A MULTI-PARTY NON-REPUDIATION PROTOCOL FOR EXCHANGE OF DIFFERENT MESSAGES

Jose Antonio Onieva*, Jianying Zhou*
Mildrey Carbonell**, Javier Lopez**

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 multi-party non-repudiation protocols, and analyze the previous work where one originator is able to send the same message to many recipients. We propose a new multi-party non-repudiation protocol for sending different messages to many recipients. We also discuss the improvements achieved with respect to the multiple instances of a two-party non-repudiation protocol, and present some applications that would benefit from them.

Keywords: non-repudiation, fair exchange, group communications

1. Introduction

During the last years the impressive growth of the Internet and more generally of open networks has created several security-related problems. Non-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 case a dispute arises, an arbitrator must be able to resolve it using the evidence generated and transferred during the non-repudiation protocol.

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. Evidence should be collected to resolve this type of

*Institute for Infocomm Research, 21 Heng Keng Mui Terrace, Singapore 119613. onieva@i2r.a-star.edu.sg, jyzhou@i2r.a-star.edu.sg
**Computer Science Department, E.T.S. Ingenieria Informatica, University of Malaga, Spain, 29071 - Malaga. mildrey@crypto.lcc.uma.es, jlm@lcc.uma.es

The original version of this chapter was revised: The copyright line was incorrect. This has been corrected. The Erratum to this chapter is available at DOI: 10.1007/978-0-387-35691-4_52

D. Gritzalis et al. (eds.), *Security and Privacy in the Age of Uncertainty*
© IFIP International Federation for Information Processing 2003

disputes arisen between participating entities in an electronic commerce scenario.

An important requirement of non-repudiation services is fairness with which neither party can gain an advantage by quitting prematurely or otherwise misbehaving during a protocol [10]. In other words, either all participating entities obtain all the messages and the evidence needed, or none of them obtains items expected (i.e., the messages for the recipients and evidence of receipt for the originators).

Several solutions to fair non-repudiation have been developed [6]. Some of them use a *Trusted Third Party* (TTP) that plays the role of an intermediary 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 [9] where the TTP intervenes during each execution as a "low weight notary" rather than as an intermediary. 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 the *optimistic approach.* There are also solutions that eliminate the TTP's involvement. However, they need a strong requirement: all involved parties must have the same computational power.

Some work about multi-party scenarios in a related topic, such as *fair exchange*, where entities have to exchange items between them without loss of fairness, exists [3, 2, 4, 8]. The research towards a generalization of non-repudiation, where multiple entities may participate in the consecutive non-repudiation protocols, has not been sufficiently undertaken. Markowitch and Kremer extended the non-repudiation scenarios to allow one originator to send the same message to multiple recipients in a general non-repudiation protocol [5, 7]. In this paper, we extend their multi-party non-repudiation protocols by allowing one originator to send different messages to multiple recipients.

In this paper we classify the multi-party scenarios into two types. The first is called *SOMR-M* (simple origin, with many recipients, for exchange of the same message). The second is called *SOMR-*M_i (simple origin, with many recipients, for exchange of different messages). The two-party non-repudiation approaches could be used to provide solutions to both types of scenarios, thus creating multiple encrypted messages with different keys. However, as we will see in this paper, it is a better solution for a transaction scenario with a low-weight TTP to store only one key k by the TTP for every protocol run.

The remainder of the paper is organized as follows: In section 2 we describe the first multi-party non-repudiation protocol designed by Kremer and Markowitch. In section 3 we present a new protocol which allows different messages to be transferred to the intended recipients. In section 4 we compare the complexity between our new multi-party protocol and n-instance of a two-party protocol. In section 5 we show typical application scenarios of our approach as well as a specific adaptation of our protocol. 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
- $h(X)$: hash function
- f : a flag indicating the purpose of a message
- $\leftrightarrow$: fetch operation

2. A Fair Multi-Party Non-repudiation Protocol with Same Message

In this section, we review the extension by Kremer et al. [5] of a low weight notary protocol [9] for multi-party purposes. This SOMR-M protocol supports a one-to-many scenario with the same message.

2.1 Additional Notation

Some useful notation in the protocol description is:

- $X \Rightarrow \prod$: multicast from entity X to the set $\prod$
- M : message being sent from the originator to the recipients
- k : key being selected by the originator O
- $c = E_k(M)$: message encrypted with k
- $l = h(M, k)$: label of message M and key k
- t : a timeout chosen by O, before which the TTP has to publish some information
- R : set of intended recipients
- R' : set of recipients that replied to the originator with the evidence of receipt
- $E_{R'}(k)$: a group encryption scheme that encrypts k for the group R'
- $EOO = S_O(feoo, R, l, t, c)$: evidence of origin
- $EOR_i = S_{R_i}(feor, O, l, t, c)$: evidence of receipt of each R_i
- $Sub_k = S_O(fsub, R', l, t, E_{R'}(k))$: evidence of submission of the key to the TTP
- $Con_k = S_{TTP}(fcon, O, R', l, t, E_{R'}(k))$: evidence of confirmation of the key by the TTP

2.2 Group Encryption

Group encryption [1] is used to encrypt the message k for the recipients R' in the Kremer-Markowitch protocol. It is based on a public-key encryption scheme and on the Chinese remainder theorem. This method is generic as it can use any public-key cryptosystem.

- Let u_{R_i} and v_{R_i} be the public and private keys of R_i, respectively (i corresponds with 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 Chinese remainder theorem, 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 Pi* using her private key v_{R_i}.

2.3 The Protocol

The Kremer-Markowitch protocol is as follows.

1. $O \Rightarrow R$: $feoo, R, l, t, c, EOO$
2. $R_i \rightarrow O$: $feor, O, R_i, l, EOR_i$ where each $R_i \in R$
3. $O \rightarrow TTP$: $fsub, R', l, t, E_{R'}(k), Sub_k$
4. $R'_i \leftrightarrow TTP$: $fcon, O, R', l, E_{R'}(k), Con_k$ where each $R'_i \in R'$
5. $O \leftrightarrow TTP$: $fcon, O, R', l, E_{R'}(k), Con_k$

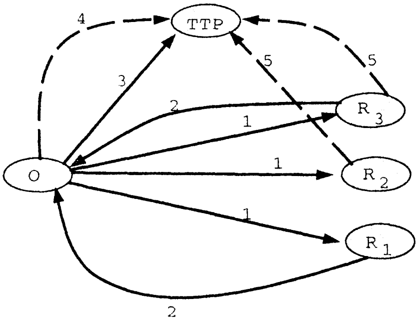

Figure 1. Protocol SOMR-M

The protocol in figure 1 uses the same key k for each recipient R_i, such that, an encrypted message c, evidence EOO, Con_k, Sub_k are generated for each protocol run. This solution claims to disclosure the key only to those recipients (R' ∈ R) that send evidence of receipt in order to achieve fairness. The solution uses a public-key group encryption scheme $E_{R'}(k)$, such that only those included in R' will be able to access to the key k.

This protocol broadcasts a message among several entities. 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.

3. A Fair Multi-Party Non-repudiation Protocol with Different Messages

To the best of our knowledge, the SOMR-M_i scenario (Simple Origin Multiple Recipients with i different messages) has not been reported in the literature yet. We propose a new protocol that extends the Kremer-Markowitch SOMR-M protocol to eliminate the restriction on the exchange of the "same-message". In other words, personal and confidential messages could be sent to multiple entities.

It is important to realize that the use of the same key by multiple recipients could be justified in several ways. Basically, this depends on the application, which is specifically discussed in section 5. Sending messages (same or different) 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 involved.

In this extension, the use of the same key for all users creates a new problem that did not appear in the SOMR-M protocol. As messages are different, when the same key 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). In this section we propose a solution to this problem.

Let R be the group of recipients and M_i the different plain messages that O wants to send to each R_i, with i ∈ $\{1..|R|\}$. A random value n_i is generated for each recipient R_i. Let

$v_i = E_{u_{R_i}}(n_i)$ be the encryption of n_i with R_i's public key

$k_i = k \; xor \; n_i$ be a key for each R_i

$c_i = E_{k_i}(M_i)$ be the encrypted message with a key k_i for each R_i

3.1 Additional Notation

Some useful notation in the protocol description is:

- $X \Rightarrow \prod$: multicast from entity X to the set $\prod$
- M_i : message being sent from the originator O to the recipient R_i
- k : key being selected by O
- $c_i = E_{k_i}(M_i)$: encrypted message for R_i
- $l_i = h(M_i, k)$: label of message M_i
- t : a timeout chosen by O, before which the TTP has to publish some information
- R : set of intended recipients
- R' : set of recipients that replied to the originator with the evidence of receipt
- L' : labels of all the messages being sent to R'
- $E_{R'}(k)$: a group encryption scheme that encrypts k for the group R'
- $EOO_i = S_O(feoo, R_i, l_i, t, v_i, u_{R_i}, c_i)$: evidence of origin
- $EOR_i = S_{R_i}(feor, O, l_i, t, v_i, u_{R_i}, c_i)$: evidence of receipt of each R_i
- $Sub_k = S_O(fsub, R', L', t, E_{R'}(k))$: evidence of submission of the key to the TTP. In this case (R',L') denotes concatenation of R_i with corresponding l_i
- $Con_k = S_{TTP}(fcon, O, R', L', t, E_{R'}(k))$: evidence of confirmation of the key by the TTP

3.2 The Protocol

Here, we describe the protocol:

1. $O \Rightarrow R_i$: $feoo, R_i, l_i, t, v_i, c_i, EOO_i$
2. $R_i \rightarrow O$: $feor, O, l_i, EOR_i$ where each $R_i \in R$
3. $O \rightarrow TTP$: $fsub, R', L', t, E_{R'}(k), Sub_k$
4. $R'_i \leftrightarrow TTP$: $fcon, O, R', L', E_{R'}(k), Con_k$ where each $R'_i \in R'$
5. $O \leftrightarrow TTP$: $fcon, O, R', L', E_{R'}(k), Con_k$

The protocol works in the following way.

Step 1: O sends to R_i the evidence of origin corresponding to the encrypted message c_i, together with v_i. In this way, R_i has the encrypted message as well as n_i. 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. Again, there is no breach of fairness if the protocol stops.

Step 3: O sends k and Sub_k to the TTP in order to obtain Con_k from the TTP at step 5. As we assume that the communication channel between O and the TTP is not permanently broken, O will be eventually able to send 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.

Step 4: Each recipient R_i fetches $E_{R'}(k)$ and Con_k from the TTP. They will obtain k_i by computing the expression: k xor n_i. Also, they save Con_k as the evidence to prove that k originated from O.

Step 5: 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'.

3.3 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 $(R_i, t, u_{R_i}, v_i, c_i, n_i, k, E_{R'}(k), M_i, R', L')$ to the arbitrator. The arbitrator will check:

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

Hence, all possible disputes can be resolved if any of the entities misbehaves during the protocol, using the generated and stored evidence as well as an arbitrator that checks the validity of the evidence.

4. Complexity Comparison

We compare our approach with the one where an n-instance of a two-party protocol [9] is used in order to send messages to the intended parties. The complexity of the computation of three principal entities participated 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 encryptions and decryptions
- modular equation computation
- store and fetch operations

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.

SOMR-M_i vs. extension of n-instanced two-party protocol. We denote:

$|R| = N$

$|R'| = N'$ (with $N' \leq N$)

$>$ greater

$\gg$ much greater

Hence we can see in table 3 the TTP's computation complexity is reduced when it is generalized to multiple entities. This extension can be used when no overload is possible for the TTP with multiple participating entities, and in scenarios where it is better that the TTP stores only one entry per transaction. 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 computation complexity is reduced too while R_i's is slightly increased.

5. Applications

Our approach fits better in shopping scenarios (i.e. O is a custom and R_i are merchants). Here we present a possible scenario:

n-instanced two-party		**SOMR-M_i**
Evidence of origin EOO_i	$=$	**Evidence of origin 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.
$E_{u_{R_i}}(k_i)$ Encrypted key k_i	$\ll$	**$E_{R'}(k)$ Encrypted key k plus $E_{u_{R_i}}(n_i)$**
N' encryption operations with each public key.		N' random numbers P_i. N' encryption operations with each public key. Compute equation $X \equiv E_{u_{R_i}}(k_i)\ mod\ P_i \rightarrow \Theta(\lg n)$. N encrypt operations with each public key (v_i).
N fetches operations of Con_{k_i}	$\gg$	**One fetch operation**

Table 1. ORIGINATOR'S COMPUTATION COMPLEXITY

n-instanced two-party		**SOMR-M_i**
Evidence of receipt EOR_i	$=$	**Evidence of receipt EOR_i**
Fetch k and Con_{k_i}	$=$	**Fetch k and Con_k**
Obtain k_i	$<$	**Obtain k plus n_i**
Decrypts $E_{u_{R_i}}(k_i)$.		compute equation $X \equiv E_{u_{R_i}}(k_i)\ mod\ P_i$. Decrypt $E_{u_{R_i}}(k_i)$. Decrypt $E_{u_{R_i}}(n_i)$.

Table 2. $R_i'S$ COMPUTATION COMPLEXITY

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

Table 3. TTP'S COMPUTATION COMPLEXITY

Suppliers and customers. In B2B scenarios, we can usually find established relations between companies, such that some of them play the role of suppliers and the others apply for supplies. Frequently, these companies need to stock up vast amounts of products. For example, an electronic equipment producer has to apply for cables, metals, sockets, etc. and send an order to various suppliers. These suppliers do not mind who of them supplies the order received; in other words, they can cooperate. Typically, in these scenarios, collusion between suppliers is not a usual matter.

We can classify the orders destined to N suppliers that supply similar or different products. Figure 2 shows a scenario with one customer (C1) and four suppliers (Si), where C1 sends two requests, one for suppliers 1, 2, and 3, and the other for suppliers 1, 3 and 4.

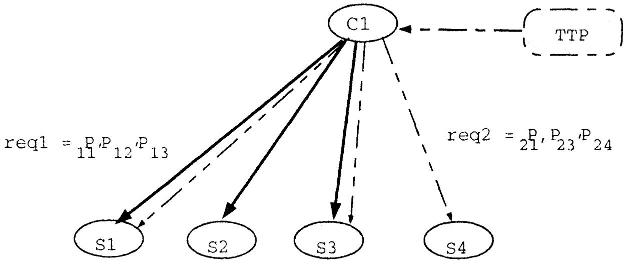
C1
TTP
req1 = P_{11}, P_{12}, P_{13}
req2 = P_{21}, P_{23}, P_{24}
S1
S2
S3
S4

Figure 2. Customers and suppliers

One-to-many protocols using the same key offer a more efficient solution than the one with an n-instantiated two-party protocol. Each transaction (order) is registered in the TTP with a key plus the evidence of this key.

Actually, B2B architecture is growing fast on Internet. It can accelerate the business process among customers and suppliers, and increase the participation of suppliers, products, and on-line supplies (i.e. software, music files, etc...). Obviously, this architecture has stronger security requirements than B2C architecture in that it usually involves high amounts of money. Some solutions have been developed, most of them are based on the *Virtual Private Network* (VPN) that provides secure connection between customers and suppliers.

We can find a good example in the *Automotive Network eXchange*[1] (ANX) . The ANX network is used for mission critical business transactions by leading international organizations and net markets in aerospace, automotive, chemical, electronics, financial services, healthcare, logistics, manufacturing, transportation and related industries. Through a global standard that assures the highest levels of security and quality, the ANX network offers connected customers the most reliable multi-vendor extranet and Virtual Private Network services available today.

[1] http://www.anx.com

If SOMR-M_i is designed over a VPN, it may use the secret session keys managed by the VPN, and there is no need to generate the random numbers and send the encrypted random numbers to the recipients. Thus, we can reduce the operation complexity of the originator and recipients (see table 4 and 5). If it is not possible to use the secret session keys (due to the limitation on the random numbers being used in the group encryption scheme), the originator only needs to generate n_i for each entity and send them to the recipients through a confidential channel provided by the VPN. That means n_i need not be encrypted with R_i's public key, thus the operation complexity of the originator and the recipients is also reduced.

n-instanced two-party		SOMR-M_i
Evidence of origin EOO_i	$=$	**Evidence of origin EOO_i**
N signatures.		N signatures.
Generation of k_i	$\gg$	**Generation of only one key k**
Generation of N random numbers.		Generation of only one number.
Evidence of submission Sub_{k_i}	$\gg$	**Sub_k**
N' signatures.		1 signature.
$E_{u_{R_i}}(k_i)$ Encrypted key k_i	$<$	**$E_{R'}(k)$ Encrypted key k**
N' encryption operations with each public key.		N' random numbers P_i. N' encryption operations with each public key. Compute equation $X \equiv E_{u_{R_i}}(k_i)$ *mod* $P_i \to \Theta(\lg n)$.
N fetches operations of Con_{k_i}	$\gg$	**One fetch operation**

Table 4. ORIGINATOR'S COMPUTATION COMPLEXITY

n-instanced two-party		SOMR-M_i
Evidence of receipt EOR_i	$=$	**Evidence of receipt EOR_i**
Fetch k and Con_{k_i}	$=$	**Fetch k and Con_k**
Obtain k_i	$\approx$	**Obtain k**
Decrypts $E_{u_{R_i}}(k_i)$.		compute equation $X \equiv E_{u_{R_i}}(k_i)$ *mod* P_i. Decrypt $E_{u_{R_i}}(k_i)$.

Table 5. $R_i'S$ COMPUTATION COMPLEXITY

6. Conclusions and Future Work

The research on non-repudiation protocols with multiple entities is still in its initial stage. Although many two-party solutions that can be instantiated for multi-party scenarios have been developed, more efficient and adapted solutions are needed.

In this paper, we analyzed the previous work on multi-party scenarios where one originator sends a message to multiple recipients. We suggested an improvement that allows the originator to send different messages to the recipients. This is the further generalization of a two-party fair non-repudiation protocol, which also reduces the computation complexity compared with the n-instanced two-party protocol. We identified applications for B2B scenarios that could make use of our protocol, and explained how these applications would benefit from it.

Further reducing the recipient's operation complexity could be carried out in a future work. It is also possible to extend our idea of SOMR-M_i to an *optimistic* multi-party fair non-repudiation protocol [7]. Other topologies and scenarios with multiple entities participating in a non-repudiation protocol will be studied as well.

References

[1] G. Chiou and W. Chen. *Secure broadcasting using the secure lock.* IEEE Transaction on Software Engineering, Vol. 15, No. 8, August 1989.

[2] M. Franklin and G. Tsudik. *Secure group barter: multi-party fair exchange with semi-trusted neutral parties.* Proceedings of Financial Cryptography 1998, Lecture Notes in Computer Science 1465, February 1998.

[3] N. Gonzalez-Deleito and O. Markowitch. *An optimistic multi-party fair exchange protocol with reduced trust requirements.* Proceedings of 4th International Conference on Information Security and Cryptology (ICISC 2001), Lecture Notes in Computer Science 2288, Seoul, Korea, December 2001.

[4] J. Kim and J. Ryou. *Multi-party fair exchange protocol using ring architecture model.* Japan-Korea Joint Workshop on Information Security and Cryptology, January 2000.

[5] S. Kremer and O. Markowitch. *A multi-party non-repudiation protocol.* Proceedings of SEC 2000: 15th International Conference on Information Security, IFIP World Computer Congress, pages 271-280, Beijing, China, August 2000.

[6] S. Kremer, O. Markowitch and J. Zhou. *An intensive survey of non-repudiation protocols.* Computer Communications, 2002.

[7] O. Markowitch and S. Kremer. *A multi-party optimistic non-repudiation protocol.* Proceedings of 3rd International Conference on Information Security and Cryptology (ICISC 2000), Lectures Notes in Computer Science vol. 2015, pages 109-122, Seoul, Korea, December 2000.

[8] O. Markowitch and S. Saeednia. *Optimistic fair-exchange with transparent signature recovery.* Proceedings of Financial Cryptography 2001, Lecture Notes in Computer Science, February 2001.

[9] J. Zhou and D. Gollmann. *A fair non-repudiation protocol.* Proceedings of the IEEE Symposium on Security and Privacy, pages 55-61, Oakland, California, May 1996.

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