

# Formal analysis of some timed security properties in wireless protocols <sup>★</sup>

Roberto Gorrieri<sup>1</sup>, Fabio Martinelli<sup>2</sup>, Marinella Petrocchi<sup>2</sup>, and Anna Vaccarelli<sup>2</sup>

<sup>1</sup> Dipartimento di Scienze dell'Informazione, Università di Bologna  
e-mail: gorrieri@cs.unibo.it

<sup>2</sup> Istituto di Informatica e Telematica - C.N.R.  
e-mail: {fabio.martinelli, marinella.petrocchi, anna.vaccarelli}@iit.cnr.it

**Abstract.** We show how a recent language for the description of cryptographic protocols in a real time setting may be suitable to formally verify security aspects of wireless protocols. We define also a compositional proof rule for establishing security properties of such protocols. The effectiveness of our approach is shown by defining and studying the timed integrity property for  $\mu$ TESLA, a well-known protocol for wireless sensor networks. We are able to deal with protocol specifications with an arbitrary number of agents (senders as well as receivers) running the protocol.

**Key words:** Security, Wireless Communication, Formal Analysis, Sensor Networks.

## 1 Introduction

Ambient intelligence is one of the main research issues in Europe. It imposes a view of the world where objects may interact with each other by means of wireless communications. In our framework, we are interested in developing formal analysis techniques for such scenarios. In particular, we aim at studying security properties of wireless networks and we start with the analysis of a secure wireless protocol for sensor networks where time plays an essential role.

Wireless systems consisting of mobile nodes self-organizing in temporary routing topologies form wireless *ad hoc* networks. Wireless *ad hoc* networks may cover various types of applications. Peculiarity in their usage is when wired infrastructure is not available at all (a typical example involves rescue operations in remote areas or disaster recovery operations). Various issues in the world of the *ad hoc* networks are greatly influenced by temporal relationships (e.g., whichever

---

<sup>★</sup> Work partially supported by MURST Project “Metodi Formali per la Sicurezza ed il Tempo” (MEFISTO); by MIUR project COVER; by IST-FET project “Design Environments for Global ApplicationS (DEGAS)”; by CNR project “Tecniche e Strumenti Software per l’Analisi della Sicurezza delle Comunicazioni in Applicazioni Telematiche di Interesse Economico e Sociale” and by a CSP grant for the project “SeTAPS II”.

the media access control protocol may be, real-time and temporal synchronization constraints characterize the hosts' communications). The reader can refer to [14] for a full-detailed survey about real-time issues in *ad hoc* networks. Actually, we are mainly interested in time synchronization issues for a subclass of *ad hoc* networks, i.e. wireless sensor networks. A wireless sensor network is typically composed of hundreds or thousands of sensors, (up to) cubic millimeters devices provided with autonomous sensing, computation and communication. The network is coordinated in a distributed mode in order to collect information on their surroundings. Sensor networks applications are expected to range over many fields, from home applications like automation and smart environments to military uses (monitoring equipment and ammunitions, battlefield management, etc.). Sensors may be used in a building for heating and air conditioning control as well as in a hospital for medical monitoring (e.g. drugs administration or telemonitoring of physiological conditions of the patients), not to mention environmental monitoring, such as the detection of possible fires in a forest.

Researchers have recently addressed the quest for securing wireless sensor networks communication. Sensors already have to cope with severe constraints in terms of power consumption, bandwidth, storage and may not have the resources to perform cryptographic operations in their completeness. Standard solutions developed for conventional computers cannot be applied, hence new schemes have been proposed and surveys have been carried out (e.g. [10, 13]). Temporal constraints occur in [13], where a time synchronization is required between a base station and the sensors in the network.

In [8] it was developed a compositional analysis technique able to deal with multicast/broadcast protocols. Actually, in that first work we were not able to manage protocols with time-dependent security properties. In this paper we aim at enhancing that analysis technique in order to cope with wireless protocols where a time synchronization is required, e.g. [13]. Among the properties we are now able to check are timed secrecy and timed integrity. The former requires that a message is kept secret only for a certain amount of time and the latter that a message sent in a time interval has not been altered during the communication. We use the approach based on non-interference developed in [5] as a method to express security properties.

This paper improves the work in [8] as follows: 1) a formal framework for modeling wireless communication protocols is outlined by means of a real-time process algebra; 2) a new compositional proof rule is given for dealing with timed security properties as timed secrecy and timed integrity; 3) a formal specification of  $\mu$ TESLA (see [13]) is defined and analyzed; it is checked that such a protocol enjoys the timed integrity property; 4) the analysis of  $\mu$ TESLA is carried out taking into consideration an arbitrary (but finite) number of senders and receivers. The previous work on a related protocol (the TESLA protocol, [12]), deals only with a fixed and small number of senders and receivers, [2].

This paper is organized as follows. Section 2 recalls the formal language we are going to adopt for the description of  $\mu$ TESLA. Section 3 presents the compositional analysis techniques in a timed setting. Section 4 introduces the  $\mu$ TESLA

formal specifications. Section 5 shows how to apply the previous theories to the analysis of  $\mu$ TESLA. Finally, Section 6 concludes the paper.

## 2 A Real-Time Language for Cryptographic Protocols

The real-time extension of the *Cryptographic Security Process Algebra* (*CryptoSPA* for short) in [3, 5] has been proposed in [6]. The new language, *timedCryptoSPA* (*tCryptoSPA* for short), is adopted for describing cryptographic protocols where information about the concrete timing of events is necessary. We remind the reader of the syntax, the operational semantics of the language and some auxiliary notions.

**The Syntax** The syntax of *tCryptoSPA* is based on a set  $\mathcal{I}$  of input channels (ranged over by  $c$ ), a set  $\mathcal{O}$  of output channels (ranged over by  $\bar{c}$ ), a set of closed terms  $\mathcal{M}$  (a set of closed terms of a term algebra which, at least, contains encryption  $\{m\}_k$  and pairing  $(m, m_1)$  operations), *Const* of constants, ranged over by  $A$ , and *Var* of variables, ranged over by  $x$ . The set  $\mathcal{L}$  of *tCryptoSPA* processes is defined as:

$$P ::= \mathbf{0} \mid c(x).P \mid \bar{c}e.P \mid \tau.P \mid tick.P \mid P_1 + P_2 \mid P_1 \mid P_2 \mid P \setminus L \mid \\ A(e_1, \dots, e_n) \mid [(e_1, \dots, e_r) \vdash_{rule} x]P_1; P_2$$

where  $e, e_1, \dots, e_r, e_n$  are messages or variables and  $L$  is a set of channels. Both the operators  $c(x).P$  and  $[(e_1 \dots e_r) \vdash_{rule} x]P_1; P_2$  bind the variable  $x$  in  $P, P_1$ .

Let  $Def : Const \rightarrow \mathcal{L}$  be a set of defining equations of the form  $A(x_1, \dots, x_n) \doteq P$ , where  $P$  may contain no free variables except  $x_1, \dots, x_n$ , which must be distinct. Constants permit us to define recursive processes, but we have to be a bit careful in using them. A term  $P$  is *closed* w.r.t.  $Def$  if all the constants occurring in  $P$  are defined in  $Def$  (and, recursively, for their defining terms). A term  $P$  is *guarded* w.r.t.  $Def$  if all the constants occurring in  $P$  (and, recursively, for their defining terms) occur in a prefix context [11].

The set *Act* of actions which may be performed by a system is defined as:  $Act = \{c(m), \bar{c}m, \tau, tick, \mid c \in \mathcal{I}, \bar{c} \in \mathcal{O}, m \in \mathcal{M}, m \text{ closed}\}$ .  $\tau$  is the internal, invisible action.  $tick$  is the special action used to model time elapsing. We let  $l$  range over  $Act \setminus \{tick\}$ . We call  $\mathcal{P}$  the set of all the *tCryptoSPA* *closed* terms (i.e., with no free variables) that are closed and guarded w.r.t.  $Def$ . We define  $sort(P)$  to be the set of all the channels syntactically occurring in the term  $P$ .

The informal semantics of the *tCryptoSPA* processes is the following:

- $\mathbf{0}$  is the process that does nothing;
- $c(x).P$  is the process that can receive a message  $m$  on channel  $c$  and then behaves like  $P$ . The received message replaces the variable  $x$ ;
- $\bar{c}m.P$  is the process that can send  $m$  on channel  $c$ , then behaving like  $P$ ;
- $\tau.P$  is the process that executes the internal, invisible action  $\tau$  and then behaves like  $P$ ;
- $tick.P$  is a process willing to let one time unit pass and then behaving as  $P$ ;

- $P_1 + P_2$  (*choice*) represents the nondeterministic choice between the two processes  $P_1$  and  $P_2$ ; with respect to *tick* actions, time passes when both  $P_1$  and  $P_2$  are able to perform a *tick* action – and in such a case by performing *tick* a configuration where both the derivatives of the summands can still be chosen is reached. When only one of the two processes can perform *tick*, say  $P_1$ , it could be either that  $P_1$  performs *tick* – and in such a case  $P_2$  is discarded – or  $P_2$  performs its normal activity – and in such a case  $P_1$  is discarded; moreover,  $\tau$  prefixed summands have priority over *tick* prefixed summands;
- $P_1 | P_2$  (*parallel*) is the parallel composition of processes that can proceed in an asynchronous way but they must synchronize on complementary actions to make a communication, represented by a  $\tau$ . Both components must agree on performing a *tick* action, and this can be done even if a communication is possible (we do not have *maximal progress* assumption);
- $P \setminus L$  allows only visible actions whose channels are not in  $L$ ;
- $A(m_1, \dots, m_n)$  behaves like the respective defining term  $P$  where all the variables  $x_1, \dots, x_n$  are replaced by the messages  $m_1, \dots, m_n$ ;
- $[(e_1, \dots, e_r) \vdash_{rule} x]P_1; P_2$  is the process used to model message handling and cryptography. The process  $[(e_1, \dots, e_r) \vdash_{rule} x]P_1; P_2$  tries to deduce an information  $z$  from the tuple of messages  $(e_1, \dots, e_r)$  through the application of rule  $\vdash_{rule}$ ; if it succeeds then it behaves like  $P_1[z/x]$ , otherwise like  $P_2$ . The set of rules that can be applied is defined through an inference system (e.g., see Figure 1).

**Auxiliary notions** The time model adopted in the language is known as the *fictitious clock* approach of, e.g., [9]. A global clock is supposed to be updated whenever all the processes agree on this, by globally synchronizing on the special action *tick*, representing the passing of a time unit. All the other actions are assumed to take no time.

In order to model message handling and cryptography we use a set of inference rules. Note that *tCryptoSPA* syntax, its semantics and the results obtained are completely parametric with respect to the inference system used. We show in Figure 1 a suitable inference system we are going to use in the following sections. This inference system can combine two messages obtaining a pair (rule  $\vdash_{pair}$ ); it can extract one message from a pair (rules  $\vdash_{fst}$  and  $\vdash_{snd}$ ); it can apply a one-way hash function  $F$  to message  $x$  and obtain digest  $F(x)$  (rule  $\vdash_{hash}$ ) and finally compute the message authentication code (MAC) of a message with a key (rule  $\vdash_{mac}$ ).

Given an inference system, we can define a *deduction function*  $\mathcal{D}$  s.t. if  $\phi$  is a finite set of closed messages, then  $\mathcal{D}(\phi)$  is the set of closed messages that can be deduced starting from  $\phi$  by applying instances of the rules in the system.

*Example 1.* We do not explicitly define an equality check among messages in the syntax. However, this can be implemented through the usage of the inference construct. E.g., consider rule  $equal \doteq \frac{x \quad x}{Equal(x, x)}$ . Then  $[m = m']A$  (with the expected semantics) may be equivalently expressed as  $[(m \ m') \vdash_{equal} y]A$  where  $y$  does not occur in  $A$ .

---


$$\begin{array}{c}
\frac{m \quad m'}{(m, m')} (\vdash_{pair}) \quad \frac{(m, m')}{m} (\vdash_{fst}) \quad \frac{(m, m')}{m'} (\vdash_{snd}) \\
\frac{F \quad x}{F(x)} (\vdash_{hash}) \quad \frac{m \quad k}{mac(m, k)} (\vdash_{mac})
\end{array}$$


---

**Fig. 1.** An example inference system.

The operational semantics of a *tCryptoSPA* term is described by means of the *labeled transition system* (*lts*, for short)  $\langle \mathcal{P}, Act, \{\xrightarrow{a}\}_{a \in Act} \rangle$ , where  $\{\xrightarrow{a}\}_{a \in Act}$  is the least relation between *tCryptoSPA* processes induced by the axioms and inference rules of Figure 2.

The expression  $P \xrightarrow{a} P'$  is a shorthand for  $P(\xrightarrow{\tau})^* P_1 \xrightarrow{a} P_2 (\xrightarrow{\tau})^* P'$ ,  $a \neq \tau$ , where  $(\xrightarrow{\tau})^*$  denotes a (possibly empty) sequence of transitions labeled  $\tau$ . The expression  $P \Rightarrow P'$  is a shorthand for  $P(\xrightarrow{\tau})^* P'$ . Let  $\gamma = a_1, \dots, a_n \in (Act \setminus \{\tau\})^*$  be a sequence of actions; then  $P \xrightarrow{\gamma} P'$  iff there exist  $P_1, \dots, P_{n-1} \in \mathcal{P}$  such that  $P \xrightarrow{a_1} P_1 \xrightarrow{a_2} \dots \xrightarrow{a_n} P'$ . Let  $\mathbf{0}' \doteq tick.\mathbf{0}'$ .

For timed behavioural relations among *tCryptoSPA* processes, we will be mainly interested in *timed trace* inclusions.

**Definition 1.** For any  $P \in \mathcal{P}$  the set  $T(P)$  of timed traces associated with  $P$  is defined as follows  $T(P) = \{\gamma \in (Act \setminus \{\tau\})^* \mid \exists P'. P \xrightarrow{\gamma} P'\}$ . The timed trace pre-order, denoted by  $\leq_{ttrace}$ , is defined as follows:  $P \leq_{ttrace} Q$  iff  $T(P) \subseteq T(Q)$ .  $P$  and  $Q$  are timed trace equivalent, denoted by  $P =_{ttrace} Q$ , if  $T(P) = T(Q)$ .

We define the concept of weak simulation as usual.

**Definition 2.** We say that a relation  $R$  among processes is a weak simulation, if for every  $(P, Q) \in R$  we have:

- If  $P \xrightarrow{a} P'$ ,  $a \neq \tau$ , then there exists  $Q'$  s.t.  $Q \xrightarrow{a} Q'$  and  $(P', Q') \in R$ .
- If  $P \xrightarrow{\tau} P'$  then there exists  $Q'$  s.t.  $Q \Longrightarrow Q'$  and  $(P', Q') \in R$ .

Let  $\prec$  the union of all weak simulations among processes. Then, we have  $\prec \subseteq \leq_{ttrace}$ .

### 3 tGNDC

A general schema for the definition of timed security properties, called *timed Generalized Non Deducibility on Compositions* (*tGNDC* for short) has been proposed in [6]: a system  $S$  is *tGNDC*  $\alpha$  iff for every enemy  $X$  the composition of the system with  $X$  satisfies the timed specification  $\alpha(S)$ . Basically, *tGNDC* guarantees that the timed property  $\alpha$  is satisfied, with respect to the  $\prec$  timed behavioural relation, even when the system is composed with any possible adversary  $X$ .

We give here the set of admissible hostile environments for our timed setting. For a certain enemy  $X$ , we call  $ID(X)$  the set of closed messages that

---


$$\begin{array}{c}
\text{(input)} \frac{m \in \mathcal{M}}{c(x).P \xrightarrow{c(m)} P[m/x]} \quad \text{(output)} \frac{}{\bar{c}m.P \xrightarrow{\bar{c}m} P} \quad \text{(internal)} \frac{}{\tau.P \xrightarrow{\tau} P} \\
\\
\text{(tick)} \frac{}{\text{tick}.P \xrightarrow{\text{tick}} P} \quad (|)_1 \frac{P_1 \xrightarrow{l} P'_1}{P_1 | P_2 \xrightarrow{l} P'_1 | P_2} \quad (|)_2 \frac{P_1 \xrightarrow{c(x)} P'_1 \quad P_2 \xrightarrow{\bar{c}m} P'_2}{P_1 | P_2 \xrightarrow{\tau} P'_1 | P'_2} \\
\\
(|)_3 \frac{P_1 \xrightarrow{\text{tick}} P'_1 \quad P_2 \xrightarrow{\text{tick}} P'_2}{P_1 | P_2 \xrightarrow{\text{tick}} P'_1 | P'_2} \quad (\setminus L) \frac{P \xrightarrow{c(m)} P' \quad c \notin L}{P \setminus L \xrightarrow{c(m)} P' \setminus L} \quad (+1) \frac{P_1 \xrightarrow{l} P'_1}{P_1 + P_2 \xrightarrow{l} P'_1} \\
\\
(+2) \frac{P_1 \xrightarrow{\text{tick}} P'_1 \quad P_2 \xrightarrow{\text{tick}} P'_2}{P_1 + P_2 \xrightarrow{\text{tick}} P'_1 + P'_2} \quad (+3) \frac{P_1 \xrightarrow{\text{tick}} P'_1 \quad P_2 \xrightarrow{\text{tick}} P'_2}{P_1 + P_2 \xrightarrow{\text{tick}} P'_1} \\
\\
\text{(Def)} \frac{P[m_1/x_1, \dots, m_n/x_n] \xrightarrow{a} P' \quad A(x_1, \dots, x_n) \doteq P}{A(m_1, \dots, m_n) \xrightarrow{a} P'} \\
\\
(\mathcal{D}) \frac{\langle m_1, \dots, m_r \rangle \vdash_{rule} m \quad P_1[m/x] \xrightarrow{a} P'_1}{[\langle m_1, \dots, m_r \rangle \vdash_{rule} x] P_1; P_2 \xrightarrow{a} P'_1} \\
\\
(\mathcal{D}) \frac{\exists m \text{ s.t. } \langle m_1, \dots, m_r \rangle \vdash_{rule} m \quad P_2 \xrightarrow{a} P'_2}{[\langle m_1, \dots, m_r \rangle \vdash_{rule} x] P_1; P_2 \xrightarrow{a} P'_2}
\end{array}$$


---

**Fig. 2.** Structured Operational Semantics for tCryptoSPA (symmetric rules for  $+1, +3, |_1, |_2$  and  $\setminus L$  are omitted)

syntactically appears in  $X$ , all the messages initially known by  $X$ . Let  $\phi_0$  be the initial knowledge we would like to give to the enemy at the beginning of the computation. We require that all the messages in  $ID(X)$  are deducible from  $\phi_0$ . We consider as hostile processes only the ones belonging to the set  $t\mathcal{E}_C^{\phi_0}$ <sup>1</sup>. They can communicate on a subset of public channels  $C$  and have an initial knowledge bound by  $\phi_0$ :

$$t\mathcal{E}_C^{\phi_0} = \{X \in \mathcal{P} \mid \text{sort}(X) \subseteq C \text{ and } ID(X) \subseteq \mathcal{D}(\phi_0)\}$$

The property  $tGNDC_{\mathcal{D}}^\alpha$  is defined as follows:

**Definition 3.**  $S$  is  $tGNDC_{\mathcal{D}}^\alpha$  iff  $\forall X \in t\mathcal{E}_C^{\phi_0} : (S|X) \setminus C \triangleleft \alpha(S)$  where  $\triangleleft$  is a timed behavioural relation between processes and  $\alpha : \mathcal{P} \rightarrow \mathcal{P}$  is a function between processes defining the property specification for  $S$  as the process  $\alpha(S)$ .

We may define several security properties through the  $tGNDC$  schema, e.g. see [6]. For instance timed secrecy expresses that a certain message  $m$  is not known by the intruder within a certain amount of time, say at least  $n$  units of time. A

<sup>1</sup> Actually, there is another constraint that imposes that the enemy must eventually let time pass. This is however not useful for safety properties we are going to study in this paper and so it has been omitted for the sake of simplicity.

specification  $\alpha_{tSec}$  dealing with timed secrecy could be the following:

$$\begin{aligned} pub(m) &= \overline{public}(m).0' + tick.pub(m) \\ \alpha_{tSec} &= tick_1 \dots tick_n.(pub(m)) \end{aligned}$$

where we assume  $public$  the unique not restricted channel.  $\alpha_{tSec}$  let  $n$  units of time pass and then behaves like  $pub(m)$ , i.e. it could either sends  $m$  over  $public$  or it could let time pass and possibly sends  $m$ .

Note that the GNDC theory is now a well established approach for security analysis and it was developed for non-deterministic, probabilistic, real time and cryptographic frameworks, e.g. see [1, 4, 5, 6, 7]. Here we present an extension of the compositional analysis in a real-time setting within the GNDC theory.

### 3.1 Time-dependent stability and compositional results

A compositional principle gives sufficient conditions to conclude that the parallel composition of two (or more) processes satisfies a certain property, provided that the single processes by themselves satisfy the same property. Compositional reasoning is often useful. An interesting application field is indeed the analysis of systems with an arbitrary number of components.

Here, we give a new result about conditions for safe composition of digital stream protocols where time plays an essential role. In order to achieve this result, we should refine the concept of stability defined in [6] basically requires that the intruder knowledge does not increase when composing the intruder process with a process  $P$ . If so, we call  $P$  a stable process. In [6] it was also noticed that if we assume that the intruder knowledge does not increase when composing the intruder process with  $P$  (i.e.  $P | X$ ) and with  $Q$  (i.e.  $Q | X$ ) (using the same communication channels) then the intruder knowledge does not increase when composing the intruder itself with the process  $P | Q$ . Unfortunately, such a form of stability is not time-dependent, i.e. it takes into account the same knowledge during all the temporal execution of the processes at stake. This does not make it feasible to check properties based on a timed notion of secrecy and, consequently, to check protocols as  $\mu$ TESLA, whose security features exactly depend on a form of timed secrecy. We give now a refined notion of stability, called time-dependent stability, that allows us to cope with timed secrecy and so also with security properties of protocols that rely on it.

We let  $\gamma$  be a sequence of actions (possibly empty) ranging over  $Act \setminus \{\tau\}$ . Let  $\#^{tick}(\gamma)$  be the number of occurrences of  $tick$  actions in the sequence  $\gamma$ .

**Definition 4.** *Let  $X_\phi$  be the closed term in  $X$  belonging to the messages deducible from  $\phi$ . We say that a process  $P$  is time-dependent stable w.r.t. the sequence  $\{\phi_i\}_{i>0}$  if, whenever  $(P | X_{\phi_0}) \setminus C \xrightarrow{\gamma} (P' | X'_{\phi'}) \setminus C$  and  $\#^{tick}(\gamma) = i$ , then  $\mathcal{D}(\phi') = \mathcal{D}(\phi_i)$ .*

Basically, a process  $P$  is time-dependent stable if an enemy cannot increase significantly its knowledge when  $P$  runs in the space of a time slot. The following proposition holds:

**Proposition 1.** *Given a sequence  $\{\phi_i\}_{i \geq 0}$  and a set of public channels  $C$ , assume  $P_r \in tGNDC_{\leq ttrace}^{\alpha_r(P_r)}$  with  $1 \leq r \leq n$ . Assume also  $P_r$  t. d. stable w.r.t.  $\{\phi_i\}$ . It follows that  $(P_1 | P_2 | \dots | P_n) \in tGNDC_{\leq ttrace}^{\alpha_1(P_1) | \alpha_2(P_2) | \dots | \alpha_n(P_n)}$  and  $(P_1 | P_2 | \dots | P_n)$  is t. d. stable w.r.t.  $\{\phi_i\}$ .*

*Example 2.* The process  $P = tick.\bar{c}k.\mathbf{0}'$  enjoys the secrecy of  $k$  for one time unit. In the more complex process  $Q = (c(x).[x = k]\bar{c}m) + tick.\mathbf{0}'$  the secrecy of  $k$  in the first time unit is crucial to get the secrecy of  $m$ . Indeed, either  $Q$  is willing to receive the key  $k$  only in the first time unit (if so, it releases  $m$ ) or it starts to idle. We have that  $P$  and  $Q$  are t.d. stable w.r.t.  $\phi_0 = \{\emptyset\}$ ,  $\phi_i = \mathcal{D}(\{k\})$  for  $i \geq 1$ . Then,  $P | Q$  is t.d. stable w.r.t.  $\{\phi_i\}_{i \geq 0}$  (by Proposition 1) and so  $m$  will never belong to the knowledge of the intruder (whose initial knowledge is  $\emptyset$ ).

## 4 The $\mu$ TESLA Protocol

In [13], Perrig *et al.* presented  $\mu$ TESLA (“micro” Timed Efficient Stream Loss-tolerant Authentication), a protocol to provide authenticated broadcast in wireless sensor networks environments. [13] considers a scenario where sensors communicate with a base-station connected to the external world. The base station may broadcast to all nodes messages for routing updates, reprogramming, reset requests. The protocol is an extension of the TESLA stream authentication protocol developed in [12] and it was intentionally developed for providing authenticated broadcast for the limited computing environments that are encountered in sensor networks.

In the original TESLA schema, a single sender broadcasts a continuous stream of packets. Receivers may use information in later packets to authenticate earlier packets. Each packet contains a message authentication code (MAC), i.e. a value computed by applying a public algorithm and a secret encryption key to the packet itself. Given a message  $m$  and an encryption key  $k$ , we call  $mac(m, k)$  the message authentication code of  $m$ . The algorithm is known by all the receivers, while the encryption keys are disclosed by the sender after a certain amount of time. When a receiver receives a key  $K_i$  it can use it to compute the MAC from the related packet  $P_i$  and compare the computed MAC with that previously received. If the two MACs match, the receiver can consider the packet  $P_i$  authentic. To avoid the event that an intruder could use a disclosed key  $K_i$  to fake the packet  $P_i$  a time synchronization protocol between the sender and the receivers is needed. Then, each receiver will not accept the packet  $P_i$  if the sender might have already sent the key  $K_i$ .

Bootstrapping authentication of the whole scheme is achieved in TESLA by signing the first packet with a regular digital signature scheme. Nevertheless, computation, communication and storage overhead make the use of asymmetric cryptography unfeasible for the net of sensors under investigation. Thus,  $\mu$ TESLA has been proposed as an optimized extension for sensor networks. It just makes use of MACs. The base-station randomly generates the last MAC

key to be used,  $K_{last}$ , and derives a key chain by repeatedly applying a publicly known one-way function  $F$  to that key, such that  $K_i = F(K_{i+1})$ . Given the non-reversibility property (at least with high probability) of function  $F$ , the disclosure of key  $K_i$  should not lead to any knowledge of  $K_{i+1}$  and subsequent keys.

Receivers' requirements for correctly joining and executing the protocol are: i) they are time synchronized with the base station; ii) they know the disclosure schedule of the MAC keys; iii) they know at least one authenticated key of the key chain, serving as a commitment to the entire chain. A protocol providing time synchronization and one authenticated key has been proposed in [13]. Basically, the base-station shares with each sensor a symmetric secret key  $K_{SM}$  and establishes a secure channel over which the exchange of a commitment to the key chain,  $K_0$ , and a set of temporal parameters,  $set_t$ , takes place<sup>2</sup>. More formally, the initial step of  $\mu$ TESLA is the following:

$$\text{Packet } P_0 \quad c_0 \ S \rightarrow \{R_n\} : K_0, set_t, mac(K_0, set_t, K_{SM})$$

where  $c_0 \in \{c_i\}_{i \in \mathbb{N}}$ , i.e. the set of communication channels,  $S$  is the identifier of the sender<sup>3</sup> (i.e. the base station) and  $\{R_n\}$  is the set of receivers (i.e. the sensors).

$\mu$ TESLA is parameterized by the schedule time at which MAC keys are disclosed. For the description of further steps in the protocol we consider a basic formalization, Fig. 3, where we suppose that the sender discloses a MAC key with a delay  $\delta = 1$ , assumed to fall in the interval after that key has been used to compute the MAC. Further, we suppose the sender sends one packet per time interval. Basically, in each time slot a packet and a key packet will be sent, Fig. 3. First of all, each receiver should check the integrity of the received key, say  $K_i$ , by verifying it w.r.t. an authenticated commitment (e.g. by checking  $K_0 = F^i(K_i)$ ), then the verified key will be used to verify the integrity of the packet received in the previous time slot.

$$\text{Packet } P_i \quad c_i \ S \rightarrow \{R_n\} : m_i, mac(m_i, K_i) \quad i \geq 1$$

Packet  $P_i$  consists of a meaningful payload  $m_i$  plus the message authentication code computed on  $m_i$  with key  $K_i$ . We assume that  $K_{SM}$  cannot be deduced from the sets  $\{m_i\}, \{K_i\}$ .

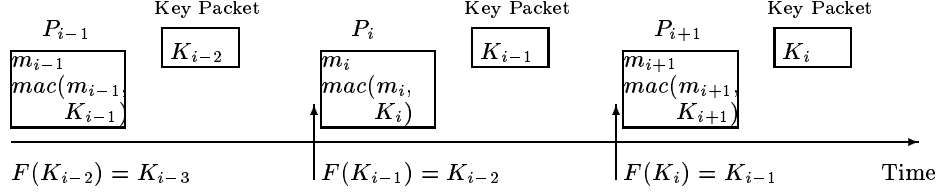
Upon receiving the packet, the sensor stores the packet until its MAC can be verified, i.e. until the sender broadcasts packet disclosing  $K_i$ :

$$\text{Key-Packet } KP_i \quad c_{i+1} \ S \rightarrow \{R_n\} : K_i$$

---

<sup>2</sup> There are as many symmetric keys as the number of sensors and the communication over channel  $c_0$  is supposed to be a point to point communication. Nevertheless, to simplify our formalization, we assume a unique key and a unique communication. This means to implicitly assume that possible adversaries are not in the set of receivers.

<sup>3</sup> To assure freshness when executing multiple runs of the same sender, one can simply insert nonces in the message authentication code of packet  $P_0$ .



**Fig. 3.** A  $\mu$ TESLA instantiation.

The integrity of key  $K_i$  can be checked by verifying  $K_0 = F^i(K_i)$  (or, equivalently,  $K_{i-1} = F(K_i)$ ). Packets may be lost in transit from the base station to the sensors. In particular  $\mu$ TESLA is tolerant to packet loss in the sense that receivers may still be able to authenticate all the received packets  $P_i$  even when the corresponding keys' disclosure packets are lost. Suppose  $K_j$  is lost, then a receiver is not able to verify MAC packet  $P_j$ . The following key the receiver recovers, let it be  $K_{j+1}$ , can be verified w.r.t. a previous authenticated key (e.g.  $K_0 = F^{j+1}(K_{j+1})$ ) and is used to derive  $K_j$ , i.e.  $K_j = F(K_{j+1})$ .

#### 4.1 The *tCryptoSPA* Specifications of the $\mu$ TESLA Protocol

We present the *tCryptoSPA* specifications of the basic  $\mu$ TESLA instantiation in Fig. 3. The fundamental requirement of a time synchronization between a base-station and each sensor in  $\mu$ TESLA is naturally captured in *tCryptoSPA* by its time modeling action *tick*, upon which sender and receivers' processes may synchronize (this allows us to avoid the explicit presence of  $set_t$  in packet  $P_0$ ).

We consider a sender machine with ample resources. It can be parallelized or split into  $n$  senders, each of them possibly sending different streams,  $\{m_i^j\}_{i \geq 1, 1 \leq j \leq n}$ . We first present the generic sender process  $S^j$ , parameterized by a sequence of MAC keys (tied together by means of a key chain)<sup>4</sup>. We assume the symmetric key  $K_{SM}$ , the keys belonging to the key chain and the streams of packets to be different for each process  $S^j, 1 \leq j \leq n$ <sup>5</sup>.

$$\begin{array}{ll}
 S_0^j(K_{SM}^j, K_0^j, K_1^j, \dots) \doteq & \\
 [K_0^j \quad K_{SM}^j \vdash_{mac} y] & \text{Compute MAC} \\
 [K_0^j \quad y \vdash_{pair} P_0] & \text{Create packet } P_0 \\
 B_0^j(P_0) & \text{Start to broadcast } P_0
 \end{array}$$

<sup>4</sup> Actually, we consider constants with an arbitrary number of parameters. We could avoid this by considering, for modeling purposes, a special function *fun*, not available to possible adversaries, that may be used to represent the keys as a sequence.

<sup>5</sup> We remind the reader that the whole formalization we are going to give is based on choices of the authors since some details are not explicitly given in [13]. In particular, the mechanism through which a receiver possibly identifies each sender process (and consequently each stream) is not defined in [13], since the original construction is described with a single sender.

$$\begin{aligned}
S_1^j(K_0^j, K_1^j, \dots) &\doteq \\
&[m_1^j \quad K_1^j \vdash_{mac} x] \text{ Compute MAC} \\
&[m_1^j \quad x \vdash_{pair} P_1] \text{ Create packet } P_1 \\
&B_1^j(P_1) \quad \text{Start to broadcast } P_1
\end{aligned}$$

$$\begin{aligned}
S_i^j(K_{i-1}^j, K_i^j, \dots) &\doteq \\
&[m_i^j \quad K_i^j \vdash_{mac} x] \text{ Compute MAC} \\
&[m_i^j \quad x \vdash_{pair} P_i] \text{ Create packet } P_i \\
&B_i^j(P_i, K_{i-1}^j) \quad \text{Start to broadcast } P_i \text{ and disclose key } K_{i-1}
\end{aligned}$$

$$\begin{aligned}
B_i^j(P_i) &\doteq \bar{c}_i P_i . B_i^j(P_i) + tick . S_{i+1}^j(K_i^j, \dots) \quad i = 0, 1 \\
B_i^j(P_i, K_{i-1}^j) &\doteq \bar{c}_i P_i . \bar{c}_i K_{i-1}^j . B_i^j(P_i, K_{i-1}^j) + tick . S_{i+1}^j(K_i^j, \dots) \quad i \geq 2
\end{aligned}$$

Construct  $B_i^j(\dots)$  is responsible for potentially sending packets (and keys) an unbounded number of times, in order to simulate a one-to-all sending typical of broadcast sessions. Sender  $S^j$  remains in the same state repeatedly sending messages unless the non-deterministic choice is resolved by choosing the derivative of the second summand in  $B_i^j$ ; this causes a time unit to pass (a *tick* action is performed). The construction models the behaviour of a wireless antenna making signals available only in a particular time interval. The presence of a non-deterministic choice in the construct makes it possible the passage to the following time interval without performing any (eventually zero) communication. This may implicitly model the unreliability of the wireless transmission and the occurrence of packet loss.

Among the receivers' set, each process behaves in the same way. The generic receiver process at step  $i$  is parameterized by a commitment to the key chain (let it be  $K_0^j$ ) and by the packets it should still authenticate. We assume the receiver's set is divided into subgroups, each of them sharing a particular  $K_{SM}$  with one sender process. Sender  $S^j$  and receivers belonging to subgroup number  $j$  share  $K_{SM}^j$ .  $K_{SM}^j$  may denote a particular service each element in subgroup  $j$  is devoted to. Let us consider pay per view-based applications: among the receivers' set, the subgroup knowing  $K_{SM}^j$  may consist of all the paying spectators for movie number  $j$ . For environments closer to those depicted for  $\mu$ -TESLA, let us consider a scenario in which sensors are used to periodically transmit readings regarding heating and air conditioning control in a building (and consequently receive broadcasted messages for routing updates or reprogramming): sensors in subgroup  $j$  may be all the sensors devoted to carry out the service for room number  $j$ . ( $S^j$  being the base station responsible for room number  $j$ ).

Below, we refer to  $R_i^{j,q}$  to indicate the  $q$ -th receiver process belonging to subgroup  $j$  and acting at step  $i$ .

$R_0^{j,q}(null) \doteq$	
$c_0(x)$	Receive first packet
$[x \vdash_{fst} x_{K_0}]$	Extract commitment to the key chain
$[x_{K_0} \quad K_{SM}^j \vdash_{mac} z]$	Compute MAC
$[x \vdash_{snd} x_{mac}]$	Extract MAC
$[z = x_{mac}]$	Verify MAC: if verified:
$tick.R_1^{j,q}(x_{K_0});$	Allow a time unit to pass and go to next state
$R_0^{j,q}(null)$	Wait for key

Upon receiving a value  $x$  on channel  $c_0$ , the receiver verifies the correctness of the commitment to the key chain,  $x_{K_0}$ : it computes  $mac(x_{K_0}, K_{SM}^j)$  and compares it with the message authentication code in the received packet. If the two MACs match, a time unit passes and the receiver goes to the next state, otherwise the receiver remains in the same state waiting for the right key  $K_{SM}^j$ . Throughout the formalization,  $null$  means an empty field.

$R_1^{j,q}(x_{K_0}) \doteq$	
$(c_1(y)$	Receive packet
$tick.R_2^{j,q}(y, x_{K_0})$	Allow a time unit to pass and go to next state
$) + tick.R_2^{j,q}(null, x_{K_0})$	Go to next state after a time unit

$R_1^{j,q}$  is willing to accept any arbitrary packet, because it cannot perform any verification yet. If nothing is received before the end of a time unit, transition takes place to next state  $R_2^{j,q}$ .

$R_i^{j,q}(p_{i-1}, x_{K_0}) \doteq$	
$c_i(p_i).R_i^{j,q}(p_i, p_{i-1}, x_{K_0})$	Receive i-th packet; go to intermediary state $R_i^{j,q}$
$+ tick.R_{i+1}^{j,q}(null, x_{K_0})$	Go to next state after a time unit

$R_i^{j,q}$  is willing to accept packet  $P_i$  and travels to an intermediary state  $R_i^{j,q}$ . If nothing is received before the end of a time unit, transition takes place to the next state.

$R_i^{j,q}(p_i, p_{i-1}, x_{K_0}) \doteq$	
$c_i(x_{K_{i-1}})$	Receive key packet
$[x_{K_0} = F^{i-1}(x_{K_{i-1}})]$	Verify the key w.r.t. the commitment
$[p_{i-1} \vdash_{fst} y_{pay}]$	Extract payload
$([y_{pay} \quad x_{K_{i-1}} \vdash_{mac} z]$	If $x_{K_{i-1}} = K_{i-1}^j$ then: Compute MAC
$[p_{i-1} \vdash_{snd} y_{mac}]$	Extract MAC
$[z = y_{mac}]$	Verify MAC
$\overline{app}y_{pay}$	Send $m_1^j$ to application level
$tick.R_{i+1}^{j,q}(p_i, x_{K_0})$	Allow a time unit to pass and go to next state
$); R_i^{j,q}(p_i, p_{i-1}, x_{K_0})$	Wait for key

In intermediary state  $R_i^{j,q}$  receives a key packet and verifies the correctness of the key w.r.t. the authenticated commitment  $x_{K_0} = K_0^j$ . Given the collision-free property of one-way functions, if the verification does not succeed it means

$x_{K_{i-1}} \neq K_{i-1}^j$  and  $R_i^{j,q}$  simply stays in the same state waiting for the right subgroup key. If the verification succeeds, the correctness of  $P_{i-1}$  is verified by checking that the enclosed MAC is authentic. The successful outcome is here modeled by a scenario where the receiver sends the payload of the accepted packet over channel  $app$ <sup>6</sup>.

Suppose packet  $P_{i-1}$  was correctly received, suppose also packet disclosing  $K_{i-1}^j$  is lost. At step  $i$  the receiver still cannot authenticate packet  $P_{i-1}$ . The key chain mechanism of the original protocol takes into account such a possibility: in interval  $i+1$  the base station broadcasts key  $K_i^j$ , which the receiver authenticates by verifying  $K_0^j = F^i(K_i^j)$ . The receiver can authenticate  $P_i$  and derives  $K_{i-1}^j = F(K_i^j)$ , so it can also authenticate  $P_{i-1}$ . Actually, our formalization does not take into account recovering lost keys. For the sake of simplicity, we prefer to suppose that the key packet related to subgroup  $j$  is received (state  $R_i^{j,q}$ ).

We report below the formalization at step  $i$ , with  $i \geq 2$ , when a packet was not received at step  $i-1$ .

$$\begin{aligned} R_i^{j,q}(null, x_{K_0}) \doteq & \\ c_i(p_i).tick.R_{i+1}^{j,q}(p_i, x_{K_0}) & \text{Receive } i\text{-th packet; go to next state} \\ +tick.R_{i+1}^{j,q}(null, x_{K_0}) & \text{Go to next state after a time unit} \end{aligned}$$

## 5 An analysis of the $\mu$ TESLA protocol: timed integrity

We focus our attention on the so called timed integrity, belonging to a new class of properties defined in [6]. A stream signature protocol guarantees timed integrity on a set of messages  $\{m_i\}$  if, whenever the generic receiver accepts an item in a time interval  $i$ , let us say item  $x$ , then  $x = m_{i-\delta}$ ,  $i-\delta$  being the time interval in which  $x$  has been received. ( $\delta = 1$  in our formalization of  $\mu$ TESLA.) Timed integrity property may be efficiently verified by means of Proposition 1 in Subsection 3.1. We consider  $\mu$ TESLA as a case study for proving its correctness in terms of messages  $m_i$  timed integrity. We refer to the instantiation in Fig. 3 and its *tCryptoSPA* specifications of Subsection 4.1. Assume that a receiver signals the acceptance of a payload as a legitimate one, by issuing it on a special channel  $app$ .

Let  $P^q \doteq S_0^j | R_0^{j,q}$  be the system consisting of a single sender and  $q$ -th receiver in subgroup  $j$ , sharing  $K_{SM}^j$ . Let function  $\alpha_{tInt}(P^q)$  be  $tSpec_0$  where

$$\begin{aligned} tSpec_0 & \doteq tick.tSpec_1 \\ tSpec_1 & \doteq tick.tSpec_2 \\ tSpec_i & \doteq tick.tSpec_{i+1} + \overline{app}(m_{i-1}^j).tick.tSpec_{i+1} \quad i \geq 2 \end{aligned}$$

$\alpha_{tInt}(P^q)$  may denote the correct external behaviour of  $P^q$ . In the first two steps it simply let time pass, while in further steps it may either let time pass

<sup>6</sup> We omitted to insert an idling behavior when a deduction construct fails to be executed and in our formalization the system simply stops without letting time pass. This is not realistic, but it has no consequences since we use trace semantics for the analysis and makes it simpler.

(denoting packet loss) or let a verified payload to be sent on the special channel *app* and then let time pass. The set of all messages sent on channel *app* is the set of all the possible ordered substreams of  $\{m_i^j\}_{i \geq 1}$ . Let function  $\alpha_{tInt}^j(P^j) \doteq \prod_{1 \leq q \leq n_j} \alpha_{tInt}(P^q)$ ,  $n_j$  being the cardinality of the receivers in subgroup  $j$ .

**Definition 5.** *The system  $P^j \doteq S_0^j | R_0^{j,1} | R_0^{j,2} | \dots | R_0^{j,n_j}$ , consisting of a sender of streamed data  $\{m_i^j\}$  and the receivers in subgroup  $j$  enjoys the timed integrity property whenever  $P^j \in tGNDC_{\leq ttrace}^{\alpha_{tInt}^j(P^j)}$ .*

Basically, it means that each receiver accepts exactly the messages belonging to  $\{m_i^j\}$  in the correct order and within the time interval following the one in which the sender actually sent the messages, even in presence of an intruder (unless packets  $P_i$  are lost). The key point is that the intruder will never acquire the shared key  $K_{SM}^j$  to establish a secure channel over which the commitment to the key chain is exchanged<sup>7</sup>.

We first consider system  $P^q$ . We may prove that  $S_0^j$  and  $R_0^{j,q}$  (Subsection 4.1) are t.d. stable w.r.t. the sequence  $\{\phi_i\} = \phi_0, \phi_1, \phi_2, \dots$  defined as follows:

$$\begin{aligned} \phi_0 &= \{K_0^j, mac(K_0^j, K_{SM}^j) \mid 1 \leq j \leq n\} \\ \phi_1 &= \phi_0 \cup \{m_1^j, mac(m_1^j, K_1^j) \mid 1 \leq j \leq n\} \\ \phi_2 &= \phi_1 \cup \{m_2^j, mac(m_2^j, K_2^j), K_1^j \mid 1 \leq j \leq n\} \\ &\dots \\ \phi_i &= \phi_{i-1} \cup \{m_i^j, mac(m_i^j, K_i^j), K_{i-1}^j \mid 1 \leq j \leq n\} \\ &\dots \end{aligned}$$

where  $n$  is the number of senders.  $\phi_i$  is equal to  $\phi_{i-1}$  plus the set of all the messages an intruder would be able to add to its knowledge by eavesdropping on a run of the protocol during the whole time interval  $i$  (of course including those messages coming from all the other senders processes). Actually, the intruder will have more powerful means to act since the beginning of each time interval.

We may prove that  $S_0^j$  enjoys  $tGNDC_{\leq ttrace}^{\mathbf{0}'}$  and  $R_0^{j,q}$  enjoys  $tGNDC_{\leq ttrace}^{\alpha_{tInt}(P^q)}$ , that is to say for all  $X \in t\mathcal{E}_C^{\phi_0}$  we have  $(S_0^j | X) \setminus C \leq_{ttrace} \mathbf{0}'$  and  $(R_0^{j,q} | X) \setminus C \leq_{ttrace} \alpha_{tInt}(P^q)$ . This may be done by finding a suitable weak simulation relation between  $(S_0^j | X_{\phi_0}) \setminus C$  and  $\mathbf{0}'$  and between  $(R_0^{j,q} | X_{\phi_0}) \setminus C$  and  $tSpec_0$ , respectively. The set  $C$  of channels over which an intruder is able to communicate is  $C = \{c_i \mid i \geq 0\}$ . The weak simulation relation dealing with the sender specifications is the following:

$$\begin{aligned} \mathcal{R}_S &= (((S_i^j(\dots) | X_{\phi_i}) \setminus C, \mathbf{0}') \mid \forall i, X_{\phi_i} \in t\mathcal{E}_C^{\phi_i}) \\ &\cup (((B_i^j(\dots) | X_{\phi_i}) \setminus C, \mathbf{0}') \mid \forall i, X_{\phi_i} \in t\mathcal{E}_C^{\phi_i}) \\ &\cup (((\bar{c}_i K_{i-1}^j . B_i^j(\dots) | X_{\phi_i}) \setminus C, \mathbf{0}') \mid i > 1, X_{\phi_i} \in t\mathcal{E}_C^{\phi_i}) \end{aligned}$$

<sup>7</sup> We remind the reader that  $K_{SM}^m \neq K_{SM}^n$  if  $m \neq n$  and  $K_i^m \neq K_i^n$  if  $m \neq n$  or  $i \neq l$ .

The weak simulation relation we consider for dealing with the receiver specifications is the following (superscript  $q$  is omitted for simplicity):

$$\begin{aligned}
\mathcal{R} = & ((R_0^j(\text{null}) | X_{\phi_0}) \setminus C, tSpec_0) | X_{\phi_0} \in t\mathcal{E}_C^{\phi_0}) \\
& \cup ((tick.(R_1^j(K_0^j) | X_{\phi_0}) \setminus C, tSpec_0) | X_{\phi_0} \in t\mathcal{E}_C^{\phi_0}) \\
& \cup (((R_1^j(K_0^j) | X_{\phi_1}) \setminus C, tSpec_1) | X_{\phi_1} \in t\mathcal{E}_C^{\phi_1}) \\
& \cup (((R_i^j(\text{null}, K_0^j) | X_{\phi_i}) \setminus C, tSpec_i) | i \geq 2, X_{\phi_i} \in t\mathcal{E}_C^{\phi_i}) \\
& \cup ((tick.(R_i^j(p_{i-1}, K_0^j) | X_{\phi_{i-1}}) \setminus C, tSpec_{i-1}) | i \geq 2, X_{\phi_{i-1}} \in t\mathcal{E}_C^{\phi_{i-1}}) \\
& \cup (((R_i^j(p_{i-1}, K_0^j) | X_{\phi_i}) \setminus C, tSpec_i) | i \geq 2, X_{\phi_i} \in t\mathcal{E}_C^{\phi_i}) \\
& \cup (((R_i^j(p_i^*, p_{i-1}, K_0^j) | X_{\phi_i}) \setminus C, tSpec_i) | i \geq 2, X_{\phi_i} \in t\mathcal{E}_C^{\phi_i}) \\
& \cup (((R_i^j(x_{i-1}, K_0^j) | X_{\phi_i}) \setminus C, tSpec_i) | fst(x_{i-1}) \neq m_{i-1}^j, i \geq 2, X_{\phi_i} \in t\mathcal{E}_C^{\phi_i}) \\
& \cup ((tick.(R_i^j(x_{i-1}, K_0^j) | X_{\phi_{i-1}}) \setminus C, tSpec_{i-1}) | fst(x_{i-1}) \neq m_{i-1}^j, i \geq 2, \\
& X_{\phi_{i-1}} \in t\mathcal{E}_C^{\phi_{i-1}}) \\
& \cup ((tick.(R_i^j(p_{i-1}^*, K_0^j) | X_{\phi_{i-1}}) \setminus C, tick.tSpec_i) | i \geq 2, X_{\phi_{i-1}} \in t\mathcal{E}_C^{\phi_{i-1}})
\end{aligned}$$

where  $p_1, p_{i-1}, p_i^*, p_{i-1}^*$  and  $x_{i-1}$  are not empty fields.  $p_i^*, p_{i-1}^*$  are shortcuts to denote either authentic packets sent by the sender or others. We omitted to explicitly put in  $\mathcal{R}_S$  and  $\mathcal{R}$  the pairs in which the first process performs deduction constructs.

**Lemma 1.**  $S_0^j$  and  $R_0^{j,q}$  are t. d. stable w.r.t.  $\{\phi_i\}$ .

**Lemma 2.**  $S_0^j \in tGNDC_{\leq ttrace}^{0'}$  and  $R_0^{j,q} \in tGNDC_{\leq ttrace}^{\alpha_{tInt}(P^q)}$

The following proposition follows by Lemma 1,2 and by Proposition 1 where  $r = 1, 2, P_1 = S_0^j, P_2 = R_0^{j,q}$ .

**Proposition 2.**  $P^q \in tGNDC_{\leq ttrace}^{\alpha_{tInt}(P^q)8}$ .

The correctness of the multiple receivers version (considering all the receivers belonging to subgroup  $j$ ), can be also proved using results of Lemma 1,2 and Proposition 1 where index  $r$  is not fixed *a priori* and  $P_1 = S_0^j$  and  $P_r = R_0^{j,q}$  with  $1 \leq q \leq n_j$ .

**Proposition 3.** System  $P^j$  (in Definition 5)  $\in tGNDC_{\leq ttrace}^{\alpha_{tInt}^j(P^j)}$ .

We get into the issue of considering a multiple senders/receivers environment. Let us consider  $\Gamma = \prod_{1 \leq j \leq n} P^j$  and  $\alpha_{tInt}(\Gamma) = \prod_{1 \leq j \leq n} \alpha_{tInt}^j(P^j)$ , where  $n$  is the cardinality of the senders processes.

**Proposition 4.** System  $\Gamma \in tGNDC_{\leq ttrace}^{\alpha_{tInt}(\Gamma)}$ .

The result follows by application of Propositions 3 and 1.

Note that in order to have timed integrity on the messages  $m_i$ ,  $\mu$ TESLA must ensure timed secrecy on the keys  $K_i$ . Indeed, we could also check explicitly timed secrecy on the keys with the same machinery.

<sup>8</sup> Note that  $0' | \alpha_{tInt}(P^q) \leq ttrace \alpha_{tInt}(P^q)$ .

## 6 Conclusions

In this paper we presented some preliminary steps towards a framework suitable for the security analysis of time-dependent wireless protocols. In particular, we developed a compositional approach for reasoning about security properties that rely on time constraints. This allowed us to check a relevant protocol, i.e.  $\mu$ TESLA. As a future work, we plan to deal with security properties in a mobile framework and offering some tool support for our compositional analysis.

Related work in security protocol verification in a timed setting may be found in [15], where tock-CSP is presented. The main differences are a different treatment of time operators and cryptography modeling. Moreover, no compositional proof rule has been provided. However, the verification proposed in [15] is automated through the use of PVS ([16]) while ours is completely manual (as of now).

**Acknowledgments.** We would like to thank the anonymous referees for their helpful comments.

## References

- [1] A. Aldini, M. Bravetti, and R. Gorrieri. A Process-algebraic Approach for the Analysis of Probabilistic Non-interference. *Journal of Computer Security*, 2003.
- [2] P. Broadfoot and G. Lowe. Analysing a Stream Authentication Protocol using Model Checking. In *Proc. of ESORICS'02, LNCS 2502, 146-161*, 2002.
- [3] R. Focardi, R. Gorrieri, and F. Martinelli. Non Interference for the Analysis of Cryptographic Protocols. In *Proc. of ICALP'00, LNCS 1853, 354-372*, 2000.
- [4] R. Focardi, R. Gorrieri, and F. Martinelli. Real-Time Information Flow Analysis. *IEEE JSAC*, 21(1), 2003.
- [5] R. Focardi and F. Martinelli. A uniform approach for the definition of security properties. In *Proc. of FM'99, LNCS 1708, 794-813*, 1999.
- [6] R. Gorrieri, E. Locatelli, and F. Martinelli. A Simple Language for Real-time Cryptographic Protocol Analysis. In *Proc. of ESOP'03, LNCS 2618, 114-128*, 2003.
- [7] R. Gorrieri and F. Martinelli. Process Algebraic Frameworks for the Specification and Analysis of Cryptographic Protocols. In *Proc. of MFCS, LNCS 2747*, 2003.
- [8] R. Gorrieri, F. Martinelli, M. Petrocchi, and A. Vaccarelli. Compositional Verification of Integrity for Digital Stream Signature Protocols. In *Proc. of IEEE ACSD'03, 142-149*, 2003.
- [9] M. Hennessy and T. Regan. A Temporal Process Algebra. *I&C*, 117:222-239, 1995.
- [10] Y. W. Law, S. Dulman, S. Etalle, and P. Havinga. Assessing Security in Energy-efficient Sensor Networks. In *Proc. of Small Systems Security Workshop'03*, 2003.
- [11] R. Milner. *Communication and Concurrency*. Prentice Hall, 1989.
- [12] A. Perrig, R. Canetti, D. X. Song, and D. Tygar. Efficient and Secure Source Authentication for Multicast. In *Proc. of NDSS'01*. The Internet Society, 2001.
- [13] A. Perrig, R. Szewczyk, J. D. Tygar, V. Wen, and D. Culler. SPINS: Security Protocols for Sensor Networks. *Wireless Networks Journal*, 8:521-534, 2002.
- [14] K. Romer. Time Synchronization in Ad Hoc Networks. In *Proc. of ACM MobiHoc'01*, pages 173-182, 2001.
- [15] S. Schneider. Analysing Time-Dependent Security Properties in CSP using PVS. In *Proc. of ESORICS'00, LNCS 1895*, 2000.
- [16] N. Shankar, S. Owre, and J. M. Rushby. *PVS Tutorial*. Tutorial Notes, *FME '93: Industrial-Strength Formal Methods*, pages 357-406, April 1993.