatively primes is negligible.)
- O computes $X \equiv E_{u_{R_i}}(k)\ mod\ P_i$. As all of P_i are prime integers, using the CRT, only one solution is obtained from this equation. Hence, $E_{R'}(k) \equiv X$. Each recipient R_i can obtain k by computing $X \equiv E_{u_{R_i}}(k)\ mod\ P_i$ using her private key $\overline{u}_{R_i}$.

From a computational point of view, we know that the CRT is an additional effort to the encryption with each user's public key. Analyzing the CRT properties we realize that X can be of the same magnitude as $\prod_{i=1}^{m} P_i$ with m the number of users in the group R'. Besides, from the second step above we know that $P_i > E_{u_{R_i}}(k)$ for every i, so the length of the message distributed to the recipients is still long. Although it is direct to prove that the length of a message M such that $M = M_1 M_2 \cdots M_n$ (a simple concatenation of messages) with M_i of the same length is greater than M' such that $M' = M_1 * M_2 * \cdots M_n$, it is also true that P_i needs to be much greater than $E_{u_{R_i}}(k)$ to avoid possible problems in future encryptions, thus making M and M' approximately of the same size.

As a result, the distribution length improvement of message X with the CRT-based group encryption is not significant. Furthermore, in that scheme, the authors assume a model in which each recipient R_i has already got the random numbers P_i. However, if we assume that the originator had no prior contact with the recipients, the random numbers have to be distributed by the originator in each protocol run, thus losing any possible advantage. For these reasons we remove the CRT operations and define the group encryption operation as a straightforward concatenation of public key encryptions to each final recipient as follows:

$$E_{R'}(k) = E_{u_{R_1}}(k), E_{u_{R_2}}(k), \cdots, E_{u_{R_m}}(k)$$

4.2 Optimization based on VPN

As each topology has its own requirements, our protocols should be adapted to fulfill these requirements, and moreover, adapted to take advantage of the new features.

Basically, a VPN is a private network that uses a public network (usually the Internet) to connect remote sites or users together. Instead of using a dedicated connection such as a leased line, a VPN uses "virtual" connections routed through the Internet. Remote access to VPNs permits secure, encrypted connections between a private network and remote users through a third-party service provider (e.g., *Virtual Private Dial-up Network*) or between more private networks (e.g., *Intranet-based and Extranet-based VPN*).

Inside a VPN, there is typically an AAA (*Authentication, Authorization and Accounting*) server that distributes symmetric keys to a user for confidential communications with other users in the VPN. Therefore a sender can use the symmetric keys to encrypt the messages for the recipients, and all the encryption operations with the recipient's public key used in our previous protocols can be replaced. For example, v_i was defined as $v_i = E_{u_{R_i}}(n_i)$. Now, n_i can be sent to each recipient without public key encryption since the channel between a pair of entities is ciphered in a VPN. This improvement on the computational overheads is applied $|R|$ (number of recipients) times in one protocol run for each public key encryption that we used. In addition, the group encryption scheme can also be changed to take advantage of the symmetric keys in the VPN, i.e., $E_{R'}(k) = E_{sk_1}(k), E_{sk_2}(k), \cdots, E_{sk_m}(k)$ where sk_i is the key shared between the sender and each recipient. With such an optimization, efficiency of final entities in our protocols is further improved.

5 Conclusion

The aim of this paper is to extend the traditional two-party non-repudiation protocols for multiple entities. This is because the two-party instance of this service in multi-party applications seems to be too heavy considering the number of network messages as well as the amount of evidence that the final entities have to manage.

At the beginning of this paper, we clearly defined the properties that a non-repudiation protocol is required to respect. Following the first research conducted on multi-party non-repudiation protocols proposed in [11,13], we extended that scenario such that sending different messages to different entities in only one transaction is possible. This protocol uses a light-weight on-line TTP in every transaction. However, there are situations in which the final entities could be willing to launch a non-repudiation protocol without the TTP's assistance (either because it is expensive for the TTP's service or because the entities have some kind of trust to each other) unless an error or channel failure occurs. For these situations we designed an optimistic protocol that uses only three steps in its main protocol.

As any non-repudiation protocol design, we further discussed the dispute resolution process and the efficiency matters of each design. We also reviewed the cryptographic primitives used in the construction of each protocol.

References

[1] N. Asokan, B. Baum-Waidner, M. Schunter, M. Waidner, Optimistic synchronous multi-party contract signing, Tech. Rep. RZ 3089, IBM Zurich Research Laboratory(1998).

[2] N. Asokan, M. Schunter, M. Waidner, Optimistic protocols for multi-party fair exchange, Tech. Rep. RZ 2892, IBM Zurich Research Laboratory (1996).

[3] B. Baum-Waidner, Optimistic asynchronous multi-party contract signing with reduced number of rounds, in: Proceedings of 28th International Colloquium on Automata, Languages and Programming (ICALP'01), Springer, 2001, pp. 898–911.

[4] G. Chiou, W. Chen, Secure broadcasting using the secure lock, IEEE Transaction on Software Engineering 15 (8) (1989).

[5] N. G.-D. Collell, O. Markowitch, Exclusion-freeness in multi-party exchange protocols, in: Proceedings of 5th International Conference on Information Security (ISC'02), Vol. 2433 of Lecture Notes in Computer Sciences, Springer, 2002, pp. 200–209.

[6] T. ElGamal, A public-key cryptosystem and a signature scheme based on discrete logarithms, IEEE Transactions on Information Theory 4 (1985) 469–472.

[7] J. L. Ferrer-Gomila, M. Payeras-Capellà, L. Huguet-Rotger, A realistic protocol for multi-party certified electronic mail, in: Proceedings of 5th International Conference on Information Security (ISC'02), Vol. 2433 of Lecture Notes in Computer Sciences, Springer, 2002, pp. 210–219.

[8] M. Franklin, G. Tsudik, Secure group barter: Multi-party fair exchange with semi-trusted neutral parties, in: Proceedings of Financial Cryptography 1998, Vol. 1465 of Lecture Notes in Computer Science, Springer, 1998, pp. 90–102.

[9] N. González-Deleito, O. Markowitch, An optimistic multi-party fair exchange protocol with reduced trust requirements, in: Proceedings of 4th International Conference on Information Security and Cryptology (ICISC'01), Vol. 2288 of Lecture Notes in Computer Science, Springer, 2001, pp. 258–267.

[10] J. Kim, J. Ryou, Multi-party fair exchange protocol using ring architecture model, in: Proceedings of Japan-Korea Joint Workshop on Information Security and Cryptology, 2000.

[11] S. Kremer, O. Markowitch, A multi-party non-repudiation protocol, in: Proceedings of 15th IFIP International Information Security Conference, Kluwer Academic Publishers, 2000, pp. 271–280.

[12] S. Kremer, O. Markowitch, J. Zhou, An intensive survey of fair non-repudiation protocols, Computer Communications 25 (17) (2002) 1606–1621.

[13] O. Markowitch, S. Kremer, A multi-party optimistic non-repudiation protocol, in: Proceedings of 3rd International Conference on Information Security and Cryptology (ICISC'00), Vol. 2015 of Lecture Notes in Computer Science, Springer, 2000, pp. 109–122.

[14] J. Onieva, J. Zhou, M. Carbonell, J. Lopez, A multi-party non-repudiation protocol for exchange of different messages, in: Proceedings of 18th IFIP International Information Security Conference, Kluwer Academic Publishers, 2003, pp. 37–48.

[15] J. Onieva, J. Zhou, M. Carbonell, J. Lopez, Intermediary non-repudiation protocols, in: Proceedings of IEEE Conference on Electronic Commerce, IEEE Computer Society Press, 2003, pp. 207–214.

[16] S. Gürgens, C. Rudolph, Security analysis of (un-) fair non-repudiation protocols, in: Proceedings of 2002 Formal Aspects of Security Conference, Vol. 2629 of Lecture Notes in Computer Sciences, Springer, 2002, pp. 97–114.

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

[18] J. Zhou, On the security of a multi-party certified email protocol, manuscript, I2R, Singapore (2003).

[19] J. Zhou, R. Deng, F. Bao, Some remarks on a fair exchange protocol, in: Proceedings of International Workshop on Practice and Theory in Public Key Cryptography (PKC'00), Vol. 1751 of Lecture Notes in Computer Science, Springer, 2000, pp. 46–57.

[20] J. Zhou, D. Gollmann, A fair non-repudiation protocol, in: Proceedings of IEEE Symposium on Security and Privacy, IEEE Computer Society Press, 1996, pp. 55–61.