Automatic Analysis of a Non-Repudiation Protocol

Ruggero Lanotte [1]

Dipartimento di Scienze della Cultura, Politiche e dell'Informazione, Università dell'Insubria, Via Valleggio 11, 22100 Como, Italy

Andrea Maggiolo-Schettini [2] Angelo Troina [3]

Dipartimento di Informatica, Università di Pisa, Via F. Buonarroti 2, 56127 Pisa, Italy

Abstract

We define a probabilistic model for the analysis of a Non-Repudiation protocol that guarantees fairness, without resorting to a trusted third party, by means of a probabilistic algorithm. By using the PRISM model checker, we estimate the probability for a malicious user to break the non-repudiation property, depending on various parameters of the protocol.

1 Introduction

Repudiation is defined as denial by one of the entities involved in a communication of having participated in all or part of the communication. One speaks of repudiation of the origin if the originator of a message denies having sent the message, and of repudiation of receipt if the recipient of the message denies having received the message.

Protocols have been defined which ensure non-repudiation by making use of a trusted third party in the communication. Protocols involving no third party have been proposed in fault-less scenarios. Markowitch and Roggeman [13] give a probabilistic protocol which achieves fair non-repudiation services without the need of a third party and without further assumptions. In particular, the probabilistic protocol is fair up to a given tolerance ε. This tolerance depends on the values chosen for various parameters of the protocol.

[1] Email: `ruggero.lanotte@uninsubria.it`
[2] Email: `maggiolo@di.unipi.it`
[3] Email: `troina@di.unipi.it`

In [2], the protocol is described by means of a probabilistic process algebra and analyzed, in an untimed setting, through a notion of probabilistic weak bisimulation. In previous work [12] we have described the protocol by using Probabilistic Timed Systems and bisimulation for Probabilistic Timed Systems, in order to show fairness up to a given tolerance. In this paper we translate the Probabilistic Timed Systems modelling the protocol, into PRISM specifications [1], in order to estimate how the tolerance varies when the parameters of the protocol are varied.

In section 2 we describe the non-repudiation protocol, in section 3 we recall Probabilistic Timed Systems, in section 4 we use the Probabilistic Timed Systems to model the protocol, in section 5 we translate our model into PRISM specifications, in section 6 we show the results obtained by running the PRISM model checker.

2 A Probabilistic Non-Repudiation Protocol

In this section we describe a protocol that guarantees a non-repudiation service with a certain probability without resorting to a trusted third party [13]. In particular, such a probabilistic protocol is fair up to a given tolerance ε decided by the originator. Assume that an authentication phase precedes the protocol. We denote by $Sign_E(M)$ the encryption of message M under the private key of the entity E and with $\{M\}_K$ the encryption of M under the key K. Finally, we use t to denote a time stamp. The protocol can be described as follows (with the notation $R \to O : Msg$ we denote a message Msg sent by R and received by O):

$$\begin{array}{lll}
1. & R \to O : Sign_R(request, R, O, t) & \\
2. & O \to R : Sign_O(\{M\}_K, O, R, t) & (= M_1) \\
3. & R \to O : Sign_R(ack_1) & \\
4. & & \\
a._{1-p} & O \to R : Sign_O(M_r, O, R, t) & (= M_i) \\
 & R \to O : Sign_R(ack_i) & \\
 & \quad\quad \text{goto step 4} & \\
b._{p} & O \to R : Sign_O(K, O, R, t) & (= M_n) \\
5. & R \to O : Sign_R(ack_n) &
\end{array}$$

The recipient R starts the protocol by sending a signed, timestamped request to the originator O. This sends to R the requested message M ciphered under the key K, and waits for the ack from R (ack_i represents the acknowledgment related to message M_i). At step 4 the originator makes a probabilistic choice according to $p = \varepsilon$. At step $4a$ (taken with probability $1 - p$) O sends to R a random message M_r (i.e. a dummy key), receives the ack and returns to step 4, while at step $4b$ (taken with probability p) O sends to R the key K necessary to decrypt the message $\{M\}_K$. Upon reception of the last ack (ack_n), related to the message containing the key K, the originator terminates the protocol correctly. We suppose that each ack_i message carries the following semantics: "R acknowledges having received message M_i from O". This could be easily obtained, for instance, by assuming that each ack_i message contains an hash of message M_i.

Intuitively, the non-repudiation of origin is guaranteed by the messages M_1 and M_n (signed with the private key of O), while the non repudiation of receipt is given by the last message $Sign_R(ack_n)$. If the protocol terminates after the delivery of the last ack, both parties obtain their expected information, and the protocol is fair. If the protocol terminates before sending the message containing the key K, then neither the originator nor the recipient obtains any valuable information, thus preserving fairness. A strategy for a dishonest recipient consists in guessing the last message containing the key K, verifying whether a received message contains the needed key and then blocking the transmission of the last ack. Therefore, for the success of the protocol, it is necessary that the ack messages are sent back immediatly. The originator decides a deadline for the reception of each ack, after which, if the ack is not received, the protocol is stopped. Obviously, the cryptosystem must be adequately chosen, in such a way that the time needed to verify a key, by deciphering the message, is longer than the transmission time of an ack message. Anyway, a malicious recipient can try to randomly guess the message containing the key K, and in this case the probability for the recipient of guessing the last message depends on the parameter p chosen by the originator.

3 Probabilistic Timed System

We give a definition of Probabilistic Timed Systems as a subclass of Probabilistic Timed Automata [5,10,3,11], assuming discrete time domain. Let us assume a set X of integer variables, with a subset Y of variables called *clocks*. A *valuation* over X is a mapping $v : X \to \mathbb{Z}$ assigning natural values to clocks and integer values to variables in $X \setminus Y$. For a valuation v and a time value $t \in \mathbb{N}$, let $v+t$ denote the valuation such that $(v+t)(x) = v(x)$, for each integer variable $x \in X \setminus Y$, and $(v+t)(y) = v(y)+t$, for each clock $y \in Y$. The set of *constraints* over X, denoted $\Phi(X)$, is defined by the following grammar, where ϕ ranges over $\Phi(X)$, $x \in X$, $c \in \mathbb{Z}$ and $\sim \in \{<, \leq, =, \neq, >, \geq\}$:

$$\phi ::= x \sim c \,|\, \phi \wedge \phi \,|\, \neg\phi \,|\, \phi \vee \phi \,|\, true$$

We write $v \models \phi$ when *the valuation v satisfies the constraint ϕ*. Formally, $v \models x \sim c$ iff $v(x) \sim c$, $v \models \phi_1 \wedge \phi_2$ iff $v \models \phi_1$ and $v \models \phi_2$, $v \models \neg\phi$ iff $v \not\models \phi$,

$v \models \phi_1 \vee \phi_2$ iff $v \models \phi_1$ or $v \models \phi_2$, and $v \models \mathit{true}$.

An *assignment* over X is a set of expressions either of the form $x' = c$ or of the form $x' = y + c$, where $x, y \in X$ and $c \in \mathbb{Z}$.

With $\mathit{Ass}(X)$ we denote the set of sets of assignments $\{x'_1 = \rho_1, \ldots, x'_n = \rho_n\}$ such that $x_i \in X$ and $x_i \neq x_j$, for any $i \neq j$.

Let $B \in \mathit{Ass}(X)$; with $v[B]$ we denote the valuation resulting after the assignments in B. More precisely, $v[B](x) = c$ if $x' = c$ is in B, $v[B](x) = v(y)+c$ if $x' = y + c$ is in B, and $v[B](x) = v(x)$, otherwise.

A *Probabilistic Timed System* A is a tuple $(\Sigma, X, Q, q_0, \delta, \gamma, \pi)$, where:

- Σ is a finite alphabet of *actions*. With $\tau \in \Sigma$ we denote the *silent* or *internal* action.
- X is a finite set of variables with a subset Y of clocks.
- Q is a finite set of *states* and $q_0 \in Q$ is the *initial state*.
- $\delta \subseteq Q \times \Sigma \times \Phi(X) \times \mathit{Ass}(X) \times Q$ is a finite set of *transitions*. With $\delta(q)$, we denote the set of transitions starting from state q. More precisely, $\delta(q) = \{(q_1, \alpha, \phi, B, q_2) \in \delta \mid q_1 = q\}$.
- γ *is an interval function* such that for each $e = (q_1, \alpha, \phi, B, q_2) \in \delta$, it holds that $\gamma(e)$ is a closed interval of naturals. With $|\gamma(e)|$ we denote the natural value $u - l + 1$, where $\gamma(e) = [l, u]$. Intuitively, $\gamma(e)$ represents an interval of time within which the time t when the transition e will be performed is probabilistically chosen. It must also hold that the constraint ϕ is true during the interval $\gamma(e)$.
- π is a *probability function* such that for each state q and transition $e \in \delta(q)$, it holds that $\pi(e) \cdot \frac{1}{|\gamma(e)|}$ is the probability of performing the transition e from state q in a generic time t in $\gamma(e)$. Hence, we have a uniform distribution for time t, since for each time $t \in \gamma(e)$ the probability is fixed to $\pi(e) \cdot \frac{1}{|\gamma(e)|}$. Therefore we require that for each state q it holds that $\sum_{e \in \delta(q)} \pi(e) \in \{0, 1\}$.

A *configuration* of A is a pair (q, v), where $q \in Q$ is a state of A, and v is a valuation over X. The set of all the configurations of A is denoted with $\mathcal{S}_A$.

There is a *step* from a configuration $s_1 = (q_1, v_1)$ to a configuration $s_2 = (q_2, v_2)$ through action $a \in \Sigma$, after $t \in \mathbb{N}$ time units, written $s_1 \xrightarrow{(a,t)} s_2$, if there is a transition $e = (q_1, a, \phi, B, q_2) \in \delta$ such that $(v_1 + t) \models \phi$, $v_2 = (v_1 + t)[B]$ and $t \in \gamma(e)$. With $\mathit{prob}(s_1 \xrightarrow{(a,t)} s_2)$ we denote the probability $\sum_{e \in E} \pi(e) \cdot \frac{1}{|\gamma(e)|}$, where E is the set of transitions that could have triggered the step $s_1 \xrightarrow{(a,t)} s_2$, namely the set $\{e = (q_1, a, \phi, B, q_2) \in \delta \mid (v_1 + t) \models \phi \wedge v_2 = (v_1 + t)[B] \wedge t \in \gamma(e)\}$.

If s is a configuration, then with $\mathit{Adm}(s)$ we denote the set of steps $s \xrightarrow{(a,t)} s'$. The configuration s is called *terminal* iff $\mathit{Adm}(s) = \emptyset$.

An *execution fragment* starting from s_0 is a finite sequence of steps $\sigma = s_0 \xrightarrow{(a_1,t_1)} s_1 \xrightarrow{(a_2,t_2)} s_2 \xrightarrow{(a_3,t_3)} \dots \xrightarrow{(a_k,t_k)} s_k$ such that $s_0, s_1, \dots, s_k \in \mathcal{S}_A$, $a_1, a_2, \dots, a_k \in \Sigma$. We define $last(\sigma) = s_k$, $|\sigma| = k$ and σ^j the sequence of steps $s_0 \xrightarrow{(a_1,t_1)} s_1 \xrightarrow{(a_2,t_2)} \dots \xrightarrow{(a_j,t_j)} s_j$, where $j \leq k$. Moreover we define the following probability

$$P(\sigma) = \begin{cases} 1 & \text{if } k = 0 \\ P(\sigma^{k-1}) \cdot \frac{prob(s_{k-1} \xrightarrow{(a_k,t_k)} s_k)}{\sum_{g \in Adm(s_{k-1})} prob(g)} & \text{if } k > 0 \end{cases} .$$

The execution fragment σ is called *maximal* iff $last(\sigma)$ is terminal. We denote with $ExecFrag(s)$ the set of execution fragments starting from s.

An *execution* is either a maximal execution fragment or an infinite sequence $s_0 \xrightarrow{(a_1,t_1)} s_1 \xrightarrow{(a_2,t_2)} \dots$, where $s_0, s_1 \dots \in \mathcal{S}_A$, $a_1, a_2, \dots \in \Sigma$. We denote with $Exec(s)$ the set of executions starting from s. Finally, let $\sigma \uparrow$ denote the set of executions σ' such that $\sigma \leq_{prefix} \sigma'$, where *prefix* is the usual prefix relation over sequences. Assuming the basic notions of probability theory (see e.g. [7]), we define the probability space on the executions starting in a given configuration $s \in \mathcal{S}_A$ as follows. Let $Exec(s)$ be the set of executions starting in s, $ExecFrag(s)$ be the set of execution fragments starting in s, and $\Sigma_F(s)$ be the smallest sigma field on $Exec(S)$ that contains the basic cylinders $\sigma \uparrow$, where $\sigma \in ExecFrag(s)$. The probability measure *Prob* is the unique measure on $\Sigma_F(s)$ such that $Prob(\sigma \uparrow) = P(\sigma)$.

3.1 Parallel composition

Given two probabilistic timed systems A_1 and A_2, and given the set $L = \Sigma_{A_1} \cap \Sigma_{A_2}$, where Σ_{A_i} is the set of actions of the system A_i, we define the parallel composition of A_1 and A_2, denoted $A_1||^pA_2$, where $p \in [0,1]$. Intuitively, p represents the different advancing speeds for the two systems. The set of states of $A_1||^pA_2$ is given by the cartesian product of the states of the two systems A_1 and A_2. Given a state (r,q) of $A_1||^pA_2$, the set of transitions starting from (r,q) is obtained by the following rules:

- If from state r the system A_1 has a transition $e = (r, a, \phi, B, r')$ with action $a \notin L$ and probability p', $A_1||^pA_2$ has a transition $((r,q), a, \phi, B, (r',q))$ with probability $p \cdot p'$ and interval $\gamma(e)$.
- If from state q the system A_2 has a transition $e = (q, a, \phi, B, q')$ with action $a \notin L$ and probability p', $A_1||^pA_2$ has a transition $((r,q), a, \phi, B, (r,q'))$ with probability $(1-p) \cdot p'$ and interval $\gamma(e)$.
- If from state r the system A_1 has a transition $e = (r, a, \phi_1, B_1, r')$ with action $a \in L$ and probability p', and from state q the system A_2 has a transition $e' = (q, a, \phi_2, B_2, q')$ with probability p'' and $B_1 \cup B_2 \in Ass(X)$, then A_1 and

A_2 synchronize and $A_1||^p A_2$ has a transition $((r, q), a, \phi_1 \wedge \phi_2, B_1 \cup B_2, (r', q'))$ with probability $p \cdot p' + (1-p) \cdot p''$ and interval $\gamma(e) \cap \gamma(e')$.

When we omit parameter p from the composition operator we assume the two systems to have the same advancing speeds, and hence p equal to $\frac{1}{2}$.

4 Modelling the Non-Repudiation Protocol with Probabilistic Timed Systems

In this section we use the model of Probabilistic Timed Systems to formally describe the protocol seen in section 2. When no constraint is put on a transition we assume that it will be taken instantly. Moreover, if for the transition starting from a certain state q we omit probabilities, then the transitions with source q are equiprobable. Hence, if a state has only one transition e, then $\pi(e) = 1$, and, if a state has two transitions e and e', then $\pi(e) = \pi(e') = \frac{1}{2}$. We start with introducing the Probabilistic Timed Systems modelling an originator and a recipient behaving correctly. The originator (figure 1) is always ready to start a communication by accepting a request, sending the first message containing M encrypted with K (action $firstmes$) and receiving the first ack (see steps 1, 2 and 3 of the protocol in section 2). Then, in state q_3, with probability $1-p$, it sends a random message reaching state q_2 and, with probability p, sends the last message containing K, sets the variable l to 1, and reaches state q_4 (step 4 of the protocol). We do not model value passing, hence we simply call all these actions *message*. In state q_2 the reception of the ack message is modelled by the input action *ack*, while the expiration of the deadline (represented by the constant AD) is modelled by the action *stop* executed when the clock x assumes a value greater than AD. The fair termination of the protocol is reached when the originator receives the last ack (step 5 of the protocol) and performs the action *correctstop*. The protocol terminates in an unfair way if and only if the originator does not receive the ack related to the message containing K, and in such a case it executes the action $unfair$. The constants ad and AD used in the constraints of the *ack* transitions, represent an estimation of the minimum and maximum transmission delay of an ack message, respectively. In particular, we assume that an ack, sent within the net, will always arrive at destination in time t such that $ad \leq t \leq AD$.

We modeled the actions $request$, $firstmes$, and *message* as instant actions, since the time needed for the transmission of such messages in the network is not interesting in our analysis.

In figure 2, we show the system representing a recipient that behaves correctly. The recipient starts the protocol by sending a request, receives the first message, sends the first ack and reaches state r_3, from where, whenever it receives a message, it sends an ack back. The protocol terminates when the input action *correctstop* is executed.

The whole system representing the protocol is defined as $Orig||HRecip$,

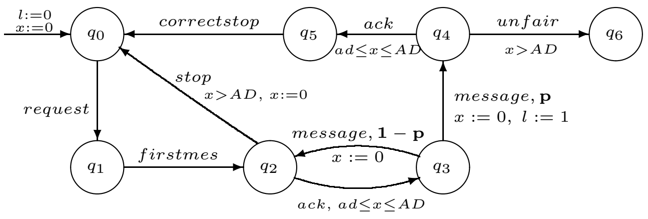

Fig. 1. Representation of $Orig$

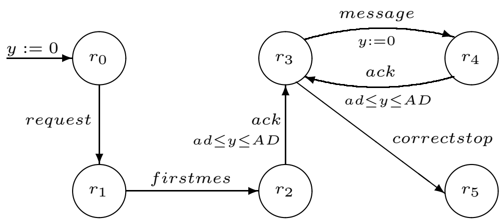

Fig. 2. Representation of $HRecip$

where originator and recipient synchronize through actions in the set $L = \{request, firstmes, ack, message, correctstop\}$.

The protocol ends in a fair correct way when the action *correctstop* is performed, i.e. when the system reaches the state (q_0, r_5). In particular, if both participants behave correctly, the unfair behavior cannot be executed; instead, it is possible to find a malicious recipient that receives the expected information and denies sending the final ack.

In figure 3 we show the system representing a malicious recipient that maximizes the probability of guessing the last message of the protocol (we assume that it knows the probability distribution chosen by the originator). It follows a Bernoulli distribution with parameter q to decide either to send the ack message (transition τ from state r_4 to state r_6) or to try to compute M by employing the last received message (transition from state r_4 to state r_5). We assume that the time necessary to decipher the message is within the interval $[dd, DD]$. Note that if $ad + dd < AD$ the recipient can send an ack even after failing to decipher the message (see transition from r_5 to r_6). So the originator should take care of the protocol parameters ad, AD, dd and DD. State r_7 represents, instead, the state reached by the malicious recipient when correctly guessing the last message. Since we set the variable l to 1 when the originator sends the last message, the malicious recipient succeeds in its strategy when in state r_5 such a variable has value 1 reaching the final state r_7. On the other hand, if variable l has value 0, the malicious recipient goes to state r_6, from which it tries to send the ack back.

The probability of executing the action $unfair$ for the system $Orig||HRecip$ is equal to 0, while the probability of executing it for the system $Orig||MRecip$

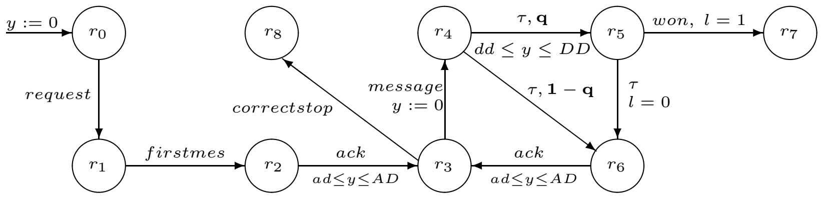

Fig. 3. Representation of $MRecip$

is (as $ad + dd > AD$):

$$z = p \cdot q \cdot \sum_{i=0}^{\infty} ((1-p) \cdot (1-q))^i = \frac{p \cdot q}{1 - (1-p) \cdot (1-q)}.$$

Given $0 < p \leq 1$ chosen by the originator and $0 < q \leq 1$, the maximum value for z is p, obtained by taking $q = 1$. The recipient model, which optimizes the probability of violating the fairness condition, is obtained by removing the transition labelled with τ from state r_4 to state r_6.

4.1 The case of slow networks

As we have seen the probability for the malicious user to break the protocol is always smaller than p, which is a parameter decided by the designer of the protocol. This happens, obviously, when the condition $ad + dd > AD$ holds. In such a case, in fact, the malicious user could only try to decrypt the first message with the last received one as key (risking in this case to stop the protocol), or send an ack to the originator.

On the other hand, the condition above holds only if the time needed to send/receive an ack within the network is smaller than the time needed to decrypt a message within a given cryptosystem. Could we still use the protocol with a reasonable margin of risk in a network where the maximum acknowledgement delay time is bigger than the maximum decryption time, or in a network that is frequently subject to congestion? If the condition $ad + dd > AD$ does not hold, in fact, the malicious user could try to send an ack to the originator even after trying to decrypt the last received message.

We are interested in analyzing how the probability of breaking the protocol fairness increases when operating with a network with a long round trip time (high values for parameter AD), and in networks that are frequently subject to congestions (high values for the length of the interval $[ad, AD]$).

5 The PRISM tool

PRISM [1] is a probabilistic model checker that allows modeling and analyzing systems which exhibit a probabilistic behavior. Given a description of the system to be modelled, PRISM constructs a probabilistic model that can be either a *discrete-time Markov chain* (DTMC), a *Markov decision process*

(MDP), or a *continuous-time Markov chain* (CTMC) [9]. On the constructed model PRISM can check properties specified by using a temporal logic (PCTL for DTMCs and MDPs, and CSL for CTMCs).

A system in PRISM is composed of modules and variables. A module has local variables and its behavior is expressed by commands of the form:

$$[sym]\ g \to \lambda_1 : u_1 + \cdots + \lambda_n : u_n$$

where *sym* is the synchronization symbol, g is a guard on all the variables in the system, and u_i is a set of updates on local variables. The constant λ_i is the probability of performing the update u_i. Constraints can be of the form $x \sim c$, where c is a natural and x a variable of the whole system. Updates are expressed by using primed variables; as an example $x' = 0$ represents the reset to 0 of variable x.

Since we used the model of DTMC, we use the PCTL specification language [8,6,4] to specify properties of systems.

The syntax of PCTL is given by the following grammar:

$$\phi ::= true \mid false \mid a \mid \phi \wedge \phi \mid \phi \vee \phi \mid \neg\phi \mid \mathcal{P}_{\sim p}[\psi]$$

$$\psi ::= X\phi \mid \phi\mathcal{U}^{\leq k}\phi \mid \phi\mathcal{U}\phi$$

where a is an atomic proposition, $\sim \in \{<, \leq, \geq, >\}$ is a relational operator, $p \in [0,1]$ is a probability, and k is an integer.

An atomic proposition a is satisfied or not by a given state of a Probabilistic Timed System. Symbol X denotes the "next state operator", symbol $\mathcal{U}$ denotes the "until" operator, and $\mathcal{U}^{\leq k}$ denotes the "bounded until" (i.e. within k steps) operator. Intuitively, $\phi_1\mathcal{U}\phi_2$ is satisfied when the formula ϕ_1 holds until ϕ_2 holds; $\phi_1\mathcal{U}^{\leq k}\phi_2$ is satisfied if ϕ_2 becomes true within k steps. Moreover, $\mathcal{P}_{\sim p}[\psi]$ is satisfied by a given set of computations iff the overall probability p' of the computations satisfying ψ is such that $p' \sim p$.

5.1 *Modeling the Non-Repudiation Protocol in PRISM*

In this section we show the PRISM description of the systems introduced in section 4, and the results of some properties verified on such models [4].

5.1.1 *Honest Recipient*

In figure 4 we show the PRISM code of the modules representing the originator (`Orig`) and an honest recipient (`HRecip`). A third module (`AckDel`) is used for modelling the probability distribution of the variable `t` representing the acknowledgement delay. We decided to use a uniform probability distribution

[4] Tests are done on a 1,15GHz AMD Athlon PC with Linux OS and 512 MB of RAM.

for the variable `t`. So, given the random variable `t` that can range in the interval [`ad`,`AD`], we have that:

$$\forall k \in [\mathtt{ad}, \mathtt{AD}] \qquad p(\mathtt{t} = k) = \frac{1}{\mathtt{AD} - \mathtt{ad} + 1},$$

where $p(\mathtt{t} = k)$ represents the probability that variable `t` assumes value k.

The constant `p1` used in the `Orig` module represents the probability p used in the Probabilistic Timed System *Orig* seen in section 4.

```
probabilistic

const AD = 3;
const ad = 1;

rate p1 = 0.001;

rate u1 = 1/(1+AD-ad);

module Orig
    q : [0..6];
    l : [0..1];

    [request]      (q=0)&(r=0)          ->  (q'=1);
    [firstmes]     (q=1)&(r=1)          ->  (q'=2);
    [ack_r]        (q=2)&(a=2)&(t<=AD)  ->  (q'=3);
    [stop]         (q=2)&(t>AD)         ->  (q'=0);
    [message]      (q=3)&(r=3)          ->  p1:(q'=4)&(l'=1) + (1-p1):(q'=2);
    [ack_r]        (q=4)&(a=2)&(t<=AD)  ->  (q'=5);
    [unfair]       (q=4)&(t>AD)         ->  (q'=6);
    [correctstop]  (q=5)&(r=3)          ->  (q'=0);
endmodule

module HRecip
    r : [0..5];

    [request]      (r=0)&(q=0)  ->  (r'=1);
    [firstmes]     (r=1)&(q=1)  ->  (r'=2);
    [ack_s]        (r=2)&(a=0)  ->  (r'=3);
    [message]      (r=3)&(q=3)  ->  (r'=4);
    [ack_s]        (r=4)&(a=0)  ->  (r'=3);
    [correctstop]  (r=3)&(q=5)  ->  (r'=5);
endmodule

module AckDel
    a: [0..2];
    t: [0..AD];

    [ack_s]  (a=0)&((r=2)|(r=4))  ->  (a'=1);
    []       (a=1)                ->  u1:(a'=2)&(t'=1) + u1:(a'=2)&(t'=2) + u1:(a'=2)&(t'=3);
    [ack_r]  (a=2)&((q=2)|(q=4))  ->  (a'=0);
endmodule
```

Fig. 4. Orig + HRecip

Given the above participants, the protocol always ends in a fair correct way. When the protocol ends in a fair way, the recipient state variable `r` assumes value 5. As a consequence, we can verify that the protocol always ends in a fair way by simply checking the property:

$$\mathcal{P}_{\geq 1}[true\,\mathcal{U}\ (r = 5)].$$

The model checking for the above property took 0,004 seconds, giving as result that the property is true in all the states of the model.

5.1.2 Malicious Recipient

In figure 5 we show the PRISM code of the modules representing the originator (`Orig`) and a malicious recipient (`MRecip`). As in the previous case, a third module (`AckDel`) is used for modelling the uniform probability distribution of the variable `t` representing the acknowledgement delay. Also the random variable `x`, used in the module `MRecip` and representing the time needed to decrypt a message, follows a uniform distribution with range [`dd`,`DD`]. The constant `q1` used in the `MRecip` module represents the probability q used in the Probabilistic Timed System *MRecip* seen in section 4.

6 Experimental Results

Setting :	set_a	set_b	set_c	set_d
`ad`	10	1	1	1
`AD`	13	4	12	12
`dd`	14	1	1	1
`DD`	15	5	2	2
`p1`	0.1	0.1	0.1	0.001

Table 1

We studied the probability for a malicious user of breaking the fairness of the non-repudiation for several settings of our model parameters. More precisely in the settings set_a, set_ b, set_c and set_d, we studied the probability of breaking the protocol fairness as a parameter of the value of `q1`, assuming various values for the constants `ad`, `AD`, `dd`, `DD` and `p1` (see table 1). Since the malicious user's state variable `r` assumes value 7 when it successfully gets its information without sending the last ack, the properties we used in order to estimate the probability of breaking the protocol fairness are of the form:

$$\mathcal{P}_{\geq v}[true\, \mathcal{U}\ (r = 7)].$$

So we can approximate to v the probability of breaking the protocol when the property $\mathcal{P}_{\geq v}[true\, \mathcal{U}\ (r = 7)]$ is satisfied by the initial state of the model, and the property $\mathcal{P}_{\geq v+\varepsilon}[true\, \mathcal{U}\ (r = 7)]$ is not, with ε a small value. The checking procedure of a property of the above type always took less then half a second.

```
probabilistic

const AD = 3;
const ad = 1;
const DD = 2;
const dd = 1;

rate p1 = 0.001;
rate q1 = 1;

rate u1 = 1/(1+AD-ad);
rate v1 = 1/(1+DD-dd);

module Orig
    q : [0..6];
    l : [0..1];

    [request]      (q=0)&(r=0)            ->  (q'=1);
    [firstmes]     (q=1)&(r=1)            ->  (q'=2);
    [ack_r]        (q=2)&(a=2)&(t+x<=AD)  ->  (q'=3);
    [stop]         (q=2)&(t+x>AD)         ->  (q'=0);
    [message]      (q=3)&(r=3)            ->  p1:(q'=4)&(l'=1) + (1-p1):(q'=2);
    [ack_r]        (q=4)&(a=2)&(t+x<=AD)  ->  (q'=5);
    [unfair]       (q=4)&(r=7)            ->  (q'=6);
    [correctstop]  (q=5)&(r=3)            ->  (q'=0);
endmodule

module MRecip
    r : [0..8];
    x : [0..DD];

    [request]      (r=0)&(q=0)  ->  (r'=1);
    [firstmes]     (r=1)&(q=1)  ->  (r'=2);
    [ack_s]        (r=2)&(a=0)  ->  (r'=3);
    [message]      (r=3)&(q=3)  ->  (r'=4);
    []             (r=4)        ->  q1:(r'=5) + (1-q1):(r'=6)&(x'=0);
    []             (r=5)&(l=0)  ->  v1:(r'=6)&(x'=1) + v1:(r'=6)&(x'=2);
    [won]          (r=5)&(l=1)  ->  (r'=7);
    [ack_s]        (r=6)&(a=0)  ->  (r'=3);
    [correctstop]  (r=3)&(q=5)  ->  (r'=8);
endmodule

module AckDel
    a : [0..2];
    t : [ad..AD];

    [ack_s]  (a=0)&((r=2)|(r=6))  ->  (a'=1);
    []       (a=1)                ->  u1:(a'=2)&(t'=1) + u1:(a'=2)&(t'=2) + u1:(a'=2)&(t'=3);
    [ack_r]  (a=2)&((q=2)|(q=4))  ->  (a'=0);
endmodule
```

Fig. 5. Orig + MRecip

Figures 6, 7 and 8 show the probability of breaking the protocol fairness as a function of `q1` for the parameter settings in table 1.

In the set_a setting, we have that $\texttt{ad}+\texttt{dd} > \texttt{AD}$. In such a case the malicious recipient could not send an ack after trying to decrypt the first message using the last received one as key, and so the probability of breaking the protocol fairness, shown in figure 6, is given by the formula $\frac{\texttt{p1}\cdot\texttt{q1}}{1-(1-\texttt{p1})\cdot(1-\texttt{q1})}$. As we have seen, the maximum value for this probability is represented by the parameter `p1`.

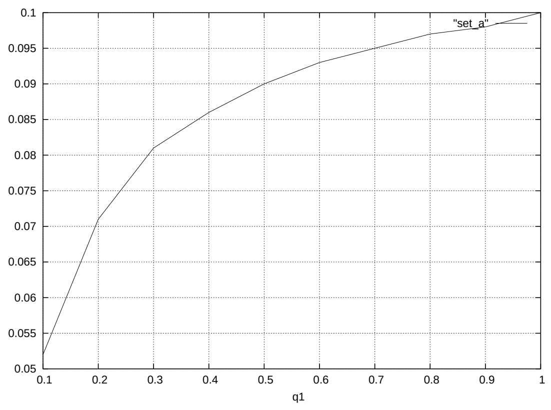

Fig. 6. Varying `q1` in an ideal network.

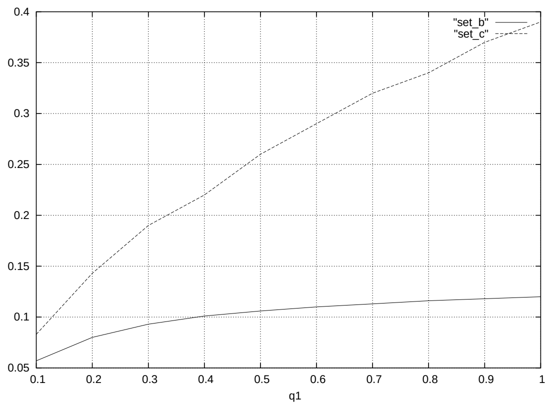

Fig. 7. Varying `q1` in slow networks.

In the set_b, set_c and set_d settings we have, instead, that $\mathtt{ad} + \mathtt{dd} \leq \mathtt{AD}$. With these three settings we propose a first analysis of the case of slow networks described in section 4.1. In such a case, as can be seen from figures 7 and 8, we have that the probability of breaking the protocol fairness increases as `q1` goes to 1 and reaches also values that are bigger than `p1`.

Setting :	set_e	set_f	set_g
`ad`	1	1	1
`dd`	1	1	1
`DD`	4	4	4
`p1`	0.001	0.01	0.1
`q1`	1	1	1

Table 2

In the settings `e`,`f`,`g`, we studied the probability of breaking the protocol

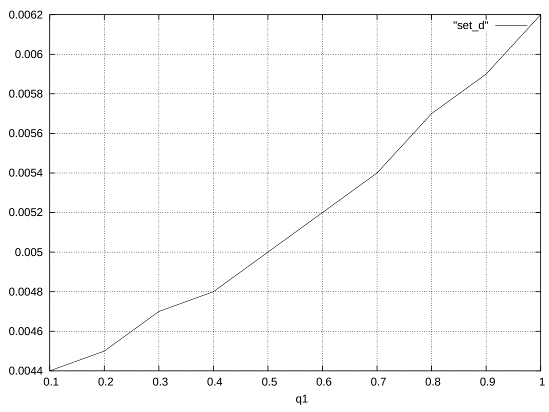

Fig. 8. Varying `q1` in a slow network, given a low `p1`.

fairness as a parameter of the value of `AD`, assuming various values for the constants `ad`, `dd`, `DD` and `p1` (see Tab. 2).

Figure 9 shows the probability of breaking the protocol fairness as a function of `AD` for the parameter settings in Table 2. In particular those settings model the case of "slow and congestioned network" described in section 4.1.

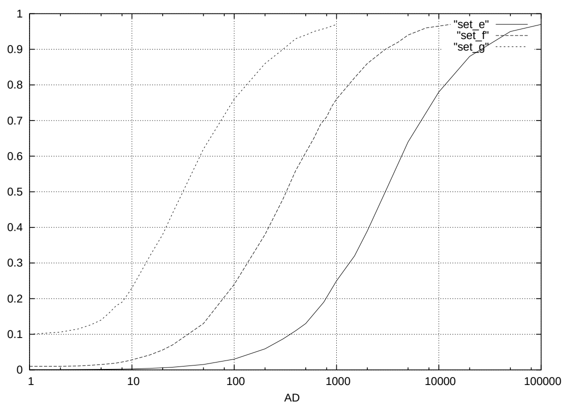

Fig. 9. Analysis of congestioned networks.

Obviously, as `AD` increases, also the probability of breaking the non-repudiation protocol increases. But, as it can be seen from figure 9, with a small value of `p1`, the protocol could be reasonably used also in the case when the time needed to send an ack is bigger than the time needed to decrypt a message.

7 Conclusions

We have given a model of the non-repudiation protocol proposed by Markowitch and Roggeman [13], and we studied the probability that the protocol achieves fair non-repudiation by varying parameters of the protocol. In particular, we analyzed the case when the protocol is run in networks with a long round trip time.

An important implementation detail when constructing our PRISM models regards the choice of how the probabilistic time variable `t` is modeled. When it was reasonably feasible we simply listed all the possible values that `t` could assume. We did this for the cases modeled with the specifications given in set_a, set_b, set_c and set_d, expressing the list in the second transition of the `AckDel` modules. Since the parameter `t` assumed extremely large values in the tests involving the specifications of set_e, set_f and set_g (according to the parameter `AD`), it was practically infeasible to list all the values that `t` could assume. However, since in these latter cases the time needed to decrypt a message is assumed to be in the interval $[1, 4]$, the only interesting values for `t` are $\mathtt{AD}, \mathtt{AD} - 1, \ldots, \mathtt{AD} - 3$. For any other value of `t` contained in the interval $[\mathtt{ad}, \mathtt{AD} - 4]$, the malicious user succeeds in its attempt to decrypt the message and then to send the ack (in this case, in fact, $\mathtt{t} + \mathtt{x}$ is always equal or smaller than `AD`). For this reasons, we simply used the following transition in the module `AckDel`:

```
[] (a=1) -> u1*(AD-ad-3):(a'=2)&(t'=ad) +
+ u1:(a'=2)&(t'=AD-3) + u1:(a'=2)&(t'=AD-2) + u1:(a'=2)&(t'=AD-1) + u1:(a'=2)&(t'=AD);.
```

Hence, the first update (concerning all the cases when $\mathtt{t} \in [\mathtt{ad}, \mathtt{AD} - 4]$) is taken with probability $\mathtt{u1} \cdot (\mathtt{AD} - \mathtt{ad} - 3)$, while all the other interesting updates are taken with probability `u1`.

References

[1] PRISM URL: http://www.cs.bham.ac.uk/~dxp/prism.

[2] A. Aldini, R. Gorrieri: *Security Analysis of a Probabilistic Non-repudiation Protocol.* Proc. of PAPM-PROBMIV '02, LNCS **2399**, 17–36, 2002.

[3] R. Alur, C. Courcoubetis and D. L. Dill: *Verifying Automata Specifications of Probabilistic Real-Time Systems.* Real-Time:Theory in Practice, LNCS **600**, 28–44, 1992.

[4] C. Baier and M. Kwiatowska: *Model Checking for a Probabilistic Branching Time Logic with Fairness.* Distributed Computing 11(3), 125–155, 1998.

[5] D. Beauquier: *On Probabilistic Timed Automata* Theoretical Computer Science **292**, 65–84, 2003.

[6] A. Bianco and L. de Alfaro: *Model Checking of Probabilistic and Nondeterministic Systems.* Proc. Int. Conference on Foundation of Software Technologies and Theoretical Computer Science, LNCS **1026**, 499–513, 1995.

[7] P. R. Halmos: *Measure Theory.* Springer-Verlag, 1950.

[8] H. Hansson and B. Jonsson: *A Logic for Reasoning About Time and Probability.* Formal Aspects of Computing 6(5), 512–535, 1994.

[9] M. Kwiatkowska: *Model Checking for Probability and Time: From Theory to Practice.* LICS'03, IEEE CS Press, 351–360, 2003.

[10] M. Kwiatkowska, G. Norman, R. Segala, J. Sproston: *Automatic Verification of Real-time Systems with Discrete Probability Distribution.* ARTS'99, LNCS **1601**, 75–95, 1999.

[11] M. Kwiatkowska, G. Norman, R. Segala, J. Sproston: *Verifying Quantitative Properties of Continuous Probabilistic Real-Time Graphs.* CONCUR'00, LNCS **1877**, 123–137, 2000.

[12] R. Lanotte, A. Maggiolo-Schettini, A. Troina: *Weak Bisimulation for Probabilistic Timed Automata and Applications to Security.* SEFM'03, IEEE CS Press, 34–43, 2003.

[13] O. Markowitch, Y. Roggeman: *Probabilistic Non-Repudiation without Trusted Third Party.* 2nd Conference on Security in Communication Network, 1999.