Available online at www.sciencedirect.com
SCIENCE DIRECT
Electronic Notes in Theoretical Computer Science
ELSEVIER Electronic Notes in Theoretical Computer Science 99 (2004) 205–227
www.elsevier.com/locate/entcs

Specification and Verification of Protocols With Time Constraints[1]

Margherita Napoli[†] Mimmo Parente[†] Adriano Peron[‡]

[†] *Dipartimento di Informatica e Applicazioni, Universtità degli Studi di Salerno, I-84041 Baronissi(Salerno).*
[‡] *Dipartimento di Scienze Fisiche, Università di Napoli Federico II, Via Cintia, I-80126 Napoli.*

Abstract

In this paper we face the problem of specifying and verifying security protocols where temporal aspects explicitly appear in the description. For these kinds of protocols we have designed a specification formalism, which consists of a state-transition graph for each participant of the protocol, with edges labelled by trigger/action clauses. The specification of a protocol is translated into a Timed Automaton on which standard techniques of model checking can be exploited (properties to be checked can be expressed in a linear/branching untimed/timed temporal logic). Along all the presentation we use, as running example, a two parties non-repudiation protocol for which we show how our framework applies in the verification of the fairness property for the protocol (establishing whether there is a step of the protocol in which one of the two participants can take any advantage over the other).

Keywords: Specification, Model-Checking, Timed Automata, Non-Repudiation Protocol

1 Introduction

From the early 90s, formal methods have been profitably used in various phases of the design of cryptographic protocols (specification, construction and verification) since a number of examples have shown that their informal design is error prone. Many works have been then devoted to formal specification

[1] This work was supported by the MURST in the framework of the project "Metodi Formali per la Sicurezza ed il Tempo" (MEFISTO).

doi:10.1016/j.entcs.2004.02.009

and analysis of cryptographic protocols, leading to a number of different approaches and encouraging results (e.g. see [8]). However, most of the proposed techniques consider cryptographic protocols where concrete information about timing of events is not crucial (e.g. delay, timeout, timed disclosure or expiration of information do not determine the correctness of the protocol) and details on some low level timing aspects of the protocol are abstracted (e.g. timestamps, duration of channel delivery etc).

In this paper we focus on the problem of specifying and verifying security protocols where temporal aspects explicitly appear in the specification and are considered in the verification. The formal basic notation we use is that of Timed Automata [2] and the approach to verification is *model checking*. A variety of tools have been proposed in recent years for the verification of real-time systems described by means of Timed Automata. For instance, the tool KRONOS [4] developed at VERIMAG, supports model checking of branching time requirements. The UPPAAL toolkit [7] allows checking of safety and bounded liveness properties.

Unfortunately, the formalism of Timed Automata, is a rather low level formalism, unsuitable to express a high level specification of security protocols and it would be hard for the protocol designer to use it as a specification language. Timed Automata lacks the ability of explicitly representing parallelism and communication between parallel components (each participant to a protocol is naturally described as a component of the specification acting in parallel with other component/participants). Moreover, the specification of a security protocol usually requires the description of structured messages exchanged by participants possibly composed by using cryptographic primitives (encryption, hashing, signature etc).

In this paper we propose a specification formalism called *Message passing Timed Automata* (MTA, for short) which retains the graphical nature of Timed Automata (state-transition diagrams) and allows the specification of a cryptographic protocol in a style which is very close to the way a protocol designer is accustomed to use. In particular, MTA allow the explicit representations of the protocol parties and the communication among parties. Parties communicate by sending/receiving structured messages belonging to a term algebra suitable to express cryptographic primitives and concepts (e.g. public/private keys, encryption/decryption, hashing, nonces etc). As concerns timing aspects, three kinds of temporal constraints can be differently expressed in the formalism: temporal constraints of the control flow (e.g. usual delays and timeouts associated with performing some action), temporal constraints over the availability and usability of (communicated) pieces of information (i.e. disclosure and expiration of messages) and duration of channel delivery.

The connection with the more basic formalism of Timed Automata is not lost. In fact, the semantics of the specification language is given in terms of Timed Automata thus obtaining an executable and verifiable (in the standard framework of model checking) specification. The Timed Automaton obtained by compiling a protocol specification encodes in its states all the detailed information (including temporal aspects) necessary for proving relevant properties of the protocol.

The idea of using Timed Automata for specifying real time systems and proving security properties is not new (e.g. see [3,6]). Our approach differs in that Timed Automata are not the specification language itself but the front-end of a new specification language specialized for security protocols. From this perspective, our work is closer to [5], where the specification language is a timed process algebra. The two approaches differ in the treatment of time (discrete versus continuous) and in the verification techniques.

To support our formalism, we provide an application to a very well-known non-repudiation protocol, see [9]. In this protocol Alice sends a message to Bob and at the end no one of them can claim not having sent or received the message. More precisely at the end of the protocol Bob has collected enough information to prove that Alice is the source of the message, and Alice has enough information as well to prove that Bob has indeed received her message. Thus no one can claim the false (not having sent that message and not having received it), as the other will provide evidences of the contrary. Based on this protocol we model-check the following *fairness property*: there is no step in which if the protocol would stop, one of the two parties has an "advantage" on the other.

The rest of the paper is organized as follows: in the next section we recall the non-repudiation protocol of Zhou and Gollmann [9]. In section 3.1 we present the specification language along with the resulting MTAs for the example protocol and in section 4 we show the translation into Timed Automata. Finally we conclude with the verification of the fairness property for the example protocol in section 5 and some conclusions in the last section.

2 The Zhou-Gollmann efficient non-repudiation protocol

In this section we briefly recall the Zhou-Gollmann protocol of [9]. Given two parties, Alice and Bob, a non-repudiation protocol aims at giving to both parties evidences of the sending and the receipt of the message. More precisely, when the protocol run to deliver a message from Alice to Bob terminates, the following properties are fulfilled:

- *Non-repudiation of Origin (NRO)* Bob, interacting with Alice collects enough information to provide evidence of origin of the message and can use such an evidence as a proof if Alice denies having sent it;
- *Non-repudiation of Receipt (NRR)* Alice interacting with Bob, collects enough information to provide evidence of the receipt of the message and can use such an evidence as a proof if Bob denies having received it.

To efficiently implement the non-repudiation property, the authors considered another party in the protocol, a *Trusted Third Party*, that intervenes only if a party cannot get the expected non repudiation evidence, providing the impaired party with the desired evidence.

Fairness Property. The protocol satisfies the *fairness* property: it provides the sender and the receiver with valid irrefutable evidence after its completion, without giving a party any advantage over the other at *any* stage of the protocol [ZG97]. As the evidence of the sending is given by the message itself (or something else shipped along with it), the originator needs an acknowledgement of the receipt of the message. Fairness can be broken for two reasons: either the communication channel is faulty and a transmitted message is never delivered or a party does not play *fair* by not adhering to the protocol rules. We will assume that the channel is resilient, that is, it is never faulty and *always* delivers the message transmitted in a finite *unknown* amount of time. By paraphrasing [9], the main idea of the protocol is to split the message into two parts, a *commitment* and a key K, which is sent both to B and to the TTP. If a dispute occurs, both the parties have the ability to retrieve the key from the TTP. Let us first give some notation necessary to describe the protocol: M is the message sent from A to B, T is the timeout, K is the key of A, C is the commitment (i.e., M cyphered with K), L is a unique label chosen by A to identify a protocol run, f is a flag indicating the purpose of a message, S_A and S_B are A's and B's private keys, $sK(M)$ is the signature of message M with key K, EOO is the evidence of origin of C, EOR is the evidence of receipt of C, sub_K is the evidence of submission of K and finally, con_K is the evidence of confirmation of K issued by the TTP .

Moreover, the specification of the protocol exploits the following shorthands:

$$EOO = sS_A(f_{EOO}\ ,B,L,T,C)$$
$$EOR = sS_B(f_{EOR}\ ,A,L,T,C)$$
$$EOO\_K = sS_A(f_{EOO\_K}\ ,B,L,K)$$
$$EOR\_K = sS_B(f_{EOR\_K}\ ,A,L,K)$$
$$sub\_K = sS_A(f_{SUB}\ ,B,L,K)$$

$con\_K = sS_{TTP}\ (f_{CON}\ ,A,B,L,K)$

The specification is now as follows:

(i) $A \rightarrow B : f_{EOO}\ , B, L, T, C, EOO$

(ii) $B \rightarrow A : f_{EOR}\ , A, L, EOR$

(iii) $A \rightarrow B : f_{EOO\_K}\ , B, L, K, EOO\_K$

(iv) $B \rightarrow A : f_{EOR\_K}\ , A, L, EOR\_K$

If B does not receive the key K, at step 3, the protocol halts satisfying the fairness property. On the other side, if A does not receive the message of step 4, then she starts the following recovery phase:

3' $A \rightarrow TTP\ : f_{SUB}\ , B, L, K, sub\_K$

4' $B \leftrightarrow TTP\ : f_{CON}\ , A, B, L, K, con\_K$

4' $A \leftrightarrow TTP\ : f_{CON}\ , A, B, L, K, con\_K$

Evidences. If the protocol stops legally, then A and B get the non-repudiation evidences (EOR and EOR_K for A, and EOO and EOO_K for B). Otherwise, if something goes wrong, A starts the recovery phase and both get the evidences with the help of the TTP (EOO and con_K for A, and EOR and con_K, for B). Let us note that the parameter L is needed to specify a protocol run (it is chosen by Alice at the very beginning along with T). For a more detailed presentation of the protocol the reader is referred to [9].

3 Specification

In this section we first introduce the formalism of Message passing Timed Automata giving its intuitive semantics (the formal semantics is given in section 4), and then we specify the Zhou-Gollman non-repudiation protocol in the defined setting.

3.1 Message passing Timed Automata

We start by defining the algebra allowing to express structured messages used in the communication among participants to a protocol section. The algebra has operations corresponding to the most widely used cryptografic primitives (e.g. encryption, decryption, hashing, signature etc.) and operations for associating temporal constraints to messages (e.g. timed disclosure, expiration etc.).

Let $\mathcal{M}$, $\mathcal{K}$, $\mathcal{PK}$, $\mathcal{I}$, and $\mathcal{N}$ be pairwise disjoint alphabets for *messages*, *keys*, *public keys*, *identities* and *nonces*, respectively, and let $\Sigma = \mathcal{K} \cup \mathcal{PK} \cup \mathcal{M} \cup \mathcal{I} \cup \mathcal{N}$.

Definition 3.1 Let $\mathcal{X}$ be an alphabet of formal parameters. The set of *structured messages* over Σ and $\mathcal{X}$, denoted by $SM_{\Sigma,\mathcal{X}}$, is inductively defined as follows:

- $m \in SM_{\Sigma,\mathcal{X}}$, for any $m \in \Sigma \cup \mathcal{X}$;
- $!X \in SM_{\Sigma,\mathcal{X}}$, for any $X \in \mathcal{X}$;
- $(m, m') \in SM_{\Sigma,\mathcal{X}}$, for any $m, m' \in SM_{\Sigma,\mathcal{X}}$;
- $\{m\}_K \in SM_{\Sigma,\mathcal{X}}$, for any $m \in SM_{\Sigma,\mathcal{X}}$ and $K \in \mathcal{K} \cup \mathcal{PK}$;
- $h(m), \dagger(m), sign_{id}(m) \in SM_{\Sigma,\mathcal{X}}$, for any $m \in SM_{\Sigma,\mathcal{X}}$ and $id \in \mathcal{I}$;
- $\Delta_\tau(m), \Theta_\tau(m), I^{\tau_2}_{\tau_1}(m) \in SM_{\Sigma,\mathcal{X}}$, for any $m \in \Sigma$, and $\tau, \tau_1, \tau_2 \in \mathcal{Q}^{\geq 0}$ with $0 < \tau_1 \leq \tau 2$.

A structured message m is *ground* if any formal parameter symbol occurring in m is within the scope of a ! symbol. The set of ground messages will be denoted by $SM_{\Sigma,\emptyset}$. A message is *timed* if it contains any occurrence of a (sub)message of the form $\Delta_\tau(m)$, $\Theta_\tau(m)$ or $I^{\tau_2}_{\tau_1}(m)$.

In the above definition $!X$ can be interpreted as the ground message (if any) associated with the formal parameter X, $\{m\}_K$ as the encryption of message m with a private or public key K; $h(m)$ as the hash of the message m; $sign_{id}(m)$ as the signature of message m with the identity id. Pairing of (m, m') of messages m and m' allows to construct structured messages. As concerns timing messages, we have the following: $\Delta_\tau(m)$ represents the fact that m is disclosed only at the elapsing of an interval of time τ since the communication of m, $\Theta_\tau(m)$ that m expires after an interval of time τ (since the communication of m), and $I^{\tau_2}_{\tau_1}(m)$ the fact that m is disclosed after τ_1 and expires after τ_2; $\dagger(m)$ represents the fact that m is expired.

Notice that temporal constraints can be associated only with non-structured messages (i.e. messagges in Σ).

Each participant in a protocol execution collects evidences by receiving structured messages sent by the other participants. In the following, for and identity id, we define a relation $\vdash_{id}$ describing the set of evidences (namely, masseges known to or composable by a participant of identity id) which can derived (or composed) from a collection of structured messages.

For an identity symbol $id \in \mathcal{I}$, the relation of *evidence derivation* $\vdash_{id} \subseteq 2^{SM_{\Sigma,\mathcal{X}}} \times SM_{\Sigma,\mathcal{X}}$ is recursively defined by the following rules (for simplicity we use infix notation):

- $Kw \vdash_{id} m$ if $m \in Kw$;
- $Kw \vdash_{id} K$ if $K \in \mathcal{PK}$;
- $Kw \vdash_{id} h(m)$ if $Kw \vdash_{id} m$;

- $Kw \vdash_{id} sign_{id}(m)$ if $Kw \vdash_{id} m$;
- $Kw \vdash_{id} m$ if $Kw \vdash_{id} sign_{id'}(m)$;
- $Kw \vdash_{id} (m_1, m_2)$ if $Kw \vdash_{id} m_1$ and $Kw \vdash_{id} m_2$;
- $Kw \vdash_{id} m_1$ and $Kw \vdash_{id} m_2$ if $Kw \vdash_{id} (m_1, m_2)$;
- $Kw \vdash_{id} \{m\}_K$ if $Kw \vdash_{id} m$ and $Kw \vdash_{id} K$;
- $Kw \vdash_{id} m$ if $Kw \vdash_{id} \{m\}_K$ and $Kw \vdash_{id} K$;
- $Kw \vdash_{id} m$ if $Kw \vdash_{id} \Theta_\tau(m)$;
- $Kw \vdash_{id} m$ if $Kw \vdash_{id} \dagger(m)$.

Notice that any public key is known; a message can be hashed and signed (with identity *id*); pair of messages can be coupled and component messages of a couple can be extracted; an encrypted message can be decrypted if the encryption key is known, and a message can be encrypted by a known key; a message with expiration time is known, whereas a message with a (non null release time) is not.

Participants to a protocol are described by state transition diagrams where transition are labelled by events representing sending and receiving a structured message along a channel, called *trigger* and *action*, respectively.

More formally, given a set Γ of *channels*, the set of *triggers* over Γ, written $Trigger_\Gamma$, is the set

$$\{True\} \cup \{?\alpha(m) : \text{with } \alpha \in \Gamma, \text{ and } m \in SM_{\Sigma,\mathcal{X}}\}.$$

The set of *actions* over Γ, written $Action_\Gamma$, is the set:

$$\{Nil\} \cup \{!\alpha(m) : \text{with } \alpha \in \Gamma, \text{ and } m \in SM_{\Sigma,\mathcal{X}}\}.$$

Definition 3.2 A *Message passing Timed Automaton*, *MTA* for short, over an alphabet Σ, a set of parameters $\mathcal{X}$ and a set of channels Γ, is a tuple $\langle C_1, \ldots, C_n, \lambda, \eta \rangle$, where

- $\lambda : \Gamma \to \mathcal{Q}^{\geq 0} \times (\mathcal{Q}^{\geq 0} \cup \{\omega\})$ is the *timing channel* function;
- $\eta : \{1, \ldots, n\} \to \mathcal{I}$ is the *identity* (injective) function;
- any *sequential component* C_i, with $1 \leq i \leq n$, is a tuple

$$\langle S_i, I_i, \delta_i, Kw_i \rangle, \text{ where}$$

  - S_i is the (finite) set of *states* (we assume that $S_i \cap S_j = \emptyset$, for $i \neq j$);
  - $I_i \subseteq S_i$ is the set of *input states*;
  - $\delta_i \subseteq S_i \times Trigger_\Gamma \times Action_\Gamma \times S_i^\Delta \times S_i^\Theta \times S_i$ is the *transition relation*, where, for a set of states S, S^Δ is the set $\{(s, \tau) : s \in S, \tau \in \mathcal{Q}^{\geq 0}\}$ and S^Θ

is the set $\{(s,\tau) : s \in S, \tau \in \mathcal{Q}^{\geq 0} \cup \{\omega\}\}$;

· $Kw_i : I_i \rightarrow 2^{\Sigma}$ is the *initial evidence* function.

In the above definition, the timing channel function, gives for a channel α, the endpoints of an interval bounding the time required for delivering a message. For instance, if $\lambda(\alpha) = (2,5)$, then α is an operational channel which takes at least time 2 to deliver a message and which is known to deliver the message within time 5; if $\lambda(\alpha) = (0,\omega)$, then α is a resilient channel which delivers the mesage in a finite but unpredictable amount of time.

The identity function is an injective fuction naming each sequential conponent by an identity symbol.

Each sequential component of the MTA is a state transition diagram describing a participant in the protocol. It consists of a finite set of states, a subset of which, are input states (i.e. initial states for a protocol execution). The initial evidence function Kw_i, assignes, to each initial state, the set of evidence (i.e. private keys, identities, nonces, unstructured messages) which are supposed to be known by the participant at the beginning of the execution of a protocol. The actual set of evidences known by a participant to a protocol section can be augmented (from its initial state) by receiving messages from other participants, and it is reset to the initial conditions every time an input state is reached.

Any transition transition from a state s_1 to s_2 is a tuple of the form

$$\langle s_1, trigger, action, delay, timeout, s_2\rangle, \text{ labelled by}$$

- a *triggering event*: if the trigger is an expression of the form $?\alpha(m)$, the transition is enabled to fire when it receives a message along the channel α; if the trigger equals $True$, the transition is unconditionately enabled to fire;
- an *action* which is taken when the transition fires: if the action is an expression of the form $!\alpha(m)$, a message m is sent in the channel α; if the action equals Nil, no action is performed;
- a *delay* having the form (s,τ), with $s \in S_i$, meaning that the transition can be performed only an amount of time τ after the last entering of state s;
- a *timeout* having the form (s,τ), with $s \in S_i$, meaning that the transition can be performed only within an amount of time τ after the last entering of state s.

Notice that the performance of a transition is conditioned to the fulfilment of two kinds of constraints: the trigger of a communication event, and the temporal constraints imposed by a delay and/or a timeout.

Sequential components communicate by synchronously sending and receiv-

ing messages along channels (one to one handshaking). Asynchronous sending is not allowed. Communication has also to preserve the structure of expected messages. A message m occurring in a receiving event $?\alpha(m)$ is a structured message usually containing formal parameters. On the contrary, a message m' sent by an action $!\alpha(m')$ is a ground message. A synchronization of $?\alpha(m)$ with $!\alpha(m')$ can take place only if m and m' have compatible structure, i.e. they are unifiable.

For instance, a message m of the form $(X, Sign_A(X))$, with X a parameter symbol, is unifiable with a message of the form $(\overline{m}, Sign_A(\overline{m}))$, and it is not unifiable with a message of the form $(m_1, Sign_A(m_2))$, with $m_1 \neq m_2$. In the former case, the side effect of a synchronization is the binding of the actual value $\overline{m}$ to the formal parameter X.

The outlined notions of unifiability of terms and assignment of actual values to formal parameters, are formalized by the *message unification relation* defined as follows.

For a set of keys $T \subseteq \mathcal{K}$, and a function $\rho : \mathcal{X} \to SM_{\Sigma,\emptyset}$, the *message unification relation* $\Longrightarrow_{MU}^{T,\rho} \subseteq SM_{\Sigma,\mathcal{X}} \times SM_{\Sigma,\emptyset} \times SM_{\Sigma,\mathcal{X}} \times 2^{\mathcal{X} \times SM_{\Sigma,\emptyset}}$ associates, with a couple of messages m_1 and m_2 (with m_2 ground), a couple consisting of a unifying message m_u and a partial fuction σ_u from parameters to (ground) messages (for readability, we shall write $(m_1, m_2) \Longrightarrow_{MU}^{T,\rho} (m_u, \sigma_u)$ instead of $(m_1, m_2, m_u, \sigma_u) \in \Longrightarrow_{MU}^{T,\rho}$. The relation $\Longrightarrow_{MU}^{T,\rho}$ is inductively defined as follows:

- $(m_1, m_2) \Longrightarrow_{MU}^{T,\rho} (m_2, \emptyset)$ if m_1 is ground and $m_1 = m_2$;
- $(X, m_2) \Longrightarrow_{MU}^{T,\rho} (m_2, \{(X, m_2)\})$ with $X \in \mathcal{X}$;
- $(!X, m_2) \Longrightarrow_{MU}^{T,\rho} (m_2, \emptyset)$ if $\rho(X) = m_2$;
- $(\{m_1\}_K, \{m_2\}_K) \Longrightarrow_{MU}^{T,\rho} (\{m_u\}_K, \sigma_u)$ if $K \in T \cup \mathcal{PK}$ and $(m_1, m_2) \Longrightarrow_{MU}^{T,\rho} (m_u, \sigma_u)$;
- $(Sign_{id}(m_1), Sign_{id}(m_2)) \Longrightarrow_{MU}^{T,\rho} (Sign_{id}(m_u), \sigma_u)$ if $(m_1, m_2) \Longrightarrow_{MU}^{T,\rho} (m_u, \sigma_u)$;
- $((m_1, m_2), (m_3, m_4)) \Longrightarrow_{MU}^{T,\rho} ((m'_u, m''_u), \sigma'_u \cup \sigma''_u)$ if $(m_1, m_3) \Longrightarrow_{MU}^{T,\rho} (m'_u, \sigma'_u)$, $(m_2, m_4) \Longrightarrow_{MU}^{T,\rho} (m''_u, \sigma''_u)$ and $\sigma'_u \cup \sigma''_u$ is a partial function.

The parameter T in the unification relation is the set of private keys which are supposed to be known when the messages are unified. For instance, a message $\{X\}_K$ can be unified with a message $\{m\}_K$ only if either K is a public key or K is a known private key. (Notice that this restriction prevents from disclosing an encrypted message by unification.) For the same reason, a message $h(X)$ cannot be unified with a message $h(m)$. The condition for the unification of a pair of messages ensures that two different actual values are

not assigned to the same formal parameter.

Performing transitions is not, in general, instantaneous. A transition t_1 labelled by a trigger $?\alpha(m)$ and an action $!\beta(m')$ can be performed when a synchronization with a transition t_2 (in a parallel component) labelled by an action $!\alpha(m'')$ is possible. The synchronization is instantaneous and releases immediately t_2. On the other hand, the completion of the enabled transition t_1, may be deferred by two factors which contributes to the overall duration of the transition:

(i) t_1 may be forced to wait the completion of the transmission of message m'' which takes a time belonging to the interval $\lambda(\alpha)$;

(ii) a synchronization is required for the action $!\beta(m')$.

A consequence of such a semantics for transitions, is that a transition labelled by a trigger $?\alpha(m)$ and an action $!\beta(m')$ can be equivalently replaced by the sequentialization of two transitions, the former labelled by $?\alpha(m)$ with a Nil action and the latter labelled by a trigger $True$ and action $!\beta(m')$. For the sake of simplicity, in the following we shall consider sequential components having transitions labelled in such a restricted way.

More formally, a transition is *unidirectional* if it has one of the following forms:

- $\langle s, \gamma, Nil, (s_1, \tau_1), (s_2, \tau_2), s' \rangle$, for some $\gamma \in Trigger(\Gamma)$ (a *receiving transition*);
- $\langle s, True, \beta, (s, 0), (s, \omega), s' \rangle$ for some $\beta \in Action(\Gamma)$ (a *sending transition*).

It is easy to transform a MTA into a MTA having unidirectional transitions.

Given a sequential component $C = \langle S, I, \delta, Kw \rangle$, we denote by $Unidir(C)$ the sequential machine (with unidirectional transitions) $\langle S_0 \cup S_1, I_0, \overline{\delta}, \overline{Kw} \rangle$, where

- $S_i = \{(s, i) : s \in S, \text{ with } i \in \{0, 1\}\}$ are two disjoint copies of S;
- $I_0 = \{(s, 0) : s \in I\}$;
- $\overline{\delta} = \{\langle (s,0), \gamma, Nil, ((s_1,0), \tau'), ((s_2,0), \tau''), (s,1) \rangle,$
  $\langle (s,1), True, \beta, ((s,1),0), ((s,1), \omega), (s',0) \rangle :$
  $\langle s, \gamma, \beta, (s_1, \tau'), (s_2, \tau''), s' \rangle \in \delta\}$;
- $\overline{Kw} = \{((s,0), A)) : (s, A) \in Kw\}$.

For a MTA $C = \langle C_1, \ldots, C_n, \lambda, \eta \rangle$, $Unidir(C)$ is the automaton $\langle Unidir(C_1), \ldots, Unidir(C_n), \lambda, \eta \rangle$.

3.2 The MTA for the Non-Repudiation Protocol

In this subsection we present the MTA describing the the protocol presented in the previous section. The MTA consists of three sequential components, one for each participant, depicted in Figure 1, 2, and 3.

For description pourposes, triggers and actions of transitions are decorated by labels o the form A_i (for Alice in Figure 1) B_i (for Bob in Figure 2) and T_i (for TTP in Figure 3). Null delays and unbounded timeouts are omitted, whereas a label $\leq T$ stands for a timeout (s_0, T), with s_0 the initial state. The channel α connects Alice and Bob, β connects Alice to the TTP, and γ connects Bob to the TTP. We assume that channels are resilient, that is, the timing channel function λ is such that $\lambda(\alpha) = \lambda(\beta) = \lambda(\gamma) = (0, \omega)$. Moreover, we assume that the identity function η assigns A to Alice, B to Bob and TT to the TTP. Symbols M, L, T and all the flags of the form f_n (with n the flag name) are messages; K is a key of A and C is the structured message $\{M\}_K$; X, V, Y, Z, W, and H are formal parameters.

Let us consider in more detail the sequential component for Alice in Figure 1. Alice starts the protocol execution by sending the evidence EOO (notice that the trigger A_1 is $True$). The structured message corresponding to EOO is $sign_A(f_{EOO}, B, L, T, C)$ (for the sake of simplicity we forget pairing and, for instance, we write f_{EOO}, B, L, C instead of $(f_{EOO}, (B, (L, (T, C)))))$. With reference to Figure 2, Bob is waiting for the corresponding parametric message labelled by B_1. Notice that the ground messages L and C sent by Alice will be unified with the formal parameters X and Y, respectively.

The list of the other shorthands used in Figure 1, is the following:

EOR has the form $sign_B(f_{EOR}, A, L, T, C)$;
EOO_K has the form $sign_A(f_{EOO_K}, B, L, K)$;
EOR_K has the form $sign_B(f_{EOR_K}, A, L, K)$;
sub_K has the form $sign_A(f_{sub\_K}, B, L, K)$;
con_K has the form $sign_{TT}(f_{CON}, B, L, K)$.

According to the protocol description, from now on all the steps of Alice have to complete within time T. Once she has received from Bob the first part of the aimed evidence (EOR), she sends the last part of the evidence (EOO_K) to Bob.

Alice has then to wait for the last part of evidence (EOR_K). If this evidence is not delivered (either because Bob did not send it or because the channel is delaying the message too much), she has to invoke the TTP within the given

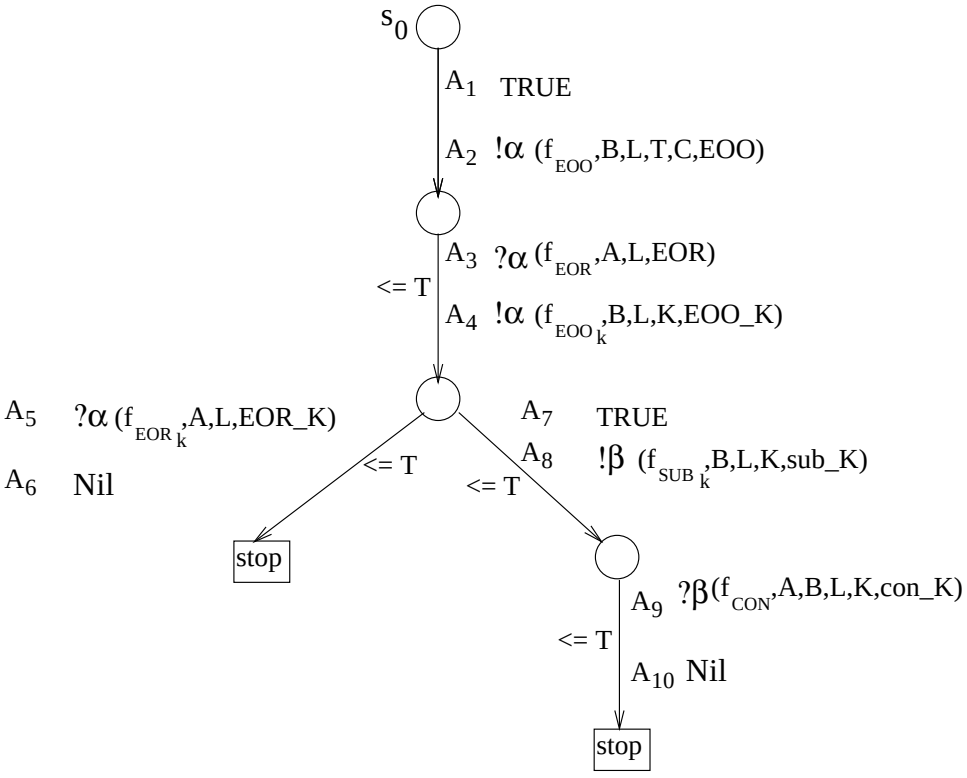

Fig. 1. The sequential component for Alice

time T, chosen in advance before starting the protocol. This is represented by a branch of the sequential component: on the left branch Alice waits for the *EOR_K*, on the right one she invokes the intervention of the TTP within time T.

The sequential components for Bob and the TTP (see Figures 2 and 3) are simpler. We only observe that the sequential component of Bob has a branching point dealing with the situation in which he may receive either *EOO_K* from Alice or *con_K* from the TTP, or both.

The list of shorthands used in the sequential components for Bob and the TTP are the following:

EOO -B has the form $sign_A(f_{EOO}, B, X, T, Y)$;
EOR -B has the form $sign_B(f_{EOR}, A, X, T, Y)$;
EOO_K -B has the form $sign_A(f_{EOO_K}, B, X, W)$;
EOR_K -B has the form $sign_B(f_{EOR_K}, A, X, W)$;
$con\_K - B$ has the form $sign_{TT}(f_{CON}, A, B, !X, W)$;

$sub\_K - T$ has the form $sign_A(f_{sub\_K}, B, V, H)$;

$con\_K - T$ has the form $sign_{TT}(f_{CON}, A, B, V, H)$.

Notice that the sequential component for Bob receives the identifier of the protocol run L in the parameter X in the trigger B_1, and then forces Alice to use the same identifier in the following steps by exploiting $!X$ in the trigger B_3 and B_4.

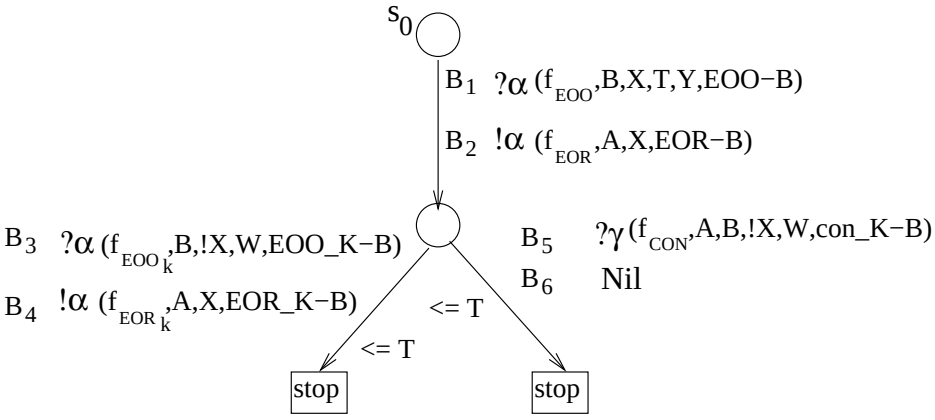

Fig. 2. The sequential component for Bob

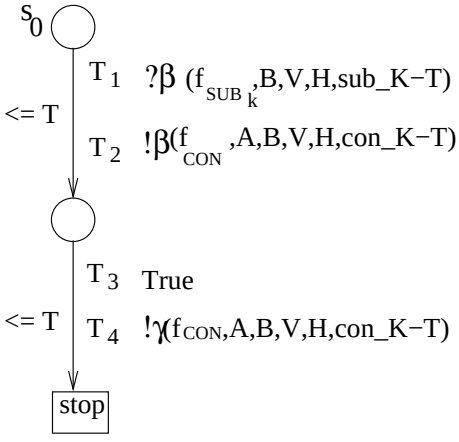

Fig. 3. The sequential component for TTP

4 From MTA to Timed Automata

The semantics of MTA's is given by translation into the well known setting of Timed Automata [2]. The translation is performed in two steps: we first translate a sequential component into a (open) Timed Automaton with transition labelled by symbols for incomplete communications; then, we take the product of the so obtained sequential components and we synchronize incomplete communications. In general, the resulting translation is not guaranteed to be a Timed Automaton since the set of locations resulting from our translation may be infinite. Fortunately, in the cases of interest it is easy to find syntactical constraints which are sufficient to guarantee finiteness of locations. For instance, a sufficient condition which could be naturally enforced is that each cycle in sequential component should contain at least an input state.

For sake of completeness, we start defining Timed Automata and their semantics.

Definition 4.1 A *Timed Automaton* over an alphabet $\mathcal{S}$ is a tuple

$$\langle L, L_0, CK, I, \delta \rangle, \text{ where}$$

- L is a finite set of *locations*;
- $L_0 \subseteq L$ is a set of *initial* locations;
- CK is a finite set of *clocks*;
- $I : L \to \Phi(CK)$ is the *invariant map* associating a *clock constraint* with each location, where the set of clock constraints $\Phi(CK)$ is defined by the following grammar:

$$\phi := True \mid z \leq c \mid c \leq z \mid z < c \mid c < z \mid \phi_1 \wedge \phi_2,$$

  with z a clock in CK and c a constant in $\mathcal{Q}^{\geq 0}$;
- $\delta \subseteq L \times \mathcal{S} \times \Phi(CK) \times 2^{CK} \times L$ is the *transition relation*.

An element $\langle s, a, \phi, \lambda, s' \rangle \in \delta$, written also $s \xrightarrow{a,\phi,\lambda} s'$, represents a transition from the location s to the location s' on symbol a; the clock constraints ϕ specifies when the transition is enabled, and the set $\lambda \subseteq CK$ gives the clocks to be reset when the transition is performed.

We recall now, rather informally, the semantics of Timed Automata refering the reader to the literature for the standard definition.

The semantics of a Timed Automaton $A = \langle L, L_0, CK, I, \delta \rangle$, is defined by associating a transition system TS_A. A state of TS_A is a pair $\langle s, \nu \rangle$ such that s is a location of A and ν is a clock valuation of CK (i.e. a mapping from

CK to $\mathcal{Q}^{\geq 0}$) such that ν satisfies the invariant $I(s)$. A state is an initial state if s is an initial location and $\nu(z) = 0$, for all clocks z. There are two types of transitions in TS_A:

State change due to elapse of time: for a state $\langle s, \nu\rangle$ and a time increment $\gamma \in \mathcal{Q}^{\geq 0}$, $\langle s, \nu\rangle \xrightarrow{\gamma} \langle s, \nu + \gamma\rangle$, if for all $0 \leq \gamma' \leq \gamma$, $\nu + \gamma'$ satisfies the invariant $I(s)$ ($\nu + \gamma$ denotes the clock valuation which maps each clock z to $\nu(z) + \gamma$);

State change due to a location transition: for a state $\langle s, \nu\rangle$ and a transition $\langle s, a, \phi, \lambda, s'\rangle$ such that ν satisfies ϕ, $\langle s, \nu\rangle \xrightarrow{a} \langle s, \nu[\lambda := 0]\rangle$ ($\nu[\lambda := 0]$ denotes the clock valuation which assigns 0 to each $z \in \lambda$ and agrees with ν over the rest of the clocks).

In the following we assume that MTA's have unidirectional transitions.

For a map $\rho : \mathcal{X} \to SM_{\Sigma,\emptyset}$, we denote by $\hat{\rho}$ the extension of the map ρ to elements of $SM_{\Sigma,\mathcal{X}}$: for an element $m \in SM_{\Sigma,\mathcal{X}}$, $\hat{\rho}(m)$ gives the simultaneous replacement in m of any occurrence of a submessage of the form $!X$ or X by $\rho(X)$.

For a sequential component $C_i = \langle S, I, \delta, Kw\rangle$, of a MTA $\langle C_1, \ldots, C_n, \lambda, \eta\rangle$, we define a Timed Component $TC(C_i) = \langle TS, TS_I, CK, TI, T\delta\rangle$ (i.e. a Timed Automaton), where TS is the set of locations, TS_I is the set of initial locations, CK is the set of clocks, TI is the invariant fuction, and $T\delta$ is the transition relation.

A location in TS is a tuple $\langle KwS, Ch, s, \rho\rangle$, where

- KwS is the collection of evidences (i.e. ground messages) known in that location;
- Ch keeps trace of the set of messages the component is receiving from communications which have been established but which are not yet completed (due to transmission duration); each incomplete interaction is described by the channel name, the (parametric) expected message and the ground sent message; the pair of expected and sent messages allows to bind formal parameters with messages at the transmission completion;
- $s \in S$ is the current state of the component C_i;
- ρ is a partial map which binds parameters to messages.

More formally, the timed component $TC(C_i)$ is as follows:

(i) $TS = \{\langle KwS, Ch, s, \rho\rangle : KwS \subseteq SM_{\Sigma,\emptyset},\ Ch \subseteq \Gamma \times SM_{\Sigma,\chi} \times SM_{\Sigma,\emptyset},\ s \in S, \rho \subseteq \mathcal{X} \times SM_{\Sigma,\emptyset}\}$;

(ii) $TS_I = \{\langle Kw(s), \emptyset, s, \emptyset\rangle : s \in I\}$;

(iii) $CK = \{ck_s : s \in S\} \cup \{ck_{\eta(i),m} : m \in \mathcal{M}\} \cup \{ck_{\eta(i),\alpha} : \alpha \in \Gamma\}$;

(iv) TI is such that $TI(\langle KwS, Ch, s, \rho\rangle) = \phi_1 \wedge \phi_2 \wedge \phi_3$ with
  (a) ϕ_1 is the conjunction of the set of clock constraints $\{True\} \cup \{ck_s \leq \tau : ck_s \leq \tau$ occurs in the clock constraint of a transition $t \in T\delta$ departing from location $\langle KwS, Ch, s, \rho\rangle\}$.
  (b) ϕ_2 is the conjunction of the set of clock constraints $\{True\} \cup \{ck_{\eta(i),m} \leq \tau :$ there is $m' \in KwS \cup \pi_3(Ch)$ with an occurrence of a message of the form $\Delta_\tau(m)$ or $\Theta_\tau(m)$ or $I_\tau^{\tau'}(m)\}$ (π_j is the extension to sets of the standard projection function along the j-th component);
  (c) ϕ_3 is the conjunction of the set of clock constraints $\{True\} \cup \{ck_{\eta(i),\alpha} \leq \tau_2 : \lambda(\alpha) = (\tau_1, \tau_2), \alpha \in \pi_1(Ch)\}$;

(v) $T\delta$ is the union of the following sets:
  (a) $\{\langle KwS, Ch, s, \rho\rangle \stackrel{!\alpha(\hat\rho(m)),True,\{ck_{s'}\}}{\longrightarrow} \langle KwS, Ch, s', \rho\rangle :$ $\langle s, True, !\alpha(m), (s,0), (s,\omega), s'\rangle \in \delta$, $s' \notin I$, $\hat\rho(m)$ is a ground message and $KwS \vdash_{\eta(i)} \hat\rho(m)\}$;
  (b) $\{\langle KwS, Ch, s, \rho\rangle \stackrel{!\alpha(\hat\rho(m)),True,\{ck_{s'}\}}{\longrightarrow} \langle KwS', \emptyset, s', \emptyset\rangle :$ $\langle s, True, !\alpha(m), (s,0), (s,\omega), s'\rangle \in \delta$, $KwS' = Kw(s')$, $s' \in I$, $\hat\rho(m)$ is a ground message and $KwS \vdash \hat\rho(m)\}$;
  (c) $\{\langle KwS, Ch, s, \rho\rangle \stackrel{Nil,True,\{ck_{s'}\}}{\longrightarrow} \langle KwS, Ch, s', \rho\rangle :$ $\langle s, True, Nil, (s,0), (s,\omega), s'\rangle \in \delta$, $s' \notin I$ $\}$;
  (d) $\{\langle KwS, Ch, s, \rho\rangle \stackrel{Nil,True,\{ck_{s'}\}}{\longrightarrow} \langle KwS', \emptyset, s', \emptyset\rangle :$ $\langle s, True, Nil, (s,0), (s,\omega), s'\rangle \in \delta$, $KwS' = Kw(s')$, $s' \in I$ $\}$;
  (e) $\{\langle KwS, Ch, s, \rho\rangle \stackrel{?\alpha(m),\phi_\Delta\wedge\phi_\Theta,Clocks}{\longrightarrow} \langle KwS, Ch', s', \rho\rangle :$ $\langle s, ?\alpha(m), Nil, (s_1,\tau_1), (s_2,\tau_2), s'\rangle \in \delta$, $Ch' = Ch \cup \{\langle \alpha, m, m'\rangle\}$, $s' \notin I$, there is no tuple t in Ch having α as first component, $(m, m') \Longrightarrow_{MU}^{T,\rho} (m_u, \rho_u)$ with $T = \{K \in \mathcal{K} : KwS \vdash_{\eta(i)} K\}$, there is no timed atomic submessage of m_u occuring timed in KwS, $\phi_\Delta = ck_{s_1} \geq \tau_1$, $\phi_\Theta = True$ if $\tau_2 = \omega$ and $\phi_\Theta = ck_{s_2} \leq \tau_2$ otherwise, $Clocks = \{ck_{s'}, ck_{\eta(i),\alpha}\} \cup \{ck_{\eta(i),\overline{m}} : \overline{m} \in \mathcal{M}$ is a timed submessage of $m'\}$ $\}$
  (f) $\{\langle KwS, Ch, s, \rho\rangle \stackrel{?\alpha(m),\phi_\Delta\wedge\phi_\Theta,\{ck_{s'}\}}{\longrightarrow} \langle KwS', \emptyset, s', \emptyset\rangle :$ $\langle s, ?\alpha(m), Nil, (s_1,\tau_1), (s_2,\tau_2), s'\rangle \in \delta$, $s' \in I$, there is no tuple t in Ch having α as first component, $(m, m') \Longrightarrow_{MU}^{T,\rho} (m_u, \rho_u)$ with $T = \{K \in \mathcal{K} : KwS \vdash_{\eta(i)} K\}$, there is no timed atomic submessage of m_u occuring timed in KwS; $KwS' = Kw(s')$, $\phi_\Delta = ck_{s_1} \geq \tau_1$, $\phi_\Theta = True$ if $\tau_2 = \omega$ and $\phi_\Theta = ck_{s_2} \leq \tau_2$ otherwise $\}$
  (g) $\{\langle KwS, Ch, s, \rho\rangle \stackrel{True,\phi_\Delta\wedge\phi_\Theta,\{ck_{s'}\}}{\longrightarrow} \langle KwS, Ch, s', \rho\rangle :$

$\langle s, True, Nil, (s_1,\tau_1), (s_2,\tau_2), s'\rangle \in \delta,\ s' \notin I,$
$\phi_\Delta = ck_{s_1} \geq \tau_1,\ \phi_\Theta = True$ if $\tau_2 = \omega$ and $\phi_\Theta = ck_{s_2} \leq \tau_2$ otherwise $\}$

(h) $\{\langle KwS, Ch, s, \rho\rangle \overset{True,\phi_\Delta\wedge\phi_\Theta,\{ck_{s'}\}}{\longrightarrow} \langle KwS', \emptyset, s', \emptyset\rangle :$
$\langle s, True, Nil, (s_1,\tau_1), (s_2,\tau_2), s'\rangle \in \delta,\ s' \in I,\ KwS' = Kw(s'),$
$\phi_\Delta = ck_{s_1} \geq \tau_1,\ \phi_\Theta = True$ if $\tau_2 = \omega$ and $\phi_\Theta = ck_{s_2} \leq \tau_2$ otherwise $\}$

(i) $\{\langle KwS, Ch, s, \rho\rangle \overset{\epsilon,\phi,\emptyset}{\longrightarrow} \langle KwS', Ch', s, \rho'\rangle :$
$\phi = \tau_1 \leq ck_{\eta(i),\alpha} \leq \tau_2$ if $\lambda(\alpha) = (\tau_1,\tau_2)$ and $\phi = \tau_1 \leq ck_{\eta(i),\alpha}$ if $\lambda(\alpha) = (\tau_1,\omega)$,
$\langle \alpha, m, m'\rangle \in Ch,\ KwS' = KwS \cup \{m_u\},$
with $(m,m') \Longrightarrow_{MU}^{T,\rho} (m_u,\rho_u)$ and $T = \{K \in \mathcal{K} : KwS \vdash_{\eta(i)} K\},$
$Ch' = Ch \setminus \{\langle \alpha, m, m'\rangle\},$
$\rho' = \rho_u \cup \{(X, \overline{m}) : (X,\overline{m}) \in \rho, \rho_u \text{ is not defined for } X\}\ \},$

(j) $\{\langle KwS, Ch, s, \rho\rangle \overset{\epsilon,ck_{\eta(i),m'}=\tau,\emptyset}{\longrightarrow} \langle KwS', Ch, s, \rho\rangle :$
$KwS' = KwS \setminus \{m\} \cup \{m[m'|_{\Delta_\tau(m')}, \Theta_{\tau'}(m')|_{I_\tau^{\tau'}(m')}]\},$ for some
$m \in KwS$ with m having a submessage either of the form $\Delta_\tau(m')$ or $I_\tau^{\tau'}(m')$ $\}$

(k) $\{\langle KwS, Ch, s, \rho\rangle \overset{\epsilon,ck_{\eta(i),m'}=\tau,\emptyset}{\longrightarrow} \langle KwS', Ch, s, \rho\rangle :$
$KwS' = KwS \setminus \{m\} \cup \{m[\dagger(m')|_{\Theta_\tau(m')}]\},$ for some $m \in KwS$ with m having a submessage of the form $\Theta_\tau(m')$ $\}$

(l) $\{\langle KwS, Ch, s, \rho\rangle \overset{\epsilon,ck_{\eta(i),m'}=\tau,\emptyset}{\longrightarrow} \langle KwS, Ch', s, \rho\rangle :$
$Ch' = Ch \setminus \{\langle \alpha, m'', m\rangle\} \cup \{\langle \alpha, m'', m[m'|_{\Delta_\tau(m')}, \Theta_{\tau'}(m')|_{I_\tau^{\tau'}(m')}]\rangle\},$ for some $\langle \alpha, m'', m\rangle \in Ch$ with m having a submessage either of the form $\Delta_\tau(m')$ or $I_\tau^{\tau'}(m')$ $\}$

(m) $\{\langle KwS, Ch, s, \rho\rangle \overset{\epsilon,ck_{\eta(i),m'}=\tau,\emptyset}{\longrightarrow} \langle KwS, Ch', s, \rho\rangle :$
$Ch' = Ch \setminus \{\langle \alpha, m'', m\rangle\} \cup \{\langle \alpha, m'', m[\dagger(m')|_{\Theta_\tau(m')}]\rangle\},$ for some $\langle \alpha, m'', m\rangle \in Ch$ with m having a submessage of the form $\Theta_\tau(m')$ $\}$.

The set of clocks CK of the component provides a distinct clock name for each state in S, each ground message and each channel name. With reference to the transition relation $T\delta$ we have:

a is the set of transitions corrisponding to a sending action and leading to a non input state; the action of sending does not affect the collection of evidences and the binding of parameters; notice that only ground (derivable) evidences can be sent;

b is the set of transitions corresponding to a sending action and leading to an input state; the collection of evidences and the binding of parameters is reset; moreover, incomplete messages are lost;

c-d are the set of transitions corresponding to a Nil action leading to a non

input state and input state, respectively;

e is the set of transitions corresponding to a receiving action (leading to a non input state); for the sake of compositionality, the set includes a transition for each ground message m' unifiable with m provided that m' is not sent in a busy channel (i.e. still involved in an incomplete communication) and m' does not retime timed ground messages already belonging to the collection of evidences; since a communication might take time, the received messages is added to the compomente Ch of the location unaffecting the collection of evidences or the binding of variables; ϕ_Δ and ϕ_Θ translates the delay and timeout, respectively, of the sequential component transition into suitable conditions on clocks;

f is the set of transitions corresponding to a receiving action leading to an input state;

g-h is the set of transitions corresponding to a $True$ triggered transition leading to a non input state and input state, respectively;

i is the set of transitions representing the completion of the transmission of a message (the transition checks the clock associated with the transmission channel α against the expected transmission duration of the channel given by $\lambda(\alpha)$); the location is altered by removing the message from the component Ch and by updating accordingly the set of evidences KwS;

j is the set of transitions which handle the disclosure of a (sub)message of the form $\Delta_\tau(m)$ or $I_\tau^{\tau'}(m)$; after the disclosure, m belongs to the set of (derived) evidences;

k is the set of transitions which handle the expiration of a (sub)message of the form $\Theta_\tau(m)$; after the expiration, m is replaced in the set of (derived) evidences by $\dagger(m)$;

l-m are the sets of transitions which handle the disclosure and expiration, respectively, of messagges in Ch (i.e messages whose transmission has not yet been completed).

Moreover, notice that location invariants ensure that completion of communication, disclosure and expiration of messages are not delayed beyond their specified timeout.

Let us consider now the translation of a MTA $G = \langle C_1, \ldots, C_n, \lambda, \eta \rangle$.

Assuming that $TC(C_i) = \langle TS_i, TS_{Ii}, CK_i, TI_i, T\delta_i \rangle$, the Timed Automaton associated with G is

$$TG = \langle TQ, TQ_0, CK, TI, T\delta \rangle, \text{ where}$$

(i) $TQ = \{\langle ts_1, \ldots, ts_n \rangle : ts_i \in TS_i, 1 \leq i \leq n\}$;

(ii) $TQ_0 = \{\langle ts_1, \ldots, ts_n\rangle : ts_i \in TS_{Ii}, 1 \leq i \leq n\}$;

(iii) $CK = \bigcup_{i=1}^{n} CK_i$;

(iv) TI is such that $TI(\langle ts_1, \ldots, ts_n\rangle) = \bigwedge_{i=1}^{n}(TI_i(ts_i))$;

(v) $T\delta$ is the union of the following sets of transitions:

(a) $\{\langle ts_1, \ldots, ts_n\rangle \xrightarrow{\alpha,\phi_1\wedge\phi_2,\beta_1\cup\beta_2} \langle ts'_1, \ldots, ts'_n\rangle$: there are i and j $(1 \leq i, j \leq n)$ such that $ts_i = \langle KwS_i, Ch_i, s_i, \rho_i\rangle$ and $ts_j = \langle KwS_j, Ch_j, s_j, \rho_j\rangle$, there are transitions $ts_i \xrightarrow{?\alpha(m),\phi_i,\beta_i} ts'_i$ and $ts_j \xrightarrow{!\alpha(m'),\phi_j,\beta_j} ts'_j$, $ts'_i = \langle KwS'_i, Ch'_i, s'_i, \rho'_i\rangle$ with $\langle \alpha, m, m', \rangle \in Ch'_i$, and $ts_k = ts'_k$, for any $k \neq i$ and $k \neq j\}$;

(b) $\{\langle ts_1, \ldots, ts_n\rangle \xrightarrow{\epsilon,\phi_i,\lambda_i} \langle ts'_1, \ldots, ts'_n\rangle$: there is i such that $ts_i \xrightarrow{a,\phi_i,\gamma_i} ts'_i$, with $a \in \{\epsilon, True, Nil\}$ and $ts_k = ts'_k$, for any $k \neq i$.

5 Verification

In this section we present the translation of the MTA for the considered Non-Repudiation Protocol into the corresponding Timed Automaton, and then we show how the fairness property can be checked on the Timed Automaton itself. With reference to Figure 4, in each node-location l_i, with $1 \leq i \leq 6$, the *knowledge* of the participants is denoted by kw_P^j, where $P \in \{A, B\}$ and j is used to remark the change (in particular the increase) of the knowledge of P. Note that this knowledge increases after any occurences of a *receive* action (labelled by a question mark on the incoming edge). The timed automaton is developed according to the matchings of the actions in the MTA: l_1 is the starting location, then Alice sends the commitment C to Bob (action A_2) yielding to l_2 and so on until localtion l_6 is reached. At this point the designers of the protocol have given two options, either the *optimistic* one in which Alice and Bob cooperates, there are no delays on the involved channels and both get their own evidence or something goes wrong (a channel delay or a participant is dishonest) and the TTP must intervene. Here is where our tool plays a crucial role, as these two scenarios may interleave leading to an undesired or better, an unpredictable run of the protocol (see Figure 5 where we have exploited the labels introduced in the specification of the protogol given in Figure 1, 2 and 3). Note that the triggers/actions $A_8 < T_1 < T_2$ may interleave with $B_3 < B_4$, where the symbol $<$ is a precedence relation over the set of triggers and actions, (that is for example meaning that A_8 must occur before T_1). In the figure we have omitted the locations where the protocol halts for simplicity reasons. Actually, we have drawn only one of these locations (l_{11}), namely the one indicating that both Alice and Bob have received the evidences from the TTP. The dotted edges indicates that, from

that point on, A_5, B_5 and A_9 may interleave: for instance, Alice may receive *EOR_K* from Bob while Bob is receiving *con_K* from the TTP.

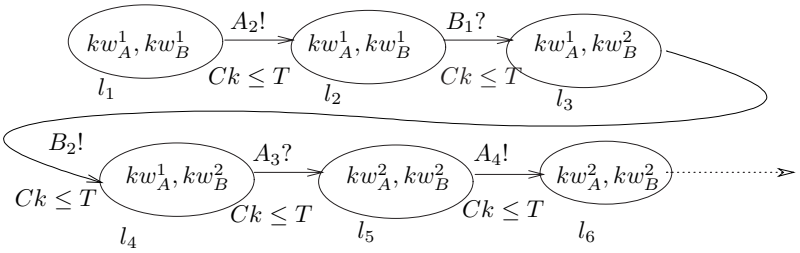

Fig. 4. The first part of the resulting Timed Automata

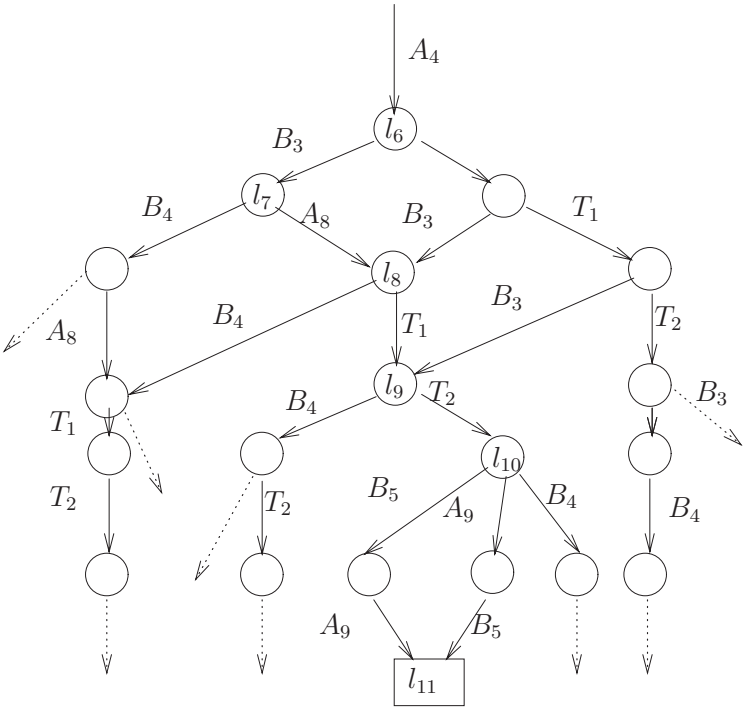

Fig. 5. The last part of the resulting Timed Automata

Verification of the fairness property. We have modelled a protocol by a Timed Automaton. A run of the protocol is a (finite) sequence of states of the TA, where each state is determined by both a location and the current values of the clocks. If we label the locations with sets of atomic propositions,

we can check properties of the runs, expressed as (T)CTL formulas. In the example of non-repudation protocol we have considered, we are interested in verifying whether the fairness property is satisfied. Recall from section 2 that the *fairness* property requires that no party can get any advantage over the other: if one receives the desired evidence (either the evidence of the origin or that one of the receipt), then eventually the other party receives her/his evidence, too. The atomic propositions (AP) mark whether a participant in a given location has got a fragment of his/her evidence:

$AP = \{EOR\,,\ EOR\_K\,,\ EOO\,,\ EOO\_K\,,\ CONA\_K\,,\ CONB\_K\ \}.$

Each location is labelled with those atomic propositions which evaluates to true in it and inherits the propositions from its ancestors (note that the TA is a dag). For example, the location l_2 and the successive locations are labelled by EOO, denoting that Bob has got the evidence of the origin, the location l_4 and its successors are labelled by EOR, the successors of l_7 are labelled by EOO_K, and those of l_{11} are labelled by $CONA\_K$. Note that this labelling can be automatically deducted from the knowledge of the participants.

Verifying the fairness property for Alice amounts to check the following CTL formula:

$$\forall\Box(NRO \implies \forall\Diamond NRR\ ,\ \text{where}$$

$NRO\ = EOO\ \wedge (EOO\_K\ \vee CONB\_K\ )$ and
$NRR\ = EOR\ \wedge (EOR\_K\ \vee CONA\_K\ )$

This formula expresses the requirement that, for each run of the protocol, whenever Bob gets his evidence, constituted by NRO, then eventually Alice gets her evidence NRR. Viceversa, the fairness property for Bob is given by the following formula, expressing the requirement that if Alice gets her evidence, NRR, then eventually Bob gets his evidence NRO:

$$\forall\Box(NRR \implies \forall\Diamond NRO\,).$$

To check fairness property for Alice assume, for simplicity, that she has an *honest* behaviour: upon the receipt of a message, she immediately sends the message according to the protocol specification. Bob, instead, may delay sending the message. Clearly, the TTP acts immediately upon the receipt of the request of its intervention. As regards the channels, let us make first the somewhat realistic assumption that all channels are resilient: there is an unknown finite bound on the time elapsing between the sending and the receiving of a message on a channel. Under this hypothesis the fairness property is not ensured to hold. Let us see how our setting automatically detects this drawback.

Consider a run going through the location l_7, taking the transition A_8 at time $T - d$ for some $d > 0$, and getting stuck in l_8 for a period of time greater than d (this is possible since the channel β is resilient and, moreover, Bob may delay the action B_4). The run then stops, since no further transition can be taken (recall that the transitions are triggered by the time constraint $Ck \leq T$). In this run the fairness property is not satisfied as *NRO* holds in a state with location l_7, whatever values the clocks have, but there are no states in the run satisfying *NRR*, since no locations labelled by *EOR_K* or *CONA_K* are reachable from l_7 (see Figure 5).

Assume now that channels β and γ are operational and let δ be the maximum delay for β (the maximum time a message can take to be transmitted over β). The timed automaton obtained from the specification of the protocol turns out to be slightly different, as there are now time constraints in some locations (for example in l_8) forcing the run to exit within δ time units. It is easy see that also with this modification there is a run for which the fairness property does not hold: consider the transition A_8 from l_7 taken at a time $T - d$, for some $d < \delta$, and let a run idle in l_8 for a time d^R, with $d < d^R < \delta$. Similar arguments can be used for channel β.

Thus, we can conclude that in both cases the given specification leads to unfair behaviours, even when an operational channel is considered to link both Alice and Bob to the TTP. In a correct specification the actions A_8 and A_4 must be triggered within a time-out of $T - 2\delta$. In this way a run going through l_8 eventually reaches a location labelled by *EOR_K* or *CONA_K* . Let us remark that these modifications on the protocol are sufficient also to ensure the fairness property for Bob.

6 Conclusions

In this paper we have introduced a simple, graphical specification formalism to specify security protocols. There are two main advantages of this approach. First, the specification language is very close to the specification style of the protocol designer and allows the explicit representations of protocol parties and communication among parties. Moreover, a protocol specified with this formalism can be automatically translated into a Timed Automaton, in such a way that a security property, expressed by formulas in a temporal logic, can be checked using standard model checking techniques.

In the non-repudiation protocol considered in this paper, an adversary can be seen as a *dishonest* party. Some kinds of dishonest behaviour of one party can be modelled by an augmentation phase (implemented as preprocessing) before the translation from the MTA into a resulting TA. For instance, a

dishonest behaviour could be the one in which each party can take any action of the protocol whenever he/she wants, disregarding the order dictated by the protocol.
It is our intention to enrich in the future the specification language in such a way that an adversary can be taken into consideration in any other kinds of protocols.

In Timed Automata, transitions are constrained with timing requirements where only numerical constants can appear. With Parametric Timed Automata (see [1]), whose transitions are constrained with parametric timing requirements, more realistic situations can be described. We claim that this model is attractive also when security protocol are dealt with and thus the translation of our specification language into a subclass of Parametric Timed Automata can be an interesting further extension of our work.

References

[1] R. Alur, T. Henzinger, M. Vardi, Parametric real-time reasoning, STOC 1993, pp.592-601.

[2] R. Alur, D. Dill, A theory of timed automata, Theoretical Computer Science, 126, pp. 183-235, 1994.

[3] R. Barbuti, N. De Francesco, A. Santone, L. Tesei, A Notion of Non-Interference for Timed Automata, Fundamente Informaticae ,51 2003 pp. 1 -11.

[4] C. Daws, A. Olivero, S. Tripakis, S. Yovine, The tool KRONOS, In Hybrid Systems III: Verification and Control, LNCS 1066, pp. 208-219, 1996.

[5] R. Gorrieri E. Locatelli, F. Martinelli, A simple Language for Real Time Cryptographic Protocol Analysis, ESOP 2003, LNCS 2618, pp. 114-128, 2003-03-27

[6] R. Lanotte, A. Maggiolo-Schettini, S. Tini, Timed Information Flow for Timed Automata, submitted.

[7] K. Larsen, P. Petterson, W. Yi, UPPAL in a nutshell, Springer International Journal of Software Tools for Technology Transfer, 1, 1997.

[8] C. Meadows, Formal methods for cryptographic protocol analysis: emerging issues and trends. IEEE Journal On Selected Area in Communications, 21, 2003

[9] J. Zhou, D. Gollmann, An Efficient Non-repudiation Protocol, 10-th Computer Security Foundation Workshop (CSFW'97), Rockport, Massachusets, USA, 126–132, 1997.