# A Calculus for Non Repudiation Protocols

**Abdesselam Redouane**

**Department of Computer Science and Engineering, College of Engineering and Computing**
**Al Ghurair University, Dubai, UAE**

**Abstract**

We describe a calculus that is specific to non-repudiation protocols. The calculus uses the correspondence assertion of Woo and Lam, that is, if there is a non-repudiation of receipt there should be a corresponding non-repudiation of origin. The main contribution of this work lies in the way we model input and output and hence captures non-repudiation properties. The calculus is a subset of the Pi calculus. The basic constructs are modified in order to handle properties of non-repudiation. We offer a formal syntax and an operational semantics of the calculus. We show the usefulness of the calculus by describing Zhou optimistic protocol.

***Keywords:*** *Non repudiation protocols, Pi calculus, operational semantics.*

## 1. Introduction

One of the main concerns in e-business, in all its different forms such as B2B, B2C, E2C, is fair exchange of services. In simple terms this is concerned how to ensure fairness between parties. In that, there is no denial by one of the entities of having participated in all or part of an electronic transaction. For example, suppose that a business A instructs its bank to carry out some money transfer to a particular account. The bank executes the instruction requested by A. Later, A denies that he has sent a message for debiting money to that particular account. To avoid such denials the following non-repudiation services are required:

- Non-Repudiation of Origin (NRO) is intended to protect against the originator rejection or denial of having sent a message to the recipient.
- Non-Repudiation of Receipt (NRR) is intended to protect against the recipient rejection or denial of having received the message from the originator.

Non-repudiation protocols rely, usually, on a Trusted Third Party (TTP). All the parties involved in the transacting process trust the TTP. Any dispute will be resolved via this TTP. The trend in these protocols is that they try to minimise its use during a protocol run. Protocols, which do not respect this issue, however, will end up with a bottleneck problem. There are protocols, which eliminate the use of TTP altogether. The approach adopted in this latter case is a probabilistic one [1], [2]. An intensive survey of fair non repudiation protocols can be found in [3].

We describe a calculus which is specific to non-repudiation protocols. We are interested in the more general protocols and which involve the use of a TTP. The calculus is a sub set of the Pi calculus enriched with some primitives to handle non-repudiation properties. We use the technique of Woo and Lam [4] of the correspondence assertion in the sense that for every received NRR there must exists a corresponding NRO. The main contribution of this work lies in the way we model input and output and hence captures non-repudiation properties.

The sequel is organised as follows. In the next section we describe the calculus along with a brief introduction to the Pi calculus. In this section we provide the syntax and an operational semantics of the calculus. Section 3 illustrates the use of the calculus with an example. Related work is given in section 4 while section 5 concludes the paper.

## 2. The Calculus

The calculus is a subset of the Pi calculus [5] where we modified some of the primitive constructs to handle non-repudiation of origin and non-repudiation of receipt.

### 2.1. Pi Calculus Overview

The Pi calculus is in essence a process algebra where processes interact by sending data and channel names. The basic computational step is the transfer of a communication link between two processes. The following example illustrates this idea [6]. We have a client which wants to use the printer. The access to the printer is via the server. We have two channel of communication a and b. The channel a is used as an output channel from the server and the printer. The b channel can be used either direction

between the server and the client. Fig. 1 below shows this scenario.

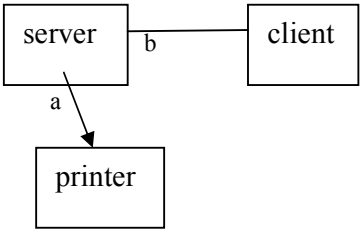



Fig.1: Before interaction between the server and client

There are three processes: S for the server, C for the client and P for the printer. There are two channels a and b as we mentioned earlier. This state can be written in the Pi calculus as follows:

$$\underline{b}a.S \mid b(m).\underline{m}d.C \qquad (1)$$

This expression state that the server will send the link a through the channel b and then behave like S. The client C is using the channel b as input where m is a place holder for the input received. The received input, which is the channel a, is then used as an output channel to send data d. The symbol | is used to mean parallel composition, that is, the two processes S and C are running in parallel and communicating via the channel b.

After the interaction between the processes S and C we have the following expression:

$$S \mid \underline{a}d.C \qquad (2)$$

That is, the channel a is being used by C to send its data to the printer.

Combining the two expressions (1) and (2) the interaction between the server and client can be formulated as follows:

$$\underline{b}a.S \mid b(m).\underline{m}d.C \longrightarrow S \mid \underline{a}d.C$$

Fig. 2 below shows now the new channel between the client and the printer.

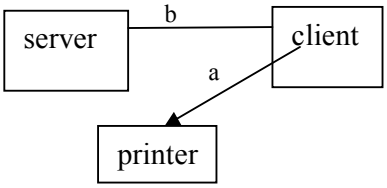



Fig.2: After interaction between the server and client

It should be noted that there are many variants of the Pi calculus which deal with specific area like the SPI calculus [7] and the Ambient calculus [8]. SPI calculus is used for modeling and analysing security protocols and the Ambient calculus is used to model and analyse mobile code.

The Pi calculus, as it is, cannot model non-repudiation of origin (NRO) and non-repudiation of receipt (NRR). It would be simpler; however, to modify the calculus to handle these specific issues elegantly rather than to model these with Pi core primitives and end up with what it might be a cumbersome description.

In order to make the calculus simple we use a biadic calculus rather than a polyadic one. We believe this will suffice to describe non-repudiation protocols, as it is usually the case that in this type of protocols there are two major elements of interest: the message and the non-repudiation service.

The framework in which the calculus should operate is that the evidence of non-repudiation, especially NRR, is generated by the protocol automatically rather than by the user. To this end, we use digital signatures to offer these services, as it is customary in these types of protocols. To accomplish this need, we suppose the availability, to a protocol, the followings: the participating agent identification and his private key. In addition, the generated signature makes use of the notion of a session of a protocol run. Thus, a digital signature is a tuple of the form: (typ, Id, K, $\alpha$).

Where:

Typ: is the type of the signature: {nro, nrr, sub, con}
Id: the identification of the agent
K: private key of the agent
$\alpha$: the session of a protocol run

We distinguish two types of digital signatures: the ones, which require non-repudiation of services, and the ones, which do not require such services.

The verification of a received digital signature represented by a NRO or NRR is assumed to be possible by each agent participating in a protocol. The type of communication between agents is asynchronous as we anticipate that an agent who performed a send/receive will not stay idle waiting for a response from the other agent.

## 2.2 Syntax

Let *N* be a set of names denoting communication actions and variables. Let τ be an internal action capable of being executed by any agent if he wishes to. Let *A* be a set of agent names, *T* a set of TTPs names and *D* a set of digital signatures. As stated earlier, two types of digital signatures are envisaged: those which require non-repudiation of services on one hand and those which they don't require on the other hand. We let the first type ranges over {nro, nrr} and the second type ranges over {con, sub}. The syntax is summarised in Table 1.

Table 1: Syntax

| | |
|---|---|
| a,b … x,y,z … | Names *N* |
| T, U, V | TTP agents *T* |
| no, nr, con, sub | digital signatures *D* |
| A, B ::= Agents *A* | |
| **0**(null) | |
| \|a(x,y).A (input) | |
| \|a(x,y).A (output) | |
| \| A \|\| B (parallel composition) | |
| \| rec(X).A (recursion) | |

An informal explanation of the different operators might be useful.

- **0** is the null agent that does nothing.
- a(x,y).A is the input on the channel a to be bind to x and y, and then behave like agent A. Note that y is place holder for a digital signature.
- a(x,y).A is the output that put x and y on the channel a and then behaves like A. Note that y may be a digital signature.
- A || B is the parallel composition.
- rec(X).A is the recursion to allow infinite call to the task accomplished by an agent. This expression binds free occurrences of X in A.

## 2.2 Operational Semantics

The operational semantics is explained below. Note that not all the symbols are there.

**τ** τ.A → A
**Inp$_{nro}$** a(x,y).A → A[m/x, nro/y] → a(0, nrr) →A
**Inp** a(x,y).A → A[m/x,0/y]
**Out$_{nro}$** a(m,nro).A → A → a(0,x).A
**Out$_{nrr}$** a(m,nrr).A → A
**Out** a(m).A → A

**PAR** $\dfrac{A \rightarrow A'}{A \,||\, B \rightarrow A' \,||\, B}$

**COM** $\dfrac{A \rightarrow A' \quad B \rightarrow B'}{A||B \rightarrow A'||B'}$

In the following we comment on these rules and how they should be interpreted.

**The Input Rule with an NRO (Inp$_{nro}$)**
This action is responsible for the guarantee of non repudiation of receipt and is actually formed in the following steps:

- Get action from the channel, which receive all the input in this case two parameters.
- The input parameters are substituted in their place holder, i.e. x and y respectively
- An output action is generated automatically on the same channel with the first parameter empty and the second parameter is the nrr of the recipient.
- The agent A will, then, continue performing his duties.

It should be noted that the rule is circular-free because once the originator receive the NRR he will not trigger another NRR for the recipient. Of course, the originator is able to see that the non-repudiation service received is a response of his earlier NRO.

It will be noticed from the definition of this rule is that we adopt a style of an early semantics where the substitutions occurs once they have been received and then the process evolves to another state contrary to a late semantics one.

**The Input Rule without an NRO (Inp)**
This rule is needed if non-repudiation is a not a must. This case may be of interest in a normal

communication between agents or where the NRO and NRR are not required.

In this action the second parameter is empty. Once the recipient detect that the second parameter is empty there is no need to continue, but rather it is obligatory to stop, with his non-repudiation activities.

**The Output Rule with NRO ($Out_{nro}$)**
The rule is for initiating a non-repudiation handshake. The originator starts by forming his nro and sent it to the recipient. As the rule suggests the originator has to wait on the same channel to get his nrr. It should be noted that this channel will not be used for other communication activities while it is in this status.

**The Output Rule with NRR ($Out_{nrr}$)**
The rule is for responding to a received NRO. It should be noted that this rule is triggered automatically after an input has been made which contains an NRO

**The Output Rule without NRO or NRR (Out)**
This rule allows an agent to perform regular communication with another agent where there is no need for non-repudiation services. It is the symmetric counterpart of the Inp rule without non-repudiation services.

**The parallel Rule (PAR)**
This rule defines the behaviour of the parallel action and it is self-explanatory.

**The Communication Rule (COM)**
This is the main communication between agents running in parallel and willing to communicate on a common channel. Note that the agents can communicate using non-repudiation services or without them, that is, in a regular communication.

### 2.4 Bisimulation

In this subsection we define a bisimulation method between processes. The purpose is to be able to make judgment whether two processes are equivalent. This result will be useful, for instance, to verify that an implementation meets its specification.

Two agents A and B are bisimular is that for each transition from A to be matched by a transition from B and vice-versa, leading again to equivalent derivatives A' and B'.

As it has been stated earlier in the calculus rules that we adopted an early semantics, therefore, in the definition of the bisimulation we use an early bisimulation style. A binary symmetric process relation S is bisimulation if (A,B) e S implies:

(i) if A → A' with an input action a(x,y) then for all (z,p) ∃ B': B → B' with the input action a(x,y) and (A'[z/x],B'[p/y]) ε S

(ii) (ii) if A → A' with an action different from an input then ∃ B': B → B' and (A',B') ε S

A and B are bisimular written A ~ B if (A,B) e S for some bisimulation S.

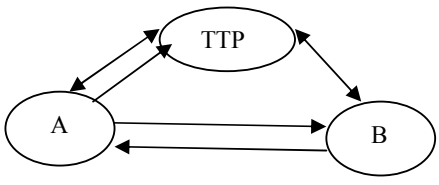
## 3. Example

In order to illustrate the calculus in practice we specify the optimistic protocol of Zhou [9] (Zhou, 1996). We follow the usual routine in this type of formalism. That is, we provide a specification of the protocol and an implementation. All this encoding is in the calculus. The final step is to proof that the implementation and the specification are bisimular. If this case holds we conclude that the protocol indeed guarantees non-repudiation properties.

### 3.1 Protocol Description

The main idea of the protocol is to minimise the use of the TTP. For that the originator starts by making a commitment to the recipient by sending the message encrypted. Note, however, that the key is not sent with the message. The originator, then, lodges the key with the TTP. Part of the non-repudiation is that the recipient must retrieve the key from the TTP and the originator, as well, has to get a confirmation from the TTP about the key. Hence the originator must retrieve this confirmation from the TTP.

### 3.2 Protocol Diagram and its Standard Notation




The protocol in standard notation is as follows:
Message 1. A → B: f_EOO,B,L,C,EOO
Message 2. B → A: f_EOR,A,L,EOR

Message 3. A➔TTP: f_SUB,B,L,K,SUB_K
Message 4. B ⇔ TTP: f_CON,A,B,L,K,CON_K
Message 5. A ⇔ TTP: f_CON,A,B,L,K,CON_K

### 3.3 Protocol Encoding

We map each send and receive between two agents as one process. That is, between A and B and between any agent (A, B) and the TTP. We have four processes in total, which should be performed in sequential order.

ZhouProtocImp = A1.A2.A3.A4

Where:

A1 = a(m,no).A || a(x,y).B
A2 = a(m,sub).A || a(x,y).T
A3 = a(m,con).T || a(x,y).B
A4 = a(m,con).T || a(x,y).A

On the other hand we need a specification for the protocol which we leave as a future work. The final task then is to show that ZhouProtocImpl is bisimilar to ZhouProtocSpec.

## 4. Related Work

It should be noted that Schneider [10] has used CSP for the analysis of the above protocol where the proof has been made by hand.

Kremer [11] verified non-repudiation, with a TTP, using a game based model that uses the model of alternating transition systems (ATS) and alternating time temporal logic (ATL) [12].

Zhou work on non repudiation also uses a TTP in his protocols and uses belief logic SVO [13] to verify non-repudiation protocols.

Formal analyses have been also used by Shmatikov [14] and [15] to study fair exchange protocols.

Zhang [16] uses labelled colored Petri nets to model and analyse non repudiation services in a distributed system.

## 5. Conclusion

We have described a calculus that is useful in the description of non-repudiation protocols. Its syntax and operational semantics have been described.

As a future work, we intend to complete the verification of Zhou optimistic protocol stated in the example. Another area of investigation is an implementation of this calculus in order to take it from a paper and a pencil work to machine automation. To this end, a tool will be useful, in that, given a protocol description, will decide if the protocol is satisfying non-repudiation properties or not.

## References

[1] O. Markowitch, Y. Roggeman, “Probabilistic non repudiation without trusted third party”, in *Second* Conference on Security in Communication Networks, 1999.
[2] A. Aldini, R. Gorrieri, “Security analysis of a probabilistic non repudiation protocol”, in PAPM-PROMIV, LCNS 2399, 2002 Springer Verlag.
[3] S. Kremer, O. Markowitch,J. Zhou, “An Intensive Survey of Fair Non-Repudiation Protocols”, in Elsevier Science, 2002.
[4] T. Woo, S. Lam, “A semantic model for authentication protocols”, in IEEE Symposium on Security and Privacy, 1993.
[5] R. Milner, Communicating and Mobile Systems: the π-Calculus, Cambridge University Press, 1999.
[6] Joachim Pi: An introduction to the calculus. pp: 479 - 543 Handbook of Process Algebra, 2001, Elsvier Science, 2001.
**[7]** M. Abadi, A. D. Gordon: A Calculus for Cryptographic Protocols: The spi Calculus. Information and Computation, Vol. 148, Issue 1, pp: 1-70, 1999.
[8] L. Cardelli, A.D. Gordon. "Mobile Ambients". in proceedings of the First international Conference on Foundations of Software Science and Computation Structure, Lecture Notes in Computer Science (Springer-Verlag), 1378, pp: 140–155, 1998.
[9] J. Zhou, D. Gollmann, “A fair non repudiation Protocol”, in IEEE Computer Society Symposium on Research in Security and Privacy, 1996.
[10] S. Schneider, “Formal Analysis of a Non-Repudiation Protocol”, in Proceeding of the 11th IEEE Computer Security Foundations Workshop, 1998.
[11] S. Kremer, J. Raskin, “A game-based verification of non-repudiation and fair exchange protocols”, Journal of Computer Security, 2003.
[12] R. Alur, T. Henzinger, O. Kupferman, “Alternating time temporal logic”, in Proceeding of the 38th Annual Symposium on Foundation of Computer Science, IEEE Computer Societry Press, 1997.
[13] J. Zhou, D. Gollmann, D., “Towards verification of non repudiation protocols”, in Proceeding of International Refinement Workshop and Formal Methods, Spring Verlag, 1998.
[14] V. Shmatikov, J. Mitchell, “Analysis of abuse-free contract signing”, in Financial Cryptography, LCNS 1962, Spring Verlag, 2000.

[15] V. Shmatikov, J. Mitchell, “Finite state of two contract signing protocols”, Theoretical Computer Science, 2002.
[16] H. Zheng, Y. Yue Du, S. Yu, ”Modeling non repudiation in distributed systems”, Information Technology Journal, 2008.

Dr. Abdesselam Redouane is currently with the college of engineering and computing, Al Ghurair University, Dubai. He received his PhD in Computer Science from Manchester University, UK. His research interest lies in the area of the application of software engineering techniques to new emerging technologies like web and mobile applications. He is also interested in computer security and especially in access control. He is an associate editor of the International Engineering Letter.