A Calculus for the Analysis of Wireless Network Security Protocols*

Francesco Ballardin and Massimo Merro

Dipartimento di Informatica, Università degli Studi di Verona, Italy

Abstract We propose a timed broadcasting calculus for *wireless systems*. The operational semantics of our calculus is given both in terms of a Reduction Semantics and in terms of a Labelled Transition Semantics. We prove that the two semantics coincide. The labelled transition system is used to derive a standard notion of (weak) bi-similarity which is proved to be a congruence. We use our *simulation theory* to adapt Gorrieri and Martinelli's *tGNDC* scheme to investigate, in our setting, the safety of non-trivial *wireless network security protocols*.

1 Introduction

Communication technologies such as WiFi, Bluetooth, and Homeplug are widely diffused and rely on short-range networking for disparate wireless devices in home and office environments. Larger wireless networks such as cellular, sensor and vehicular ad hoc networks are also becoming more and more popular.

In order to design secure wireless networks, several aspects have to be considered [1]: key establishment, secrecy, authentication, and privacy. Key establishment is at the core of any security framework. Traditional solutions for data secrecy and authentication rely on cryptographic protocols, which typically use either public key or symmetric key algorithms. However, in many wireless systems (such as sensor networks and vehicular ad hoc networks) resource limitations and/or real-time constrains impose to use symmetric key algorithms.

In the last four years, a number of distributed process calculi have been proposed for modelling different aspects of wireless systems [2,3,4,5,6,7,8,9]. However, none of this calculi deal with security protocols. On the other hand, process algebras, such as `CryptoCCS` and `tCryptoSPA` [10] have already been used in [10,11] to study network security protocols, also in a wireless scenario. These calculi are extensions of Milner's CCS [12], where node distribution, local broadcast communication, and message loss are not primitives but they are codified in terms of point-to-point transmission and a (discrete) notion of time.

We propose a simple timed broadcasting process calculus, called `tcryptoCWS`, for modelling wireless network security protocols. As usual in wireless systems, our broadcast communications span over a limited area, called *transmission range*. The time model we use is known as the *fictitious clock* approach (see e.g. [13]). A global clock is supposed to be updated whenever all nodes agree on this, by globally synchronising on a special action σ. All the other actions are assumed to take no time. This is reasonable if we choose a time unit such that the actual time

* This work was partially supported by the PRIN 2007 project "SOFT".

Table 1 The Syntax

Networks:		
M, N ::=	$\mathbf{0}$	empty network
|	$M_1 \mid M_2$	parallel composition
|	$n[P]^{\nu}$	node
Processes:		
P, Q ::=	nil	termination
|	$!\langle u\rangle.P$	broadcast
|	$\lfloor ?(x).P \rfloor Q$	receiver with timeout
|	$\lfloor \tau.P \rfloor Q$	internal with timeout
|	$\sigma.P$	delay
|	$[u_1 = u_2]P; Q$	matching
|	$[u_1 \dots u_n \vdash_{\mathrm{r}} x]P; Q$	deduction
|	$H\langle \tilde{u}\rangle$	recursion

of an action is negligible with respect to the time unit. The operational semantics of the calculus is given both in terms of a reduction semantics and in terms of a labelled transition semantics. The two operational semantics are proved to coincide. The calculus enjoys standard time properties, such as: *time determinism*, *maximal progress* and *patience* [13]. The labelled transition semantics is used to derive a standard notion of (weak) *bi-similarity* which is proved to be a congruence.

As a main application, we provide a clear and formal specification of two wireless network security protocols: (i) μTESLA [14], a well-known protocol to achieve *authenticated broadcast* in wireless sensor networks; (ii) Localized Encryption and Authentication Protocol (LEAP+) [15], a key management protocol intended for large-scale wireless sensor networks.

We use our *simulation theory*, to adapt Gorrieri and Martinelli's *Timed Generalized Non-Deducibility on Compositions* (tGNDC) scheme [10,11], a well-known general framework for the definition of timed security properties. In particular, we concentrate on two properties: *timed integrity*, which guarantees on the freshness of authenticated packets; and *timed agreement*, for which agreement between initiator and responder must be reached within a certain deadline. We formally prove that the μTESLA protocol enjoys both timed integrity and timed agreement. We then prove that the *single-hop pairwise shared key* mechanism of the LEAP+ protocol enjoys timed integrity, while it does not respect timed agreement. When showing that timed agreement fails, we provide an execution trace in which the attacker performs a *replay attack*, despite the security assessment of [15]. To our knowledge this is the first formalisation of a replay attack to LEAP+, in a timed scenario.

2 The Calculus

In Table 1, we define the syntax of `tcryptoCWS` in a two-level structure, a lower one for *processes* and an upper one for *networks*. We use letters $a, b, c, \dots$ for logical names, x, y, z for *variables*, u for *messages*, and v and w for *closed values*, i.e. values that do not contain free variables. We write F^{i} to denote constructors for messages.

The syntax and the semantics of `tcryptoCWS` are parametric with respect to a given *decidable* inference system. Inference systems consist of a set of rules to model the operations on messages by using constructors. For instance, the rules

$$\text{(pair)}\ \frac{v_1 \quad v_2}{\text{pair}(v_1, v_2)} \qquad \text{(fst)}\ \frac{\text{pair}(v_1, v_2)}{v_1} \qquad \text{(snd)}\ \frac{\text{pair}(v_1, v_2)}{v_2}$$

allow us to deal with pairs of values. An instance of the application of rule r to closed messages v_i is denoted as $v_1 \ldots v_k \vdash_r v_0$. Given an inference system, a *deduction function* $\mathcal{D}$ is defined such that, if Φ is a finite set of closed messages, then $\mathcal{D}(\Phi)$ is the set of closed messages that can be deduced from Φ by applying instances of the rules of the inference system.

Networks in `tcryptoCWS` are collections of nodes (which represent devices) running in parallel and using a unique common channel to communicate with each other. The symbol $\mathbf{0}$ denotes the empty network, while $M_1 \mid M_2$ represents the parallel composition of two sub-networks M_1 and M_2. We write $\prod_{i \in I} M_i$ to mean the parallel composition of all M_i, for $i \in I$. We assume that all nodes have the same transmission range (this is a quite common assumption in ad hoc networks [16]). The communication paradigm is *local broadcast*; only nodes located in the range of the transmitter may receive data. We write $n[P]^{\nu}$ for a node named n (the device network address) executing the sequential process P. The tag ν contains (the names of) the neighbours of n.

Processes are sequential and live within the nodes. The symbol nil denotes the skip process. The sender process $!\langle v \rangle.P$ allows to broadcast the value v. The process $\lfloor ?(x).P \rfloor Q$ denotes a receiver with timeout. Intuitively, this process either receives a value, in the current time interval, and then continues as P, or it idles for one time unit, and then continues as Q. Upon successful reception the variable x of P is instantiated with the received message. Similarly, the process $\lfloor \tau.P \rfloor Q$ denotes a process that either perform an internal action, in the current time interval, and then continues as P, or it idles for one time unit, and then continues as Q. The process $\sigma.P$ models sleeping for one time unit. Process $[v_1 = v_2]P; Q$ behaves as P if $v_1 = v_2$, and as Q otherwise. Sometime, we will write $[v_1 = v_2]P$ to mean $[v_1 = v_2]P; \mathsf{nil}$. Process $[v_1 \ldots v_k \vdash_{\mathrm{r}} x]P; Q$ is the inference construct. It tries to infer a message w from the set of premises $\{v_1 \ldots, v_k\}$ through an application of rule $\vdash_r$; if it succeeds, then it behaves as P (where w replaces x), otherwise it behaves as Q. As in the *matching* construct, we will write $[v_1 \ldots v_n \vdash_{\mathrm{r}} x]P$ to mean $[v_1 \ldots v_n \vdash_{\mathrm{r}} x]P; \mathsf{nil}$. In processes $\sigma.P$, $\lfloor ?(x).P_1 \rfloor P_2$, $\lfloor \tau.P_1 \rfloor P_2$ and $!\langle v \rangle.P$ the occurrences of P, P_1 and P_2 are said to be *guarded*. We write $H\langle \tilde{v} \rangle$ to denote a process defined via an equation $H(\tilde{x}) = P$, with $\mid \tilde{x} \mid = \mid \tilde{v} \mid$, where $\tilde{x}$ contains all variables that appear free in P. Defining equations provide *guarded recursion*, since P may contain only guarded occurrences of process identifiers, such as H itself. We assume there are no free variables in our networks. The absence of free variables in networks is trivially maintained as the network evolves. We write $\{^{v}/_{x}\}P$ for the substitution of the variable x with the value v in P.

Given a network M, $\mathsf{nds}(M)$ returns the names of M. If $m \in \mathsf{nds}(M)$, the function $\mathsf{ngh}(m, M)$ returns the set of the neighbours of m in M. Thus, for $M = m[P]^{\nu} \mid N$ it holds that $\mathsf{ngh}(m, M) = \nu$. We write $\mathsf{Env}(M)$ to mean all the nodes of the environment reachable by the network M. The formal definition is: $\mathsf{Env}(M) = \cup_{m \in \mathsf{nds}(M)} \mathsf{ngh}(m, M) \backslash \mathsf{nds}(M)$.

Table 2 Reduction Semantics

$$\text{(R-Bcast)}\ \frac{\forall i \in I \quad n_i \in \nu \quad m \in \nu_i}{m[!\langle v\rangle.P]^{\nu} \mid \prod_{i\in I} n_i[\lfloor ?(x_i).P_i \rfloor Q_i]^{\nu_i} \twoheadrightarrow m[P]^{\nu} \mid \prod_{i\in I} n_i[\{^{v}/_{x_i}\}P_i]^{\nu_i}}$$

$$\text{(R-Internal)}\ \frac{-}{m[\lfloor \tau.P \rfloor Q]^{\nu} \twoheadrightarrow m[P]^{\nu}} \qquad \text{(R-Par)}\ \frac{M \twoheadrightarrow M'}{M \mid N \twoheadrightarrow M' \mid N}$$

$$\text{(R-Sigma)}\ \frac{-}{\begin{array}{c}\prod_{i\in I} n_i[\sigma.P_i]^{\nu_i} \mid \prod_{j\in J} n_j[\lfloor \ldots \rfloor Q_j]^{\nu_j} \mid \prod_{k\in K} n_k[\mathsf{nil}]^{\nu_k} \xrightarrow{\sigma} \\ \prod_{i\in I} n_i[P_i]^{\nu_i} \mid \prod_{j\in J} n_j[Q_j]^{\nu_j} \mid \prod_{k\in K} n_k[\mathsf{nil}]^{\nu_k}\end{array}}$$

$$\text{(R-Struct1)}\ \frac{M \equiv N \quad N \twoheadrightarrow N' \quad N' \equiv M'}{M \twoheadrightarrow M'} \qquad \text{(R-Struct2)}\ \frac{M \equiv N \quad N \xrightarrow{\sigma} N' \quad N' \equiv M'}{M \xrightarrow{\sigma} M'}$$

The dynamics of the calculus is given in terms of a timed *reduction relation* described in Table 2. As usual in process calculi, the *reduction semantics* relies on an auxiliary relation, $\equiv$, called *structural congruence*, defined in Table 3. Basically, $\equiv$ brings the participants of a potential interaction into contiguous positions. In our case, for convenience, structural congruence also takes into account matching and deduction; we recall that our inference systems are always decidable.

The computation proceeds in lock-step: between global synchronisation, denoted with $\xrightarrow{\sigma}$, all nodes proceeds asynchronously by performing actions with no duration, denoted with $\twoheadrightarrow$. Rule (R-Bcast) models the broadcast of a message v. Communication proceeds even if there are no listeners: transmission is a *non-blocking* action. Moreover, communication is *lossy* as some receivers within the range of the transmitter might not receive the message. This may be due to several reasons such as signal interferences or the presence of obstacles. Rule (R-Internal) models local computations. Rules (R-Par), (R-Struct1) and (R-Struct2) are standard in process calculi. Rule (R-Sigma) models the passage of time. We write $\twoheadrightarrow^{\star}$ to denote the reflexive and transitive closure of $\twoheadrightarrow$.

The syntax presented in Table 1 allows to derive inconsistent networks. We rule out networks containing two nodes with the same name. As all nodes have the same transmission range the neighbouring relation is symmetric. Furthermore, in order to guarantee clock synchronisation, we impose network connectivity.

Definition 1 (Well-formedness). *M is said to be* well-formed *if*

- *whenever $M \equiv M_1 \mid m_1[P_1]^{\nu_1} \mid m_2[P_2]^{\nu_2}$ it holds that $m_1 \neq m_2$;*
- *whenever $M \equiv N \mid m_1[P_1]^{\nu_1} \mid m_2[P_2]^{\nu_2}$ with $m_1 \in \nu_2$ it holds that $m_2 \in \nu_1$;*
- *for all $m, n \in \mathsf{nds}(M)$ there are $m_1, \ldots, m_k \in \mathsf{nds}(M)$, such that $m{=}m_1$, $n{=}m_k$, $\nu_j = \mathsf{ngh}(m_j, M)$, for $1{\leq}j{\leq}k$, and $m_i \in \nu_{i+1}$, for $1{\leq}i{\leq}k{-}1$.*

Network well-formedness is preserved at run time.

Proposition 1. *Let M be a well-formed network. If $M \twoheadrightarrow M'$ or $M \xrightarrow{\sigma} M'$ then M' is a well-formed network.*

Table 3 Structural Congruence

$n[[v_1 \dots v_n \vdash_{\mathrm{r}} v]P;Q]^{\nu} \equiv n[P]^{\nu}$ if $v_1 \dots v_n \vdash_r v$	(Struct DedT)
$n[[v_1 \dots v_n \vdash_{\mathrm{r}} v]P;Q]^{\nu} \equiv n[Q]^{\nu}$ if $\nexists\, v.\ v_1 \dots v_n \vdash_r v$	(Struct DedF)
$n[[v = v]P;Q]^{\nu} \equiv n[P]^{\nu}$	(Struct Then)
$n[[v_1 = v_2]P;Q]^{\nu} \equiv n[Q]^{\nu}$ if $v_1 \neq v_2$	(Struct Else)
$n[A\langle\tilde{v}\rangle]^{\nu} \equiv n[\{^{v}/_{x}\}P]^{\nu}$ if $A\langle\tilde{v}\rangle \stackrel{\text{def}}{=} P \quad \wedge \quad \mid \tilde{x} \mid = \mid \tilde{v} \mid$	(Struct Rec)
$M \mid N \equiv N \mid M$	(Struct Par Comm)
$(M \mid N) \mid M' \equiv M \mid (N \mid M')$	(Struct Par Assoc)
$M \mid \mathbf{0} \equiv M$	(Struct Zero Par)
$M \equiv M$	(Struct Refl)
$M \equiv N$ implies $N \equiv M$	(Struct Symm)
$M \equiv M' \wedge M' \equiv M''$ implies $M \equiv M''$	(Struct Trans)
$M \equiv N$ implies $M \mid M' \equiv N \mid M'$, for all M'	(Struct Ctx Par)

3 Time properties

Proposition 2 formalises the deterministic nature of time passing: a network can reach at most one new state by executing the action σ.

Proposition 2 (Time Determinism). *Let M be a well-formed network. If $M \xrightarrow{\sigma\triangleright} M'$ and $M \xrightarrow{\sigma\triangleright} M''$ then $M' \equiv M''$*

The maximal progress property [13] says that processes communicate as soon as a possibility of communication arises.

Proposition 3 (Maximal Progress). *Let M be a well-formed network. If $M \equiv m[!\langle v\rangle.P]^{\nu} \mid N$ then $M \xrightarrow{\sigma\triangleright} M'$ for no network M'.*

Patience guarantees that a process will wait indefinitely until it can communicate [13]. In our setting, this means that if no transmissions can start then it must be possible to execute a σ-action to let time pass.

Proposition 4 (Patience). *Let $M \equiv \prod_{i\in I} m_i[P_i]^{\nu_i}$ be a well-formed network, such that for all $i \in I$ it holds that $m_i[P_i]^{\nu_i} \not\equiv m_i[!\langle v\rangle.Q_i]^{\nu_i}$, then there is a network N such that $M \xrightarrow{\sigma\triangleright} N$.*

4 Labelled Transition Semantics

In Table 4, we provide a Labelled Transition System (LTS) for our calculus. In rule (Snd) a sender dispatches its message to its neighbours ν, and then continues as P. In the label $m!v\triangleright\nu$ the set ν contains the neighbours of m which may receive the message v. In rule (Rcv) a receiver gets a message coming from a neighbour node m, and then evolves into process P, where all the occurrences of the variable x are replaced with the value v. If no message is received in the current time interval, the node n will continue with process Q, according to the rule (σ-Rcv).

Table 4 LTS - Transmission, internal actions, and time passing

$$\text{(Snd)}\ \frac{-}{m[!\langle v\rangle.P]^{\nu} \xrightarrow{m!v\triangleright\nu} m[P]^{\nu}} \qquad \text{(Rcv)}\ \frac{m \in \nu}{n[\lfloor ?(x).P \rfloor Q]^{\nu} \xrightarrow{m?v} n[\{^{v}/_{x}\}P]^{\nu}}$$

$$\text{(RcvEnb)}\ \frac{m \notin \mathsf{nds}(M)}{M \xrightarrow{m?v} M} \qquad \text{(RcvPar)}\ \frac{M \xrightarrow{m?v} M' \quad N \xrightarrow{m?v} N'}{M \mid N \xrightarrow{m?v} M' \mid N'}$$

$$\text{(Bcast)}\ \frac{M \xrightarrow{m!v\triangleright\nu} M' \quad N \xrightarrow{m?v} N' \quad \nu' := \nu \backslash \mathsf{nds}(N)}{M \mid N \xrightarrow{m!v\triangleright\nu'} M' \mid N'}$$

$$\text{(Tau)}\ \frac{-}{m[\lfloor \tau.P \rfloor Q]^{\nu} \xrightarrow{\tau} m[P]^{\nu}} \qquad \text{(TauPar)}\ \frac{M \xrightarrow{\tau} M'}{M \mid N \xrightarrow{\tau} M' \mid N}$$

$$(\sigma\text{-nil})\ \frac{-}{n[\mathsf{nil}]^{\nu} \xrightarrow{\sigma} n[\mathsf{nil}]^{\nu}} \qquad \text{(Delay)}\ \frac{-}{n[\sigma.P]^{\nu} \xrightarrow{\sigma} n[P]^{\nu}}$$

$$(\sigma\text{-Rcv})\ \frac{-}{n[\lfloor ?(x).P \rfloor Q]^{\nu} \xrightarrow{\sigma} n[Q]^{\nu}} \qquad (\sigma\text{-Tau})\ \frac{-}{m[\lfloor \tau.P \rfloor Q]^{\nu} \xrightarrow{\sigma} m[Q]^{\nu}}$$

$$(\sigma\text{-Par})\ \frac{M \xrightarrow{\sigma} M' \quad N \xrightarrow{\sigma} N'}{M \mid N \xrightarrow{\sigma} M' \mid N'} \qquad (\sigma\textbf{-0})\ \frac{-}{\mathbf{0} \xrightarrow{\sigma} \mathbf{0}}$$

In rule (RcvPar) we model the composition of two networks receiving the same message from the same transmitter. Rule (RcvEnb) says that every node can synchronise with an external transmitter m. This rule, together with rule (RcvPar), serves to model message loss. Rule (Bcast) models the propagation of messages on the broadcast channel. Note that in rule (Bcast) we loose track of those neighbours of m that are in N. Rule (Tau) models local computations. Rule (TauPar) serves to propagate internal computations on parallel components. The remaining rules model the passage of time. Rule (Delay) model the delay of a time unit. Rules (σ-nil) and (σ-**0**) are straightforward. Rules (σ-Rcv) models timeout on receivers. Similarly, (σ-Tau) models timeout on internal activities. Rule (σ-Par) models time synchronisation between parallel components. Rules (Bcast) and (TauPar) have their symmetric counterpart.

In Table 5 we report the obvious rules for nodes containing matching, recursion and deduction processes (we recall that only guarded recursion is allowed). In the sequel, we use the metavariable λ to range over the labels: $m!v\triangleright\nu$, $m?v$, τ, and σ.

The LTS-based semantics is consistent with the reduction semantics.

Theorem 1 (Harmony Theorem).

- *If* $M \twoheadrightarrow M'$ *then either* $M \xrightarrow{\tau} \equiv M'$ *or* $M \xrightarrow{m!v\triangleright\nu} \equiv M'$, *for some* m, v, ν.
- *If* $M \xrightarrow{\sigma} M'$ *then* $M \xrightarrow{\sigma} \equiv M'$.
- *If* $M \xrightarrow{m!v\triangleright\nu} M'$ *or* $M \xrightarrow{\tau} M'$ *then* $M \twoheadrightarrow M'$.
- *If* $M \xrightarrow{\sigma} M'$ *then* $M \xrightarrow{\sigma} M'$.

Table 5 LTS - Matching, recursion and deduction

$$\text{(Then)}\ \frac{n[P]^{\nu} \xrightarrow{\lambda} n[P']^{\nu}}{n[[v = v]P; Q]^{\nu} \xrightarrow{\lambda} n[P']^{\nu}} \qquad \text{(Else)}\ \frac{n[Q]^{\nu} \xrightarrow{\lambda} n[Q']^{\nu} \quad v_1 \neq v_2}{n[[v_1 = v_2]P; Q]^{\nu} \xrightarrow{\lambda} n[Q']^{\nu}}$$

$$\text{(Rec)}\ \frac{n[\{\tilde{v}/\tilde{x}\}P]^{\nu} \xrightarrow{\lambda} n[P']^{\nu} \quad H(\tilde{x}) \stackrel{\text{def}}{=} P}{n[H\langle\tilde{v}\rangle]^{\nu} \xrightarrow{\lambda} n[P']^{\nu}}$$

$$\text{(Dtt)}\ \frac{n[\{^{v}/_{x}\}P]^{\nu} \xrightarrow{\lambda} n[P']^{\nu} \quad v_1 \ldots v_n \vdash_{\mathrm{r}} v}{n[[v_1 \ldots v_n \vdash_{\mathrm{r}} x]P; Q]^{\nu} \xrightarrow{\lambda} n[P']^{\nu}} \quad \text{(Dff)}\ \frac{n[Q]^{\nu} \xrightarrow{\lambda} n[Q']^{\nu} \quad \nexists\, v.\ v_1 \ldots v_n \vdash_{\mathrm{r}} v}{n[[v_1 \ldots v_n \vdash_{\mathrm{r}} x]P; Q]^{\nu} \xrightarrow{\lambda} n[Q']^{\nu}}$$

4.1 Behavioural Semantics

We use our LTS to define a standard notion of timed labelled bisimilarity. In general, a bisimulation describes how two terms (in our case networks) can mimic each other actions. Since we are focusing on weak equivalences we have to distinguish between transmissions which may be observed and transmissions which may not be observed by the environment. Thus, we extend the set of rules of Table 4 with the following two rules:

$$\text{(Shh)}\ \frac{M \xrightarrow{m!v\triangleright\emptyset} M'}{M \xrightarrow{\tau} M'} \qquad \text{(Obs)}\ \frac{M \xrightarrow{m!v\triangleright\nu} M' \quad \nu \neq \emptyset}{M \xrightarrow{!v\triangleright\nu} M'}$$

Rule (Shh) models transmissions that cannot be observed because none of the potential receivers is in the environment. Rule (Obs) models a transmission of a message v that can be received (and hence observed) by those nodes of the environment contained in ν. The name of the transmitter is removed as in real networks the identity of the transmitter can only be ensured by using appropriate authentication protocols. Notice that in a derivation tree the rule (Obs) can only be applied at top-level.

In the rest of the paper, the metavariable α ranges over the following actions: $!v\triangleright\nu$, $m?v$, τ, and σ. We adopt the standard notation for weak transitions: $\Rightarrow$ denotes the reflexive and transitive closure of $\xrightarrow{\tau}$; $\stackrel{\alpha}{\Longrightarrow}$ denotes $\Rightarrow \xrightarrow{\alpha} \Rightarrow$; $\stackrel{\hat{\alpha}}{\Longrightarrow}$ denotes $\Rightarrow$ if $\alpha = \tau$ and $\stackrel{\alpha}{\Longrightarrow}$ otherwise.

Definition 2 (Bi-similarity). *A relation $\mathcal{R}$ over well-formed networks is a* simulation *if $M\ \mathcal{R}\ N$ implies that whenever $M \xrightarrow{\alpha} M'$ there is N' such that $N \stackrel{\hat{\alpha}}{\Longrightarrow} N'$ and $M'\ \mathcal{R}\ N'$. A relation $\mathcal{R}$ is called* bisimulation *if both $\mathcal{R}$ and its converse are simulations. We say that M and N are* similar*, written $M \lesssim N$ if there is a simulation $\mathcal{R}$ such that $M\ \mathcal{R}\ N$. We say that M and N are* bisimilar*, written $M \approx N$, if there is a bisimulation $\mathcal{R}$ such that $M\ \mathcal{R}\ N$.*

Our notions of similarity and bisimilarity between networks are congruence, as they are preserved by parallel composition. We only report the result for bisimilarity.

Theorem 2 ($\approx$ is a congruence). *Let M and N be two well-formed networks such that $M \approx N$. Then $M \mid O \approx N \mid O$ for all networks O such that $M \mid O$ and $N \mid O$ are well-formed.*

5 A Framework for the Analysis of Wireless Network Security Protocols

In order to perform a security analysis of wireless network security protocols, we adapt a general schema for the definition of timed security properties, called *Timed Generalized Non-Deducibility on Compositions* (*tGNDC*) [10], a real-time generalisation of *Generalised Non-Deducibility on Compositions* (*GNDC*) [17]. The main idea is the following: a system M is $tGNDC^{\alpha}_{\lhd}$ if and only if for every attacker ATT the composition of the system M with ATT satisfies the timed specification $\alpha(M)$, with respect the timed behavioural relation $\lhd$. The preorder $\lhd$ that we will be using in the following analysis is the *similarity* relation $\lesssim$. An attacker is a network, with some constrains on the data known initially, which tries to attack a protocol by stealing and faking information transmitted on the communication channel. Given a network M, we call $\mathrm{ID}(M)$ the set of messages (closed values) that appears in M.[1] In our setting, a generic attacker of a network M is a collection of nodes in the environment of M, with current knowledge Φ:

$$ATT(\Phi, M) \stackrel{\text{def}}{=} \prod_{n \in \mathsf{Env}(M)} n[P_n]^{\mathsf{nds}(M)} \text{ s.t. } \mathrm{ID}(P_n) \subseteq \mathcal{D}(\Phi) \text{ for all } n.$$

Definition 3 (tGNDC). *Let M be a network, Φ_0 the initial knowledge of the attacker, and α a function between networks defining the property specification for M as the network $\alpha(M)$. We say that M is $tGNDC^{\alpha}_{\lesssim}$ if and only if it holds that $M \mid ATT(\Phi_0, M) \lesssim \alpha(M)$.*

In order to prove that a network is $tGNDC^{\alpha}_{\lesssim}$, we need a timed notion of term stability [10]. Intuitively, a network M is said to be *time-dependent stable* if the attacker cannot increase its knowledge when M runs in the space of a time interval. This requires the notion of execution trace. A *trace* is a sequence of labelled transitions, that we will denote in the standard way. If A is sequence of labels $\alpha_1\alpha_2\ldots\alpha_n$, we write $M \stackrel{\mathrm{A}}{\Longrightarrow} M'$ to mean $M \Rightarrow \xrightarrow{\alpha_1} \Rightarrow \cdots \Rightarrow \xrightarrow{\alpha_n} \Rightarrow M'$. Let $\#^{\sigma}(A)$ be the number of occurrences of σ actions in the sequence A.

Definition 4. *We say that a network M is time-dependent stable wrt a sequence of knowledges $\{\Phi_j\}_{j\geq 0}$, if whenever $M \mid ATT(\Phi_0, M) \stackrel{\mathrm{A}}{\Longrightarrow} M' \mid ATT(\Phi', M')$ and $\#^{\sigma}(\mathrm{A}) = i$, then $\mathcal{D}(\Phi') \subseteq \mathcal{D}(\Phi_i)$.*

When two or more networks are time-dependent stable with respect a certain sequence of knowledges $\{\Phi_j\}_{j\geq 0}$, and they enjoy a certain tGNDC property, then the following compositionality property holds.

Proposition 5. *Let $\{\Phi_j\}_{j\geq 0}$ be a sequence of knowledges, and $\{M_r\}_{1\leq r\leq n}$ a set of time-dependent stable subnetworks, with respect to $\{\Phi_i\}_{i\geq 0}$, such that $M_r \in tGNDC^{\alpha_r(M_r)}_{\lesssim}$, for $1\leq r\leq n$. It follows that:*

1. *$M_1 \mid \ldots \mid M_n$ is time-dependent stable;*
2. *$M_1 \mid \ldots \mid M_n \in tGNDC^{\alpha_1(M_1)\mid\ldots\mid\alpha_n(M_n)}_{\lesssim}$.*

[1] This function can be easily defined along the lines of [10].

As in [10], we formalise two useful timed properties for security protocols in terms of use $tGNDC^{\alpha}_{\triangleleft}$: *timed integrity*, which guarantees that only fresh packets are authenticated, and *timed agreement*, for which agreement must be reached within a certain deadline, otherwise authentications does not hold. More precisely, a protocol is said to enjoy the timed integrity property if, whenever a packet p is authenticated during the time interval i, then this packet was sent at most $i - \delta$ time intervals before. A protocol is said to enjoy the timed agreement property if, whenever a responder n has completed a run of the protocol, apparently with an initiator m, then the latter has initiated the protocol, apparently with the former, at most δ time intervals before, and the two agents agreed on a set of data d.

6 The μTESLA protocol

The μTESLA protocol was designed by Perrig et al. [14] to provide authenticated broadcast for sensor networks. μTESLA calculates the *Message Authentication Code* (MAC) for every packet p_i that it is transmitted by using a different key k_i. These keys are generated with a public *one-way* function F such that, if $k_0, k_1, \ldots, k_n$ are the keys used in the transmission, $F(k_i) = k_{i-1}$, for $1 \leq i \leq n$. The transmission time is split into time intervals and each key is tied to one of them. In each time interval one or more packets are deployed by the sender, each one containing the payload and the MAC calculated with the key bound to that interval. When a new interval starts, the key tied to the previous interval is disclosed to all receivers, so that they can authenticate all the packets previously received. Sender and receivers are *loosely time synchronised* on the key disclosure time to prevent malicious nodes to forge packets with modified payloads. Nodes discard packets containing MACs calculated with already disclosed keys, as those packets could come from an attacker. This key-chain mechanism together with the *one-way* function F, provides two major advantages: (i) it allows to calculate lost keys by simply applying F to the last received key, as many times as necessary; (ii) every node can authenticate the most recent key k_i by means of the last received key k_l (stored in the node memory) and the function F; once authenticated, k_i replaces k_l in the node memory. The protocol works under the assumption that all nodes share an initial key k_0, before the protocol starts.

In Table 6 we provide a specification of the μTESLA protocol in `tcryptoCWS`. Besides the deduction rules for dealing with pairs, we require a deduction rule to build MACs: $v_1 \;\; v_2 \vdash_{\mathrm{mac}} \mathrm{mac}(v_1, v_2)$. Our encoding contains a few simplifications with respect to the original protocol. First of all, there is only one packet sent per time interval, and the sender dispatches one packet and one key alternately. This yields a simpler and easier to read model. Second, our specification does not account for bootstrapping new receivers on the fly.

Let us proceed with the description of our encoding. We essentially define two kinds of processes: senders, S_i, and receivers, $R_i^{k_l}$, where i is the index number of the current key, and k_l is the last authenticated key. Since, we bind one packet with one key, i also refers to the index number of packets. So, a network starting the protocol can be represented as:

$$\mu\mathrm{TESLA} \stackrel{\mathrm{def}}{=} m[S_1]^{\nu_m} \mid n_1[R_1^{k_0}]^{\nu_{n_1}} \mid \ldots \mid n_k[R_1^{k_0}]^{\nu n_k}$$

Table 6 μTESLA specification

Sender:	
$S_i \stackrel{\text{def}}{=} [x_i\ k_i \vdash_{\text{mac}} m_i]$	Calculate MAC using payload and key,
$[m_i\ x_i \vdash_{\text{pair}} p_i]$	build a packet with mac and payload,
$!\langle p_i\rangle.\sigma.$	broadcast packet p_i, synchronise,
$!\langle k_i\rangle.\sigma.$	broadcast key k_i, synchronise,
S_{i+1}	and go to next sending state.
Receiver:	
$R_i^{k_l} \stackrel{\text{def}}{=} \lfloor ?(p).\sigma.P_i^{k_l} \rfloor Q_i^{k_l}$	Receive a packet, synchronise, and go to $P_i^{k_l}$; if timeout go to $Q_i^{k_l}$.
$Q_i^{k_l} \stackrel{\text{def}}{=} \lfloor ?(k).\sigma.R_{i+1}^{k_l} \rfloor R_{i+1}^{k_l}$	Receive a key, synchronise, and go to next receiving state.
$P_i^{k_l} \stackrel{\text{def}}{=} \lfloor ?(k).T_i^{k_l} \rfloor R_{i+1}^{k_l}$	Receive a key k and move to state $T_i^{k_l}$; if timeout go to next receiving state.
$T_i^{k_l} \stackrel{\text{def}}{=} [F^{i-l}(k) = k_l]U_i^{k_i}; \sigma.R_{i+1}^{k_l}$	Check key k using F and the stored key k_l,
$U_i^{k_i} \stackrel{\text{def}}{=} [p \vdash_{\text{fst}} m]$	extract MAC from packet p,
$[p \vdash_{\text{snd}} x]$	extract payload from packet p,
$[x\ k \vdash_{\text{mac}} m']$	calculate mac for packet p,
$[m = m']Z_i^{k_i}; \sigma.R_{i+1}^{k_i}$	verify if it matches with the received one,
$Z_i^{k_i} \stackrel{\text{def}}{=} !\langle \mathsf{auth}_i\rangle.\sigma.R_{i+1}^{k_i}$	if so, authenticate packet i, synchronise, go to next receiving state, and store k_i.

where m is the transmitter and n_i are the receivers. Formally, $\{n_1, \ldots, n_k\} \subseteq \nu_m$, and $m \in \nu_{n_j}$, for $1 \leq j \leq k$. For verification reasons we assume that the environment contains a fresh node *test*, unknown to the attacker, to test successful packet authentication. For simplicity, we assume that this node *cannot* transmit but it can only receive messages. Thus, $test \in \nu_m$ and $test \in \nu_{n_j}$, for $1 \leq j \leq k$.

6.1 Security analysis

Let us prove that μTESLA enjoys *timed integrity*. In particular, we prove that receivers authenticate only packets that have been sent in the previous time interval ($\delta = 1$), in the correct order, even in the presence of the intruder. The key point is that even if the intruder acquires shared keys then it is "too late" to break integrity, i.e. to authenticate packets older than δ. Let us define the *timed integrity* property via an abstraction of the protocol with no possible intruders:

$$\alpha(\mu\text{TESLA}) \stackrel{\text{def}}{=} m[S_1]^{test} \mid n_1[\hat{R}_1]^{test} \mid \ldots \mid n_k[\hat{R}_1]^{test}$$

Here, S_1 is the process defined in Table 6, while $\hat{R}_i \stackrel{\text{def}}{=} \sigma.\lfloor \tau.!\langle \mathsf{auth}_i\rangle.\sigma.\hat{R}_{i+1} \rfloor \hat{R}_{i+1}$. Obviously, here we abstract on receivers.

Let us demonstrate that $\alpha(\mu\text{TESLA})$ enjoys timed integrity with $\delta = 1$.

Lemma 1. – *If* $m[S_1]^{test} \stackrel{A}{\Longrightarrow} \xrightarrow{m!p_i \triangleright test}$ *then* $\#^{\sigma}(\mathrm{A}) = 2(i-1)$.
– *If* $n_j[\hat{R}_1]^{test} \stackrel{A}{\Longrightarrow} \xrightarrow{n_j!\mathsf{auth}_i \triangleright test}$, *for some* $1 \leq j \leq k$, *then* $\#^{\sigma}(\mathrm{A}) = 2(i-1)+1$.

Proposition 6. *If* $\alpha(\mu TESLA) \stackrel{A}{\Longrightarrow} \xrightarrow{m!p_i \triangleright test} \stackrel{B}{\Longrightarrow} \xrightarrow{n_r!\mathsf{auth}_i \triangleright test}$ *then* $\#^{\sigma}(\mathrm{B})=1$.

It should be noticed than any formulation of timed agreement for μTESLA would actually coincide with timed integrity. Thus, Proposition 6 also demonstrates that $\alpha(\mu\text{TESLA})$ enjoys timed agreement, with $\delta = 1$.

Now, we prove that μTESLA satisfies our timed properties. By Proposition 5, it is enough to prove the result for each component. In particular, we notice that the nodes $m[S_1]^{\nu_m}$ and $n_j[R_1]^{\nu_{n_j}}$, for $1 \leq j \leq k$, are time-dependent stable with respect to the following sequence of knowledges:

$$
\begin{array}{l}
\Phi_0 = \{p_1\} \\
\Phi_1 = \Phi_0 \cup \{k_1\} \\
\ldots \\
\Phi_i = \Phi_{i-1} \cup \{p_{j+1}\} \quad \text{if} \quad i = 2j, \qquad j > 0 \\
\Phi_i = \Phi_{i-1} \cup \{k_{j+1}\} \quad \text{if} \quad i = 2j+1, \quad j > 0.
\end{array}
$$

Intuitively, Φ_i consists in Φ_{i-1} together with the set of messages an intruder can get by eavesdropping on a run of the protocol during the time interval i.

Lemma 2. *1.* $m[S_1]^{\nu_m} \in tGNDC_{\lesssim}^{m[S_1]^{test}}$

2. $n_j[R_1]^{\nu_{n_j}} \in tGNDC_{\lesssim}^{n_j[\hat{R}_1]^{test}}$, *for* $1 \leq j \leq k$.

By applying Lemma 2 and Proposition 5 we derive the following result.

Theorem 3 (μTESLA Correctness). $\mu TESLA \in tGNDC_{\lesssim}^{\alpha(\mu TESLA)}$.

7 The LEAP+ protocol

The LEAP+ protocol [15] provides a keying framework to establish authenticated communications. In [15], the authors describe four possible keying mechanisms, each of them providing a different level of security. In our paper, we focus on the *single-hop pairwise shared key* mechanism as it is underlying to all other keying methods. Here, a *network controller* loads each node with an initial key k_{IN} and a *computational efficient* pseudo-random function prf(), before deployment. Then, each node n derives its *master key*: $k_n = \mathrm{prf}(k_{\mathrm{IN}}, n)$.

Let us briefly describe the protocol between an initiator node m and a responder node n. Node m tries to discover its neighbours by broadcasting a *hello* packet that contains its identity, m, and a freshly created nonce, a_i, where i counts the number of attempts of the initiator. When n receives the hello packet from m, it computes its MAC, $h = \mathrm{mac}(k_n, (a_i, n))$, and sends to m a packet containing h and its identity n. If node m does not get the authenticated packet from the responder in due time, it will send a new hello packet with a fresh nonce. When m receives the packet from n, it tries to authenticate it by using n's master key and the last created nonce. If the authentication succeeds, then both nodes proceed in calculating the pairwise key k_{mn} by using the the function prf() as follows: $k_{mn} = \mathrm{prf}(k_n, m)$.

In Table 7 we provide a specification of LEAP+ in `tcryptoCWS`. Besides the standard rules for dealing with pairs, we require the following deduction rules:

$$
(\mathrm{mac})\ \frac{v_1 \quad v_2}{\mathrm{mac}(v_1, v_2)} \qquad\qquad (\mathrm{prf})\ \frac{v_1 \quad v_2}{\mathrm{prf}(v_1, v_2)}
$$

Table 7 LEAP+ specification

Sender at node m:

S_i	$\stackrel{\text{def}}{=}$	$[a_{i-1}\ m \vdash_{\text{prf}} a_i]$	Build a random nonce a_i,
		$[m\ a_i \vdash_{\text{pair}} t]$	build a pair t with m and the nonce a_i,
		$[\mathsf{hello}\ t \vdash_{\text{pair}} p]$	build hello packet using the pair t,
		$!\langle p\rangle.\sigma.P$	broadcast the hello, synchronise and move to P.
P	$\stackrel{\text{def}}{=}$	$\lfloor ?(q).P_1 \rfloor S_{i+1}$	Wait for response from neighbours,
P_1	$\stackrel{\text{def}}{=}$	$[q \vdash_{\text{fst}} n]P_2; \sigma.S_{i+1}$	extract node name n from packet q,
P_2	$\stackrel{\text{def}}{=}$	$[q \vdash_{\text{snd}} h]P_3; \sigma.S_{i+1}$	extract MAC h from packet q,
P_3	$\stackrel{\text{def}}{=}$	$[n\ a_i \vdash_{\text{pair}} t']$	build a pair t' with n and current nonce a_i,
		$[k_{\text{IN}}\ n \vdash_{\text{prf}} k_n]$	calculate n's master key k_n,
		$[k_n\ t' \vdash_{\text{mac}} h']$	calculate MAC h' with k_n and t',
		$[h' = h]P_4; \sigma.S_{i+1}$	if it matches with the received one go to P_4, otherwise steps to next time interval and restart;
P_4	$\stackrel{\text{def}}{=}$	$[k_n\ m \vdash_{\text{prf}} k_{mn}]P_5$	calculate the pairwise key k_{mn},
P_5	$\stackrel{\text{def}}{=}$	$\sigma.OK\_SND$	synchronise, and continue.

Receiver at node n:

R	$\stackrel{\text{def}}{=}$	$\lfloor ?(p).R_1 \rfloor \sigma.R$	Wait for incoming hello packets,
R_1	$\stackrel{\text{def}}{=}$	$[p \vdash_{\text{fst}} p_1]R_2; \sigma.\sigma.R$	extract the first component,
R_2	$\stackrel{\text{def}}{=}$	$[p \vdash_{\text{snd}} p_2]R_3; \sigma.\sigma.R$	extract the second component,
R_3	$\stackrel{\text{def}}{=}$	$[p_1 = \mathsf{hello}]R_4; \sigma.\sigma.R$	check if p is a hello packet,
R_4	$\stackrel{\text{def}}{=}$	$[p_2 \vdash_{\text{fst}} m]R_5; \sigma.\sigma.R$	extract the sender name m,
R_5	$\stackrel{\text{def}}{=}$	$[p_2 \vdash_{\text{snd}} a]R_6; \sigma.\sigma.R$	extract the nonce a,
R_6	$\stackrel{\text{def}}{=}$	$[n\ a \vdash_{\text{pair}} t]$	build a pair t with a and n,
		$[k_n\ t \vdash_{\text{mac}} h]$	calculate MAC h on t with n's master key k_n,
		$[n\ h \vdash_{\text{pair}} q]$	build packet q with node name n and MAC h,
		$\sigma.!\langle q\rangle.R_7$	synchronise, broadcast q, and go to R_7,
R_7	$\stackrel{\text{def}}{=}$	$[k_n\ m \vdash_{\text{prf}} k_{mn}]R_8$	calculate pairwise key k_{mn},
R_8	$\stackrel{\text{def}}{=}$	$\sigma.OK\_RCV$	synchronise and continue.

for calculating MACs and the pseudo random function prf(), respectively. Our specification considers only two nodes, to yield an easier to read model:

$$\text{LEAP+} \stackrel{\text{def}}{=} m[S_1]^{\nu_m} \mid n[R]^{\nu_n}$$

where m is the initiator and n is the responder, with $m \in \nu_n$ and $n \in \nu_m$. This does not lose any generality with respect to the multiple nodes case. Again, for verification reasons, we assume that the environment contains a fresh node *test*, unknown to the attacker, such that $test \in \nu_m$. We recall that the *test* node *cannot* transmits but it can only receive messages.

7.1 Security analysis

In LEAP+, the timed integrity property imposes that the initiator must authenticate only packets sent by the responder in the previous time interval ($\delta = 1$). Let

us slightly modify the specification of LEAP+ to represent this property. Let us define LEAP′+ by replacing the process P_5 with $P_5' \stackrel{\text{def}}{=} \sigma.[\mathsf{auth}\ t \vdash_{\text{pair}} q]!\langle q\rangle.OK\_SND$. Now, the sender process transmit a packet to signal successful authentication. Notice that authenticated messages are always sent by the responder between an hello and authentication message with the same nonce. As a consequence, time integrity imposes that hello messages and authentication messages with the same nonce must differ for at most two time intervals.

In order to show timed integrity, we specify $\alpha_{\text{int}}(\text{LEAP}'+)$ as:

$$\alpha_{\text{int}}(\text{LEAP}'+) \stackrel{\text{def}}{=} m[\hat{S}_1]^{test} \mid n[\hat{R}]^{\emptyset}$$

where $\hat{S}_i \stackrel{\text{def}}{=} !\langle p_i\rangle.\sigma.\lfloor\tau.\sigma.!\langle q_i\rangle.OK\_SND\rfloor\widehat{S_{i+1}}$, for all i, and $\hat{R} \stackrel{\text{def}}{=} \sigma.\hat{R}$, with $p_i = \text{pair}(\mathsf{hello}, \text{pair}(m, a_i))$ and $q_i = \text{pair}(\mathsf{auth}, \text{pair}(m, a_i))$. By construction, $\alpha_{\text{int}}(\text{LEAP}'+)$ satisfies timed integrity.

As we did for μTESLA, we use Proposition 5 to break down the proof into smaller chunks. In particular, we notice that the nodes $m[S_1]^{\nu_m}$ and $n[R]^{\nu_n}$ are time-dependent stable with respect to the sequence $\{\Phi_i\}_{i\geq 0}$, defined as follows:

$$\begin{array}{lll}
\Phi_0 = \{a_1\} & & \\
\Phi_1 = \Phi_0 \cup \{mac(k_n, \text{pair}(n, a_1))\} & & \\
\ldots & & \\
\Phi_i = \Phi_{i-1} \cup \{a_{j+1}\} & \text{if} \quad i = 2j, & j > 0 \\
\Phi_i = \Phi_{i-1} \cup \{mac(k_n, \text{pair}(n, a_{j+1}))\} & \text{if} \quad i = 2j+1, & j > 0.
\end{array}$$

Lemma 3. *1.* $m[S_1]^{\nu_m} \in tGNDC_{\lesssim}^{m[\hat{S}_1]^{test}}$

2. $n[R]^{\nu_n} \in tGNDC_{\lesssim}^{n[\hat{R}]^{\emptyset}}$.

By applying Lemma 3 and Proposition 5 we derive the following result.

Theorem 4 (LEAP+ Timed integrity). $LEAP'+ \in tGNDC_{\lesssim}^{\alpha_{\text{int}}(LEAP'+)}$.

Let us focus now on timed agreement. Again, let us slightly modify the specification of LEAP+ to represent timed agreement. We define LEAP″+ by replacing in LEAP+ the process R_8 with $R_8' \stackrel{\text{def}}{=} \sigma.[\mathsf{end}\ a \vdash_{\text{pair}} r]!\langle r\rangle.OK\_RCV$. Now, the responder signals the end of the protocol. For simplicity, we use the following abbreviation: $r_i = \text{pair}(\mathsf{end}, a_i)$. We also require that the node *test* is among the neighbours of n, i.e. $test \in \nu_n$, so that end messages can be observed. Now, the time agreement property for LEAP+ requires that hello packets p_i, sent by the initiator, and end packets r_i, sent by the responder, (with the same nonce) must differ for at most two time intervals ($\delta = 2$).

Unfortunately, LEAP″+ *does not satisfy* the timed agreement property:

Theorem 5 (LEAP+'s Replay Attack). *LEAP″+ does not satisfy time agreement with* $\delta = 2$.

Proof. LEAP″+ has the following trace:

$$!p_1 \triangleright test.\sigma.\tau.\sigma.\tau.!p_2 \triangleright test.\sigma.!q_1 \triangleright test.\sigma.!r_1 \triangleright test\ .$$

In this trace, the transmission of packets p_1 and r_1 are divided by four σ-actions. This denotes a *replay attack*. More precisely, if we write $ATT(\Phi_i)$ as an abbreviation for $ATT(LEAP''_+, \Phi_i)$, we have the following:

$$
\begin{array}{ll}
m[S_1]^{\nu_m} \mid n[R]^{\nu_n} \mid ATT(\Phi_0) & \xrightarrow{!p_1 \triangleright test} \\
m[\sigma.P]^{\nu_m} \mid n[R]^{\nu_n} \mid ATT(\Phi_0) & \xrightarrow{\sigma} \\
m[P]^{\nu_m} \mid n[\sigma.R]^{\nu_n} \mid ATT(\Phi_1) & \xrightarrow{\tau} \\
m[\{^{p_1}/_{q}\}P_1]^{\nu_m} \mid n[\sigma.R]^{\nu_n} \mid ATT(\Phi_1) & \xrightarrow{\sigma} \\
m[S_2]^{\nu_m} \mid n[R]^{\nu_n} \mid ATT(\Phi_2) & \xrightarrow{\tau} \\
m[S_2]^{\nu_m} \mid n[\sigma.!\langle q_1 \rangle.R_8]^{\nu_n} \mid ATT(\Phi_2) & \xrightarrow{!p_2 \triangleright test} \\
m[\sigma.P]^{\nu_m} \mid n[\sigma.!\langle q_1 \rangle.R_8]^{\nu_n} \mid ATT(\Phi_2) & \xrightarrow{\sigma} \\
m[P]^{\nu_m} \mid n[!\langle q_1 \rangle.R_8]^{\nu_n} \mid ATT(\Phi_3) & \xrightarrow{!q_1 \triangleright test} \\
m[\{^{q_1}/_{q}\}P_1]^{\nu_m} \mid n[R_8]^{\nu_n} \mid ATT(\Phi_3) & \xrightarrow{\sigma} \\
m[S_3]^{\nu_m} \mid n[!\langle r_1 \rangle.OK\_RCV]^{\nu_n} \mid ATT(\Phi_4) & \xrightarrow{!r_1 \triangleright test}
\end{array}
$$

In the first time interval the initiator broadcasts the hello packet p_1 which is lost by the responder and grasped by the attacker. Both nodes move to the second time interval (σ-action). In this time interval, the attacker replay the packet p_1 (τ-action), which is received by the initiator m. This packet however was not what m was expecting. Thus, the network moves to the next time interval (σ-action) where m goes to the next starting process S_2 and n to initial receiving process R. In the third time interval, the attacker broadcasts again the packet p_1 which is successfully received by node n (τ-action), while node m starts again the protocol with a new packet p_2 and a fresh nonce a_2 ($!p_2 \triangleright test$-action). However, packet p_2 is not received by n which is busy in processing p_1. Then, the network moves to the next time interval (σ-action). In the fourth time interval n sends its reply to the packet p_1 ($!q_1 \triangleright test$-action). Node m does not accept the packet q_1 because it contains an old nonce. The network moves to the next time interval, in which n broadcasts its end packet r_1 ($!r_1 \triangleright test$-action) and calculate the pairwise key k_{mn}, while m keeps broadcasting new hello packets. So, agreement can not be reached.

8 Conclusions, Related and Future Work

We have proposed a time broadcasting calculus for wireless network security protocols. Our calculus comes with a well-defined operational semantics and a bisimulation-based behavioural semantics. We have adapted Gorrieri and Martinelli's tGNDC framework to formally study the wireless network security protocols μTESLA and LEAP+. The design of our calculus has been inspired by `tCryptoSPA` [10], a timed cryptographic variant of Milner's CCS. The tGNDC schema for `tCryptoSPA`, has already been used by Gorrieri, Martinelli and Petrocchi [10,11] to study the WMF and the μTESLA protocol. In particular, since they used `tCryptoSPA`, the specification of μTESLA was much more involved and the abstraction for timed integrity was less intuitive.

Several process calculi for wireless systems have been proposed [2,3,4,5,6,7,8,9]. Among these, Nanz and Hankin [3] have designed a calculus for mobile ad hoc networks for specification and security analysis of communication protocols. The

authors provide a decision procedure to check security against fixed intruders known in advance.

It is our intention to apply our framework to study the correctness of a wide range of wireless network security protocols, as for instance, MiniSec [18], and evolutions of LEAP+, such as R-LEAP+ [19] and LEAP++ [20].

References

1. Perrig, A., Stankovic, J.A., Wagner, D.: Security in wireless sensor networks. Communication ACM **47**(6) (2004) 53–57
2. Lanese, I., Sangiorgi, D.: An Operational Semantics for a Calculus for Wireless Systems. Theoretical Computer Science **411** (2010) 1928–1948
3. Nanz, S., Hankin, C.: A Framework for Security Analysis of Mobile Wireless Networks. Theoretical Computer Science **367**(1-2) (2006) 203–227
4. Singh, A., Ramakrishnan, C.R., Smolka, S.A.: A Process Calculus for Mobile Ad Hoc Networks. In: COORDINATION. Volume 5052 of Lecture Notes in Computer Science., Springer (2008) 296–314
5. Merro, M.: An Observational Theory for Mobile Ad Hoc Networks (full paper). Information and Computation **207**(2) (2009) 194–208
6. Godskesen, J.: A Calculus for Mobile Ad Hoc Networks. In: COORDINATION. Volume 4467 of Lecture Notes in Computer Science., Springer (2007) 132–150
7. Ghassemi, F., Fokkink, W., Movaghar, A.: Equational Reasoning on Ad Hoc networks. In: FSEN. Volume 5961 of Lecture Notes in Computer Science., Springer (2009) 113–128
8. Merro, M., Sibilio, E.: A Timed Calculus for Wireless Systems. In: FSEN. Volume 5961 of Lecture Notes in Computer Science., Springer (2010) 228–243
9. Godskesen, J.C., Nanz, S.: Mobility Models and Behavioural Equivalence for Wireless Networks. In: COORDINATION. Volume 5521 of Lecture Notes in Computer Science., Springer (2009) 106–122
10. Gorrieri, R., Martinelli, F.: A simple framework for real-time cryptographic protocol analysis with compositional proof rules. Sci. Comput. Program. **50**(1-3) (2004) 23–49
11. Gorrieri, R., Martinelli, F., Petrocchi, M.: Formal models and analysis of secure multicast in wired and wireless networks. J. Autom. Reasoning **41**(3-4) (2008) 325–364
12. Milner, R.: Communication and Concurrency. Prentice Hall (1989)
13. Hennessy, M., Regan, T.: A Process Algebra for Timed Systems. Information and Computation **117**(2) (1995) 221–239
14. Perrig, A., Szewczyk, R., Tygar, J.D., Wen, V., Culler, D.: Spins: Security Protocols for Sensor Networks. Wireless Networks **8**(5) (2002) 521–534
15. Zhu, S., Setia, S., Jajodia, S.: Leap+: Efficient security mechanisms for large-scale distributed sensor networks. ACM Transactions on Sensor Networks **2**(4) (2006) 500–528
16. Misra, S., Woungag, I.: Guide to Wireless Ad Hoc Networks. Computer Communications and Networks. Springer London (2009)
17. Focardi, R., Martinelli, F.: A uniform approach for the definition of security properties. In: World Congress on Formal Methods. Volume 1708 of Lecture Notes in Computer Science., Springer (1999) 794–813
18. Luk, M., Mezzour, G., Perrig, A., Gligor, V.: Minisec: a secure sensor network communication architecture. In: IPSN. (2007) 479–488
19. Blackshear, S., Verma, R.: R-Leap+: randomizing Leap+ key distribution to resist replay and jamming attacks. In: SAC, ACM Press (2010) 1985–1992
20. Lim, C.H.: Leap++: A robust key establishment scheme for wireless sensor networks. In: ICDCS, IEEE Computer Society (2008) 376–381