Study for Automatically Analysing Non-repudiation

Judson Santiago and Laurent Vigneron[*]

LORIA – INRIA-UN2 (UMR 7503)
BP 239, 54506 Vandœuvre-lès-Nancy Cedex, France
{judson,vigneron}@loria.fr

Abstract. While security issues such as secrecy and authentication have been studied intensively, most interest in non-repudiation protocols has only come in recent years. Non-repudiation services must ensure that when two parties exchange informations over a network, neither one nor the other can deny having participated in this communication. Consequently a non-repudiation protocol has to generate evidences of participation to be used in the case of a dispute.

In this paper, we present a description of non-repudiation services, and illustrate them on the Fair Zhou-Gollmann protocol. Then we show how to define non-repudiation properties with the AVISPA tool and explain how they can be automatically verified.

1   Introduction

While security issues such as secrecy and authentication have been studied intensively, most interest in non-repudiation protocols has only come in recent years, notably in the yearly 1990s with the explosion of Internet services and electronic transactions. Non-repudiation services must ensure that when two parties exchange informations over a network, neither one nor the other can deny having participated in this communication. Consequently a non-repudiation protocol has to generate evidences of participation to be used in the case of a dispute. With the advent of digital signatures and public key cryptography, the base for non-repudiation services was created. Given an adequate public key infrastructure, one having a signed message has an irrefutable evidence of the participation and the identity of his party.

In the cases where the evidence is provided to both parties, the protocol might also aim to provide fairness, i.e no party should be able to reach a point where they have the evidence or the message they require without the other party also having their required evidence. Fairness is not required for non-repudiation, but it is usually desirable.

A typical non-repudiation protocol can provide a number of different non-repudiation services, like non-repudiation of origin, non-repudiation of receipt and fairness, but the actual non-repudiation services provided by a protocol depend mainly on its application. For example, the publisher of an on-line magazine may want to keep track of the users that are downloading the latest issue, so non-repudiation of receipt would be required; that means the user could not deny having received the latest issue because the publisher has an evidence of the download. In that case non-repudiation of origin or fairness is not applicable. Contrariwise an electronic transactions site would require both non-repudiation of origin, receipt and also fairness.

The first solutions providing fairness in exchange protocols were based on a gradual exchange of the expected information [5]. However this simultaneous secret exchange is troublesome for actual implementation because fairness is based on the assumption of equal computational power on both parties, which is very unlikely in a real world scenario. Therefore the solution we will

---

focus here is the adoption of a *trusted third party* (TTP). The TTP can be used as a delivery agent to provide simultaneous share of evidences.

From the existing applications, we can distinguish the following non-repudiation services:

Non-repudiation of origin (NRO) provides the recipient with the evidence NRO which ensures that the originator will not be able to deny having sent the message. The evidence of origin is generated by the originator and held by the recipient.

Non-repudiation of receipt (NRR) provides the originator with the evidence NRR which ensures that the recipient will not be able to deny having received the message. The evidence of receipt is generated by the recipient and held by the originator.

Non-repudiation of submission (NRS) is intended to provide evidence that the originator submitted the message for delivery. This service only applies when the protocol uses a TTP. Evidence of submission is generated by the delivery agent, and will be held by the originator.

Non-repudiation of delivery (NRD) is intended to provide evidence that the recipient received the message. This service also only applies when the protocol uses a TTP. Evidence of delivery is generated by the delivery agent, and will be held by the originator.

Fairness is achieved for a non-repudiation protocol if at the end of the protocol execution either the originator has the evidence of receipt for the message m and the recipient has the evidence of origin of the corresponding message m, or none of them has any valuable information.

The evidences NRO, NRR, NRS and NRD are messages signed by an agent.

In the following section we will describe the Fair Zhou-Gollmann protocol (FairZG), a fair non-repudiation protocol that uses a TTP. We have chosen the FairZG as a case study to demonstrate our analysis approach of the non-repudiation. One of the motivations for our choice is the existence of significant related work [2, 4, 9, 13].

The majority of the non-repudiation property analysis efforts in the literature are manually driven analysis though. The first effort to apply formal methods to the verification of non-repudiation protocols have been presented by Zhou et al. in [13], where they used SVO logic. Later in [9] Schneider used process algebra CSP to prove the correctness of a non-repudiation protocol and more recently Bella et al. have used the theorem prover Isabelle [2]. The only known automatic analysis attempts were done by Shmatikov and Mitchell in [10] where they use Murφ, a finite state model-checker, to analyse a fair exchange and two contract signing protocols, and by Kremer and Raskin in [6] that have used a game based model.

This case study will show more precisely what are the non-repudiation evidences for the FairZG. Moreover we will show how the non-repudiation goals of the protocol can be modelled in HLPSL [3], the High Level Protocol Specification Language of the AVISPA tool [1], and how non-repudiation properties can be expressed with authentication properties. We have specified the entire protocol in HLPSL and automatically analysed it using the AVISPA tool.

2   A Fair Non-repudiation Protocol

The protocol is presented below in Alice&Bob notation, where fNRO, fNRR, fSUB and fCON are labels used to identify the purpose of the messages (a graphical representation is given in Figure 1).

1. A → B: fNRO.B.L.C.NRO
2. B → A: fNRR.A.L.NRR
3. A → TTP: fSUB.B.L.K.subK
4. B ↔ TTP: fCON.A.B.L.K.conK
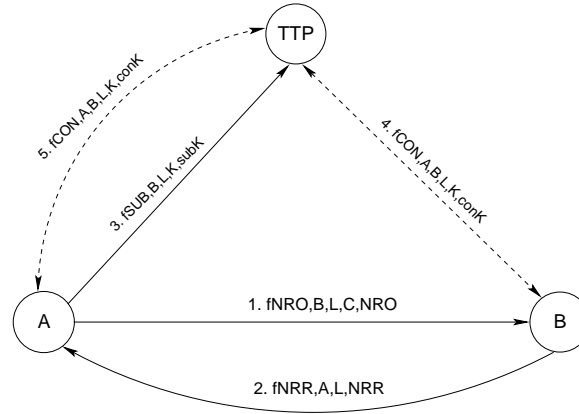5. A ↔ TTP: fCON.A.B.L.K.conK

Fig. 1. Fair Zhou Gollmann Non-repudiation Protocol

where

A : Alice, the originator of the message M
B : Bob, the recipient of the message M
TTP : trusted third party
M : message sent from A to B
C : commitment, the message M encrypted under key K
L : a session identifier, a nonce
K : symmetric key defined by A
NRO : non-repudiation of origin, the message fNRO.B.L.C signed by A
NRR : non-repudiation of receipt, the message fNRR.A.L.C signed by B
subK : proof of submission of K, the message fSUB.B.L.K signed by TTP
conK : confirmation of K, the message fCON.A.B.L.K signed by TTP

The main idea of the FairZG protocol is to split the delivery of a message into two parts. First a commitment C, containing the message M encrypted by a key K, is exchanged between Alice and Bob. Once Alice has an evidence of commitment from Bob, the key K is sent to a trusted third party. Once the TTP has received the key, both Alice and Bob can retrieve the evidence conK and the key K from the TTP. This last step is represented by a double direction arrow in the Alice&Bob-style notation because it is implementation specific and may be composed by several message exchanges between the agents and the TTP. In this scenario we assume the network will not be down forever and both Alice and Bob have access to the TTP's shared repository where it stores the evidences and the key. That means the agents will eventually be able to retrieve the key and evidences from the TTP even in case of network failures.

3 Non-repudiation Goals Based on Authentication

The current version of the HLPSL protocol specification language and the AVISPA tool can deal only with secrecy and authentication properties using a Dolev-Yao intruder model. The threat scenario for non-repudiation protocols usually comes into the form of a honest agent playing with a dishonest agent, participants require protection from each other, rather than from an external hostile agent. But a dishonest agent model can be fairly approximated by an intruder

model where the intruder can impersonate an agent. And although the threat model is slightly different, it is well know that non-repudiation is a form of authentication [8].

This is why our first attempt to support non-repudiation properties in the HLPSL level is to map the non-repudiation property into authentication problems. In the sequence we show how that can be done for the most important non-repudiation properties.

3.1   Non-repudiation of Origin

The non-repudiation of origin property requires a guarantee that Alice sent some particular message. In the FairZG protocol, non-repudiation of origin should provide the guarantee that Alice sent a message to Bob containing the label L and the cipher text C = {M}$_K$, and another message containing the same label L and the key K. If these two messages are signed by Alice and successfully received by Bob, they provide evidence that Alice sent M to Bob.

In other words, Bob needs to authenticate two messages sent by Alice, the messages fNRO and fSUB (see Figure 1). But since fSUB goes to the TTP, the authentication will occur between the TTP and Bob on the message fCON, that also contains messages L and K. The authentications will assure that the messages received are indeed the ones Alice sent and not one coming from an intruder.

Definition 1. *Suppose Alice sends the message M to Bob in the FairZG protocol, the goal of non-repudiation of origin can be expressed as an authentication problem in the following way:*

Non-Repudiation of Origin(M) =
*   Bob authenticates Alice on NRO*
*   ∧ Bob authenticates TTP on ConK*

From the above definition an external judge, willing to solve a dispute, can conclude:

- Bob authenticates Alice on NRO ⇒
  the label L and the commitment C came from Alice;
- Bob authenticates TTP on ConK ⇒
  the key received by Bob was the one stored in the TTP.

We can write these authentications in a lower level as a predicate of witness and requests. In HLPSL, witness(A,B,x,Y) means agent A delivers a witness to agent B that he knows message Y; wrequest(B,A,x,Y) means agent B expects the existence of a witness delivered previously by agent A over message Y. The witness and wrequest predicates have to be specified in the transitions of the protocol. So they are automatically fired when the transition is applied.

Note that weak authentication is used, as opposed to strong authentication that includes replay protection (see Section 4.2).

In the following definition we show how the authentications are expressed in this lower level and how they relate to a simple belief logic.

Definition 2. *Given a trace of the protocol execution tr,*

Non-Repudiation of Origin(tr) =
*   (witness (A,B,nro,{fNRO.B.L.C}$_{inv(K_A)}$) ⊂ tr*
*   ∧ wrequest (B,A,nro,{fNRO.B.L.C}$_{inv(K_A)}$) ⊂ tr*
*   ∧ witness (TTP,B,con,{fCON.A.B.L.K}$_{inv(K_{TTP})}$) ⊂ tr*
*   ∧ wrequest (B,TTP,con,{fCON.A.B.L.K}$_{inv(K_{TTP})}$) ⊂ tr)*
*   ⇒ (A sent {fNRO.B.L.C}$_{inv(K_A)}$*
*   ∧ B received {fNRO.B.L.C}$_{inv(K_A)}$*
*   ∧ TTP sent {fCON.A.B.L.K}$_{inv(K_{TTP})}$*
*   ∧ B received {fCON.A.B.L.K}$_{inv(K_{TTP})}$)*

3.2   Non-repudiation of Receipt

The non-repudiation of receipt property requires a guarantee that Bob received some particular message. In the FairZG protocol, non-repudiation of receipt should provide the guarantee that Bob received the label L and the cipher text $C = \{M\}_K$, and another message containing the same label L and the key K. The first message is the commitment NRR from Bob and the second is the receipt ConK from the TTP. These two messages provide evidence that B received M.

This means Alice needs to authenticate Bob on NRR and the TTP on ConK. Once Alice possesses the authenticated messages sent by Bob and TTP, she has an evidence that Bob has received M in an encrypted form and is able to get it in clear.

Definition 3. *Suppose Alice sends the message M to Bob in the FairZG protocol, the goal of non-repudiation of receipt can be expressed as an authentication problem in the following way:*

Non-Repudiation of Receipt(M) =
   Alice authenticates Bob on NRR
   $\wedge$ Alice authenticates TTP on ConK

From the above definition an external judge can conclude:

- Alice authenticates Bob on NRR $\Rightarrow$
  the label L and the commitment C were received by Bob;
- Alice authenticates TTP on ConK $\Rightarrow$
  Bob received or has access to the key K in TTP's repository.

Even if Bob did not take the key from TTP's repository, the non-repudiation of receipt is assured due to our assumption of the agents eventually having access to the repository even in case of network failures. The key is available, it is up to Bob to retrieve it or not. The same applies to Alice.

We can write these authentications in a lower level as a predicate of witness and requests as shown in the following definition.

Definition 4. *Given a trace of the protocol execution tr,*

Non-Repudiation of Receipt(tr) =
   (witness $(B,A,nrr,\{fNRR.A.L.C\}_{inv(K_B)}) \subset tr$
   $\wedge$ wrequest $(A,B,nrr,\{fNRR.A.L.C\}_{inv(K_B)}) \subset tr$
   $\wedge$ witness $(TTP,A,con,\{fCON.A.B.L.K\}_{inv(K_{TTP})}) \subset tr$
   $\wedge$ wrequest $(A,TTP,con,\{fCON.A.B.L.K\}_{inv(K_{TTP})}) \subset tr)$
      $\Rightarrow$ (B sent $\{fNRR.A.L.C\}_{inv(K_B)}$
        $\wedge$ A received $\{fNRR.A.L.C\}_{inv(K_B)}$
        $\wedge$ TTP sent $\{fCON.A.B.L.K\}_{inv(K_{TTP})}$
        $\wedge$ A received $\{fCON.A.B.L.K\}_{inv(K_{TTP})}$)

3.3   Fairness

In the literature authors often give different definitions of fairness for non-repudiation protocols. In some definitions none of the parties should have more evidences than the others at any given point in time. Others have a more flexible definition in which none of them should have more evidences than the others in the end of the protocol run. In many works it is also not made very clear if only successful protocol runs are taken into account or partial protocol runs are valid as well.

In this paper the later definition of fairness will be used and we take into account complete protocol runs. By complete protocol runs we mean a run where, even though the protocol could not have reached it's last transition for all the agents, there is no executable transition left, i.e. all possible protocol steps were executed, but that does not mean all agents are in their final states.

We can define fairness as a function of non-repudiation of origin and of non-repudiation or receipt. If both properties, NRO and NRR, are ensured or both are flawed for a given message M, then we have fairness. We can also write this definition as follows:

Definition 5. *Given a trace of the protocol execution* tr,

Fairness(tr) =
       Non-Repudiation of Origin(tr) $\oplus$ *Non-Repudiation of Receipt(tr)*

4   Specification and Analysis of a Non-repudiation Protocol

In this section we present the FairZG protocol specified in HLPSL, using authentication to model the non-repudiation properties. The key retrieval part of the protocol was modelled by system states without events: in the second protocol rule, Alice sends the key K to the TPP and from this point on she can ask for her evidence ConK.

```
role alice ( A,B,S: agent,
             Ka,Kb,Ks: public_key,
             Hash: function,
             Snd, Rcv: channel(dy) ) played_by A def=

  local State: nat,
        M: text,
        K: symmetric_key,
        L,C,NRO,NRR,SubK,ConK: message

  init State:=0

  transition
   1. State=0 /\ Rcv(start)
      =|>
      State':=1 /\
      M':=new() /\ K':=new() /\
      C':={M'}_K' /\ L':=Hash(M',K') /\
      NRO':={fNRO.B.L'.C'}_inv(Ka) /\
      Snd(fNRO.B.L'.C'.NRO') /\
      % Non-repudiation of Origin
      witness(A,B,bob_alice_nro,NRO')

   2. State=1 /\ Rcv(fNRR.A.L.NRR') /\
      NRR'={fNRR.A.L.C}_inv(Kb)
      =|>
      State':=2 /\ SubK':={fSUB.B.L.K}_inv(Ka) /\
      Snd(fSUB.B.L.K.SubK')

   3. State=2
      --|>
      State':=3 /\ Snd(fREQ.A.B.L)
```

```
   4. State=3 /\ Rcv(fCON.A.B.L.K.ConK') /\
      ConK'={fCON.A.B.L.K}_inv(Ks)
      =|>
      State':=4 /\
      % Non-repudiation of Receipt
      wrequest(A,B,alice_bob_nrr,NRR) /\
      % Non-repudiation of Delivery
      wrequest(A,S,alice_ttp_con,ConK')
end role
```

The role Bob works in a similar way to Alice, he can request ConK to the TTP as soon as he exchanged the commitment C with Alice. The TTP though, will only be able to answer when he receives the key K from Alice.

```
role bob ( B,A,S: agent,
           Kb,Ka,Ks: public_key,
           Snd,Rcv: channel(dy) ) played_by B def=

   local State: nat,
         M: text,
         K: symmetric_key,
         L,C,NRO,NRR,ConK: message

   init State:=0

   transition
    1. State=0 /\ Rcv(fNRO.B.L'.C'.NRO') /\
       NRO'={fNRO.B.L'.C'}_inv(Ka)
       =|>
       State':=1 /\
       NRR':={fNRR.A.L'.C'}_inv(Kb) /\
       Snd(fNRR.A.L'.NRR') /\
       % Non-repudiation of Receipt
       witness(B,A,alice_bob_nrr,NRR')

    2. State=1
       --|>
       State':=2 /\ Snd(fREQ.A.B.L)

    3. State=2 /\ Rcv(fCON.A.B.L.K'.ConK') /\
       ConK'={fCON.A.B.L.K'}_inv(Ks) /\
       C={M'}_K'
       =|>
       State':=3 /\
       % Non-repudiation of Origin
       wrequest(B,A,bob_alice_nro,NRO) /\
       % Non-repudiation of Delivery
       wrequest(B,S,bob_ttp_con,ConK')
end role
```

The role TTP has a small participation in the protocol, although very important. It receives the key K from Alice and sends the evidence ConK to both parties upon request.

```
role ttp ( S,A,B: agent,
```

```
          Ks,Ka: public_key,
          Snd,Rcv: channel(dy) ) played_by S def=

  local State: nat,
        K: symmetric_key,
        L,SubK,ConK : message

  init State:=0

  transition
   1. State=0 /\ Rcv(fSUB.B.L'.K'.SubK') /\
      SubK'={fSUB.B.L'.K'}_inv(Ka)
      =|>
      State':=1

   2. State=1 /\ Rcv(fREQ.A.B.L)
      =|>
      ConK':={fCON.A.B.L.K}_inv(Ks) /\
      Snd(fCON.A.B.L.K.ConK') /\
      % Non-repudiation of Delivery
      witness(S,A,alice_ttp_con,ConK') /\
      witness(S,B,bob_ttp_con,ConK')
end role
```

For studying the protocol we have to define what is a session: it corresponds to a parallel execution of roles Alice, Bob and TTP.

```
role session ( A,B,S: agent,
               Ka,Kb,Ks: public_key,
               H: function,
               Snd,Rcv: channel(dy) ) def=

  composition
       alice(A,B,S,Ka,Kb,Ks,H,Snd,Rcv)
    /\ bob(B,A,S,Kb,Ka,Ks,Snd,Rcv)
    /\ ttp(S,A,B,Ks,Ka,Snd,Rcv)
end role
```

The HLPSL specification is completed with the environment role, containing the session instances to be analysed (one normal session in the example below).

```
role environment () def=

  local Snd,Rcv: channel(dy)

  const a,b,s,i: agent,
        ka,kb,ks,ki: public_key,
        h: function,
        bob_alice_nro, alice_bob_nrr,
            alice_ttp_con, bob_ttp_con,
            fNRO,fNRR,fSUB,fREQ,fCON: protocol_id

  intruder_knowledge = {i,a,b,s,ka,kb,ks,ki,inv(ki),fNRO,fNRR,fSUB,fREQ,fCON}

  composition
```

```
        session(a,b,s,ka,kb,ks,h,Snd,Rcv)
end role
```

Finally, the authentication properties to be checked are listed in the goal section:

```
goal
  weak_authentication_on bob_alice_nro
  weak_authentication_on alice_bob_nrr
  weak_authentication_on alice_ttp_con
  weak_authentication_on bob_ttp_con
end goal
```

4.1   Automatic Analysis of the Protocol

The analysis with the AVISPA tool is performed on 3 parallel sessions of the protocol: one normal session between agents **a** and **b**, one session between **a** and the intruder (**i**), and one session between the intruder and **b**. The contents of the environment role is:

```
  composition
      session(a,b,s,ka,kb,ks,h,Snd,Rcv)
   /\ session(a,i,s,ka,ki,ks,h,Snd,Rcv)
   /\ session(i,b,s,ki,kb,ks,h,Snd,Rcv)
```

The intruder follows the behavior of the standard Dolev-Yao model: it acts as an external intruder for the first session, and as a dishonnest participant for the last two sessions.

The analysis of this specification resulted in no attacks. But the consideration of different scenarios brought to surface interesting remarks, detailed in the next two sessions.

4.2   No Replay Attack Protection

Normally replay attacks should not be meaningful to non-repudiation protocols when the repeated messages come from a successful run of the protocol. That is because both parties will have already their non-repudiation evidences for that particular session. So if the intruder is able to give advantage to an agent in a subsequent session by replaying old messages, the other agent already has the evidences for that messages and, consequently, the replay attack does not mean a non-repudiation attack.

Specifying the non-repudiation properties of the FairZG protocol by using strong authentication leads to those kinds of replay attacks where the replay is made from a successful run of the protocol. So even though those attacks are not meaningful ones, it is important to point out that *the FairZG protocol does not provide replay protection*.

The analysis is performed by transforming the HLPSL specification:

- all wrequest predicates are replaced by request predicates;
- the environment role contains two identical instances of the same session between **a** and **b**;
- in the goal section, authentication_on replaces weak_authentication_on.

The attack trace obtained when using strong authentication to model the non-repudiation properties is showed below.

```
i -> a(1):  start
a(1) -> i:  fNRO.b.h(m1.k1).{m1}_k1.
            {fNRO.b.h(m1.k1).{m1}_k1}_(inv(ka))
            & witness(a,b,bob_alice_nro,{fNRO.b.h(m1.k1).{m1}_k1}_(inv(ka)))
```

```
i -> b(1):  fNRO.b.h(m1.k1).{m1}_k1.
            {fNRO.b.h(m1.k1).{m1}_k1}_(inv(ka))
b(1) -> i:  fNRR.a.h(m1.k1).
            {fNRR.a.h(m1.k1).{m1}_k1}_(inv(kb))

b(1) -> i:  fREQ.a.b.h(m1.k1)

i -> b(2):  fNRO.b.h(m1.k1).{m1}_k1.
            {fNRO.b.h(m1.k1).{m1}_k1}_(inv(ka))
b(2) -> i:  fNRR.a.h(m1.k1).
            {fNRR.a.h(m1.k1).{m1}_k1}_(inv(kb))

b(2) -> i:  fREQ.a.b.h(m1.k1)

i -> a(1):  fNRR.a.h(m1.k1).
            {fNRR.a.h(m1.k1).{m1}_k1}_(inv(kb))
a(1) -> i:  fSUB.b.h(m1.k1).k1.
            {fSUB.b.h(m1.k1).k1}_(inv(ka))

a(1) -> i:  fREQ.a.b.h(m1.k1)

i -> s(1):  fSUB.b.h(m1.k1).k1.
            {fSUB.b.h(m1.k1).k1}_(inv(ka))

i -> s(1):  fREQ.a.b.h(m1.k1)
s(1) -> i:  fCON.a.b.h(m1.k1).k1.
            {fCON.a.b.h(m1.k1).k1}_(inv(ks))

i -> b(1):  fCON.a.b.h(m1.k1).k1.
            {fCON.a.b.h(m1.k1).k1}_(inv(ks))
            & request(b,a,bob_alice_nro,{fNRO.b.h(m1.k1).{m1}_k1}_(inv(ka)))

i -> b(2):  fCON.a.b.h(m1.k1).k1.
            {fCON.a.b.h(m1.k1).k1}_(inv(ks))
            & request(b,a,bob_alice_nro,{fNRO.b.h(m1.k1).{m1}_k1}_(inv(ka)))
```

The first session between a(1), b(1) and s(1) is entirely done: a witness for NRO is generated by a(1), and the corresponding request is generated by b(1). In the second session, the intruder plays the role of a(2) with b(2) using the NRO message of the first session; then b(2) generates a second request for this NRO, raising the detection of the replay attack.

4.3   Non-repudiation is Based on Signatures

When using weak authentication to model the non-repudiation properties, the analysis reports no attacks. However often, non-repudiation protocols can be attacked when a honest agent tries to team up with the intruder. Usually this team up involves the disclosure of important information to the intruder by the honest agent. To simulate this scenario we modified the protocol specification to add the private key of one agent to the intruder's knowledge. When the private key of a, inv(ka), is given to the intruder, the attack found is described below:

```
i -> b:  fNRO.b.L.{M}_K.{fNRO.b.L.{M}_K}_(inv(ka))
b -> i:  fNRR.a.L.{fNRR.a.L.{M}_K}_(inv(kb))
```

```
b -> i:  fREQ.a.b.L
i -> s:  fSUB.b.L.K.{fSUB.b.L.K}_(inv(ka))

i -> s:  fREQ.a.b.L
s -> i:  fCON.a.b.L.K.{fCON.a.b.L.K}_(inv(ks))

i -> b:  fCON.a.b.L.K.{fCON.a.b.L.K}_(inv(ks))
         & wrequest(b,a,bob_alice_nro,{fNRO.b.L.{M}_K}_(inv(ka)))
```

This trace shows that the intruder can impersonate `a` in the role of Alice, and play the entire protocol without `b` (role Bob) to notice it. The attack found is a non-repudiation of origin and a fairness attack. At the end of the protocol run, Bob has a message and all the evidences that Alice sent that message, but Alice has not played the protocol. The attack shows how important is the agent signatures to the security of the FairZG protocol; if an agent signature is disclosed, the intruder can do whatever he wants in the protocol.

But the attack, which is similar if Bob, instead of Alice, discloses his private key to the intruder, does not benefit the dishonest agent that is trying to team up with the intruder. The only prejudice is for themselves. Since it is very unlikely that an agent will do something harmful for himself, the attack is not very important for the security of the protocol.

5   Results and Conclusion

The Dolev-Yao intruder model implemented in the AVISPA tool has been used to automatically verify the properties of non-repudiation of origin, non-repudiation of receipt and fairness using authentication to model the protocol goals.

Although the non-repudiation goals can be based on authentication problems, this solution does not fully suit the current implementation of the intruder model in the AVISPA tool, as explained next.

In a protocol run we can distinguish two scenarii:

1. The intruder acts as an external agent: when the intruder plays the protocol like an external agent, he will not interfere with the firing of the witness and request predicates by the honest agents. So if we have a request without its corresponding witness in a trace of a given protocol run, that means the intruder found a way to execute the protocol in which a message M was sent but the message either did not arrive to its destination or the message was changed by the intruder on its way to the recipient.
2. The intruder impersonates an agent: if the intruder takes the identity of a honest agent of the protocol, this agent will act as a dishonest agent. In such situation the current implementation of the AVISPA tool will throw away the protocol transitions of the impersonated agent and the knowledge of this agent will be added to the intruder's knowledge. For authentication and secrecy properties this behaviour is alright, but since the intruder will not fire the transitions that would generate the witness and request predicates of the impersonated agents, we cannot verify non-repudiation.

Because of the above restriction, specifying a session like `session(a,i,s,ka,ki,ks,h,Snd,Rcv)`, where the intruder explicitly plays the role of a honest agent, will not permit to consider all the possible scenarii. Since some attacks may be missed with such restriction, representing non-repudiation by authentication problems is not enough and a new method, representing non-repudiation goals by agents knowledge is under study.

Some restrictions apply to the current results: all protocols traces are traces where the protocol finishes. It is not possible to simulate an unresponsive agent using this model and then we

can not find attacks in which an agent does not respond to a message when he is supposed to, like the attack described by [4].

The main result presented in this paper is the completely automatic analysis of non-repudiation properties, and the modelisation of the properties as authentication problems.

Future work includes the representation of non-repudiation as agents knowledge and a bigger case study of several non-repudiation protocols to assess the efficiency of the method and of the AVISPA tool over non-repudiation protocols.

References

1. A. Armando, D. Basin, Y. Boichut, Y. Chevalier, L. Compagna, J. Cuellar, P. Hankes Drielsma, P.-C. Héam, J. Mantovani, S. Mödersheim, D. von Oheimb, M. Rusinowitch, J. Santos Santiago, M. Turuani, L. Viganò, and L. Vigneron. The AVISPA Tool for the automated validation of internet security protocols and applications. In K. Etessami and S. Rajamani, editors, *17th International Conference on Computer Aided Verification, CAV'2005*, volume 3576 of *Lecture Notes in Computer Science*, Edinburgh, Scotland, 2005. Springer. `http://www.avispa-project.org/`.
2. G. Bella and L. C. Paulson. Mechanical Proofs about a Non-repudiation Protocol. In *14th Int. Conference Theorem Proving in Higher Order Logics, TPHOLs*, volume 2152 of *Lecture Notes in Computer Science*, pages 91–104, Edinburgh, Scotland, 2001. Springer.
3. Y. Chevalier, L. Compagna, J. Cuellar, P. Hankes Drielsma, J. Mantovani, S. Mödersheim, and L. Vigneron. A High Level Protocol Specification Language for Industrial Security-Sensitive Protocols. In *Automated Software Engineering, Proc. of the Workshop on Specification and Automated Processing of Security Requirements, SAPS*, pages 193–205, Linz, Austria, September 2004. Austrian Computer Society.
4. S. Gürgens, C. Rudolph, and H. Vogt. On the Security of Fair Non-repudiation Protocols. In *6th Int. Conference on Information Security, ISC*, volume 2851 of *Lecture Notes in Computer Science*, pages 193–207. Springer, 2003.
5. S. Kremer, O. Markowitch, and J. Zhou. An Intensive Survey of Non-repudiation Protocols. *Computer Communications*, 25(17):1606–1621, 2002.
6. S. Kremer and J.-F. Raskin. A Game-Based Verification of Non-repudiation and Fair Exchange Protocols. In K. Guldstrand Larsen and M. Nielsen, editors, *12th Int. Conference Concurrency Theory, CONCUR*, volume 2154 of *Lecture Notes in Computer Science*, pages 551–565, Aalborg, Denmark, 2001. Springer.
7. A. McCullagh and W. Caelli. Non-Repudiation in the Digital Environment. Technical report, Peer-Reviewed Jounal on the Internet, 2000.
8. P. Ryan, M. Goldsmith, G. Lowe, B. Roscoe, and S. Schneider. *Modelling & Analysis of Security Protocols*. Addison Wesley, 2000.
9. S. Schneider. Formal Analysis of a Non-Repudiation Protocol. In *Proc. of 11th Computer Security Foundations Workshop, CSFW*, pages 54–65. IEEE Computer Society, 1998.
10. V. Shmatikov and J. C. Mitchell. Analysis of Abuse-Free Contract Signing. In *4th Int. Conference on Financial Cryptography, FC*, volume 1962 of *Lecture Notes in Computer Science*, pages 174–191. Springer, 2001.
11. J. Zhou and D. Gollmann. A Fair Non-repudiation Protocol. In *Proc. of the IEEE Symposium on Research in Security and Privacy*, pages 55–61, Oakland, CA, 1996. IEEE Computer Society.
12. J. Zhou and D. Gollmann. Observations on Non-repudiation. In *Advances in Cryptology – Int. Conference on the Theory and Applications of Cryptology and Information Security, ASIACRYPT*, volume 1163 of *Lecture Notes in Computer Science*, pages 133–144, Kyongju, Korea, 1996. Springer.
13. J. Zhou and D. Gollmann. Towards Verification of Non-repudiation Protocols. In *Proc. of Int. Refinement Workshop and Formal Methods Pacific*, pages 370–380, Canberra, Australia, 1998.