A semantic analysis of wireless network security protocols

Damiano Macedonio and Massimo Merro

Dipartimento di Informatica, Università degli Studi di Verona, Italy

Abstract Gorrieri and Martinelli's *tGNDC* schema is a well-known general framework for the formal verification of security protocols in a concurrent scenario. We generalise the *tGNDC* schema to verify wireless network security protocols. Our generalisation relies on a simple *timed broadcasting process calculus* whose operational semantics is given in terms of a labelled transition system which is used to derive a standard *simulation theory*. We apply our *tGNDC* framework to perform a security analysis of LiSP, a well-known *key management protocol* for wireless sensor networks.

1 Introduction

Wireless communication has become very popular in industry, business, commerce, and in everyday life. Wireless technology spans from user applications, such as personal area networks, ambient intelligence, and wireless local area networks, to real-time applications, such as cellular and ad hoc networks.

In this paper, we adopt a process calculus approach to formalise and verify wireless network protocols. We propose a simple *timed broadcasting process calculus*, called `aTCWS`, for modelling wireless networks. The time model we adopt is known as the *fictitious clock* approach (see e.g. [7]): A global clock is supposed to be updated whenever all nodes agree on this, by globally synchronising on a special timing action σ.[1] Broadcast communications span over a limited area, called *transmission range*. Both broadcast actions and internal actions are assumed to take no time. This is a reasonable assumption whenever the duration of those actions is negligible with respect to the chosen time unit. The operational semantics of our calculus is given in terms of a labelled transition semantics in the SOS style of Plotkin. The calculus enjoys standard time properties, such as: *time determinism*, *maximal progress*, and *patience* [7]. The labelled transition semantics is used to derive a (weak) *simulation theory* which can be easily *mechanised*. Based on our simulation theory, we generalise Gorrieri and Martinelli's *timed Generalized Non-Deducibility on Compositions* (*tGNDC*) schema [5,6], a well-known general framework for the formal verification of timed security properties. The basic idea of *tGNDC* is the following: a protocol M satisfies $tGNDC^{\rho(M)}$ if the presence of an arbitrary *attacker* does not affect the behaviour of M with

[1] Time synchronisation relies on some clock synchronisation protocol [16].

respect to the abstraction $\rho(M)$. By varying $\rho(M)$ it is possible to express different timed security properties for the protocol M. Examples are the *timed integrity* property, which ensures the freshness of authenticated packets, and the *timed agreement* property, when agreement between two parties must be reached within a certain deadline. In this paper, we will focus on the first property. In order to avoid the universal quantification over all possible attackers when proving *tGNDC* properties, we provide a sound proof technique based on the notion of *the most powerful attacker*.

As a main application of our theory, we provide a formal specification of LiSP [13], a well-known key management protocol for *wireless sensor networks* that, through an efficient mechanism of re-keying, provides a good trade-off between resource consumption and network security. We perform our *tGNDC*-based analysis on LiSP showing that old packets can be authenticated as a consequence of a *replay attack*. To our knowledge this attack has never appeared in the literature. Then, we formally prove that similar attacks can be avoided if nonces are added to the original LiSP protocol.

Related Work A number of process calculi have been proposed for modelling different aspects of wireless systems [8,15,9,3,2,10,4]. The paper [12] proposes an algebraic approach to perform security analysis of communication protocols for ad hoc networks. The paper [1] proposes a first formalisation of *tGNDC* in our setting and a security analysis of the authentication protocols μTESLA [14] and LEAP+ [17]. The protocol μTESLA has also been studied within the process algebra `tCryptoSPA` [5,6], an extension of Milner's CCS, where node distribution, local broadcast communication, and message loss are codified in terms of point-to-point transmission and a (discrete) notion of time. As a consequence, specifications and security analyses of wireless network protocols in `tCryptoSPA` are much more complicated than ours.

2 The Calculus

In Table 1, we provide the syntax of our *applied Timed Calculus for Wireless Systems*, in short `aTCWS`, in a two-level structure: A lower one for *processes* and an upper one for *networks*. We assume a set *Nds* of logical node names, ranged over by letters m, n. *Var* is the set of *variables*, ranged over by x, y, z. We define *Val* to be the set of values, and *Msg* to be the set of *messages*, i.e., closed values that do not contain variables. Letters $u, u_1 \ldots$ range over *Val*, and $w \ldots$ range over *Msg*. We assume a class of *message constructors* ranged over by F^i.

Both syntax and operational semantics of `aTCWS` are parametric with respect to a given *decidable* inference system, i.e. a set of rules to model operations on messages by using constructors. For instance, the rules

$$\text{(pair)}\ \frac{w_1 \quad w_2}{\mathrm{pair}(w_1, w_2)} \qquad \text{(fst)}\ \frac{\mathrm{pair}(w_1, w_2)}{w_1} \qquad \text{(snd)}\ \frac{\mathrm{pair}(w_1, w_2)}{w_2}$$

allow us to deal with pairs of values. We write $w_1 \ldots w_k \vdash_{\mathrm{r}} w_0$ to denote an application of rule r to the closed values $w_1 \ldots w_k$ to infer w_0. Given an inference

Table 1 Syntax of `aTCWS`

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

system, the *deduction function* $\mathcal{D} : 2^{Msg} \to 2^{Msg}$ associates a (finite) set ϕ of messages to the set $\mathcal{D}(\phi)$ of messages that can be deduced from ϕ, by applying instances of the rules of the inference system.

Networks are collections of nodes running in parallel and using a unique common channel to communicate with each other. All nodes have the same transmission range (this is a quite common assumption in models for ad hoc networks [11]). 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 neighbours of n ($\nu \subseteq Nds \setminus \{n\}$). Our wireless networks have a fixed topology as node mobility is not relevant to our analysis.

Processes are sequential and live within the nodes. In the processes $!\langle w\rangle.P$, $\lfloor ?(x).P \rfloor Q$, $\lfloor \sum_{i\in I} \tau.P_i \rfloor Q$ and $\sigma.Q$, the occurrences of P, P_i and Q are said to be *guarded*; the occurrences of Q are also said to be *time-guarded*. In the processes $\lfloor ?(x).P \rfloor Q$ and $[w_1 \ldots w_n \vdash_{\mathrm{r}} x]P$ the variable x is said to be *bound* in P. A variable which is not bound is said to be *free*. We adopt the standard notion of α-conversion on bound variables and we identify processes up to α-conversion. We assume there are no free variables in our networks. The absence of free variables will be maintained as networks evolve. We write $\{^{w}/_{x}\}P$ for the substitution of the variable x with the message w in P. We write $H\langle w_1, \ldots, w_k\rangle$ to denote a recursive process H defined via an equation $H(x_1, \ldots, x_k) = P$, where (i) the tuple $x_1, \ldots, x_k$ contains all the variables that appear free in P, and (ii) P contains only guarded occurrences of the process identifiers, such as H itself. We say that recursion is *time-guarded* if P contains only time-guarded occurrences of the process identifiers.

We report some notational *conventions*. We write $\prod_{i\in I} M_i$ to mean the parallel composition of all M_i, for $i \in I$. We identify $\prod_{i\in I} M_i = \mathbf{0}$ if $I = \emptyset$. The

Table 2 LTS - Transmissions, internal actions and time passing

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

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

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

$$\text{(Tau)}\ \frac{h \in I}{m[\lfloor \textstyle\sum_{i\in I} \tau.P_i \rfloor Q]^{\nu} \xrightarrow{\tau} m[P_h]^{\nu}} \qquad \text{(TauPar)}\ \frac{M \xrightarrow{\tau} M'}{M \mid N \xrightarrow{\tau} M' \mid N}$$

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

$$(\sigma\text{-Rcv})\ \frac{-}{n[\lfloor ?(x).P \rfloor Q]^{\nu} \xrightarrow{\sigma} n[Q]^{\nu}} \qquad (\sigma\text{-Sum})\ \frac{-}{m[\lfloor \textstyle\sum_{i\in I} \tau.P_i \rfloor Q]^{\nu} \xrightarrow{\sigma} m[Q]^{\nu}}$$

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

process $[w_1 = w_2]P$ is an abbreviation for $[w_1 = w_2]P; \mathsf{nil}$. Similarly, we will write $[w_1 \dots w_n \vdash_{\mathrm{r}} x]P$ to mean $[w_1 \dots w_n \vdash_{\mathrm{r}} x]P; \mathsf{nil}$.

In the sequel, we will make use of a standard notion of structural congruence to abstract over processes that differ for minor syntactic differences.

Definition 1. Structural congruence *over networks, written* $\equiv$*, is defined as the smallest equivalence relation, preserved by parallel composition, which is a commutative monoid with respect to parallel composition and internal choice, and for which* $n[H\langle\tilde{w}\rangle]^{\nu} \equiv n[\{^{\tilde{w}}/_{\tilde{x}}\}P]^{\nu}$*, if* $H(\tilde{x}) = P$*.*

Here, we provide some definitions that will be useful in the remainder of the paper. Given a network M, $\mathsf{nds}(M)$ returns the node names of M. More formally, $\mathsf{nds}(\mathbf{0}) = \emptyset$, $\mathsf{nds}(n[P]^{\nu}) = \{n\}$ and $\mathsf{nds}(M_1 \mid M_2) = \mathsf{nds}(M_1) \cup \mathsf{nds}(M_2)$. For $m \in \mathsf{nds}(M)$, the function $\mathsf{ngh}(m, M)$ returns the set of the neighbours of m in M. Thus, if $M \equiv m[P]^{\nu} \mid N$ then $\mathsf{ngh}(m, M) = \nu$. We write $\mathsf{Env}(M)$ to mean all the nodes of the environment reachable by the network M. Formally, $\mathsf{Env}(M) = \cup_{m\in\mathsf{nds}(M)}\mathsf{ngh}(m, M) \setminus \mathsf{nds}(M)$.

The syntax provided in Table 1 allows us to derive networks which are somehow ill-formed. The following definition identifies well-formed networks.

Definition 2 (Well-formedness). *M is said to be* well-formed *if (i) $M \equiv N \mid m_1[P_1]^{\nu_1} \mid m_2[P_2]^{\nu_2}$ implies $m_1 \neq m_2$; (ii) $M \equiv N \mid m_1[P_1]^{\nu_1} \mid m_2[P_2]^{\nu_2}$, with $m_1 \in \nu_2$, implies $m_2 \in \nu_1$; (iii) for all $m, n \in \mathsf{nds}(M)$ there are $m_1, \dots, m_k \in \mathsf{nds}(M)$, such that $m{=}m_1$, $n{=}m_k$, $m_i \in \mathsf{ngh}(m_{i+1}, M)$, for $1 \leq i \leq k{-}1$.*

In Table 2, we provide a labelled transition system (LTS) for `aTCWS` in the SOS style of Plotkin. Intuitively, the computation proceeds in lock-steps: between

Table 3 LTS - Matching, recursion and deduction

$$\text{(Then)}\ \frac{n[P]^\nu \xrightarrow{\lambda} n[P']^\nu}{n[[w=w]P;Q]^\nu \xrightarrow{\lambda} n[P']^\nu} \qquad \text{(Else)}\ \frac{n[Q]^\nu \xrightarrow{\lambda} n[Q']^\nu \quad w_1 \neq w_2}{n[[w_1=w_2]P;Q]^\nu \xrightarrow{\lambda} n[Q']^\nu}$$

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

$$\text{(DT)}\ \frac{n[\{^w/_x\}P]^\nu \xrightarrow{\lambda} n[R]^\nu \ \ w_1 \ldots w_n \vdash_r w}{n[[w_1 \ldots w_n \vdash_{\mathrm{r}} x]P;Q]^\nu \xrightarrow{\lambda} n[R]^\nu} \qquad \text{(DF)}\ \frac{n[Q]^\nu \xrightarrow{\lambda} n[R]^\nu \quad \nexists\, w.\ w_1 \ldots w_n \vdash_r w}{n[[w_1 \ldots w_n \vdash_{\mathrm{r}} x]P;Q]^\nu \xrightarrow{\lambda} n[R]^\nu}$$

every global synchronisation all nodes proceeds asynchronously by performing actions with no duration, which represent either broadcast or input or internal actions. 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.

The metavariable λ ranges over the set of labels $\{\tau, \sigma, m!w{\triangleright}\nu, m?w\}$ denoting internal action, time passing, broadcasting and reception. Let us comment on the transition rules of Table 2. In rule (Snd) a sender m dispatches a message w to its neighbours ν, and then continues as P. In rule (Rcv) a receiver n gets a message w coming from a neighbour node m, and then evolves into process P, where all the occurrences of the variable x are replaced with w. If no message is received in the current time slot, a timeout fires and the node n will continue with process Q, according to the rule (σ-Rcv). The rule (RcvPar) models 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. Notice that a node $n[\lfloor ?(x).P \rfloor Q]^\nu$ might execute rule (RcvEnb) instead of rule (Rcv). This is because a potential receiver may miss a message for several reasons (internal misbehaving, interferences, weak radio signal, etc); in this manner we model message loss. Rule (Bcast) models the propagation of messages on the broadcast channel. Note that this rule looses track of the neighbours of m that are in N. Thus, in the label $m!w{\triangleright}\nu$ the set ν always contains the neighbours of m which can receive the message w. The remaining rules are straightforward. Rules (Bcast) and (TauPar) have their symmetric counterparts. Table 3 reports the standard rules for nodes containing matching, recursion or deduction.

Below, we report a number of basic properties of our LTS.

Proposition 1. *Let M, M_1 and M_2 be well-formed networks.*

1. *$m \notin \mathsf{nds}(M)$ if and only if $M \xrightarrow{m?w} N$, for some network N.*
2. *$M_1 \mid M_2 \xrightarrow{m?w} N$ if and only if there are N_1 and N_2 such that $M_1 \xrightarrow{m?w} N_1$, $M_2 \xrightarrow{m?w} N_2$ with $N = N_1 \mid N_2$.*
3. *If $M \xrightarrow{m!w{\triangleright}\mu} M'$ then $M \equiv m[!\langle w\rangle.P]^\nu \mid N$, for some m, ν, P and N such that $m[!\langle w\rangle.P]^\nu \xrightarrow{m!w{\triangleright}\nu} m[P]^\nu$, $N \xrightarrow{m?w} N'$, $M' \equiv m[P]^\nu \mid N'$ and $\mu = \nu \setminus \mathsf{nds}(N)$.*

4. *If* $M \xrightarrow{\tau} M'$ *then* $M \equiv m[\lfloor \sum_{i\in I} \tau.P_i \rfloor Q]^\nu \mid N$, *for some* m, ν, P_i, Q *and* N *such that* $m[\lfloor \sum_{i\in I} \tau.P_i \rfloor Q]^\nu \xrightarrow{\tau} m[P_h]^\nu$, *for some* $h \in I$, *and* $M' \equiv m[P_h]^\nu \mid N$.
5. $M_1 \mid M_2 \xrightarrow{\sigma} N$ *if and only if there are* N_1 *and* N_2 *such that* $M_1 \xrightarrow{\sigma} N_1$, $M_2 \xrightarrow{\sigma} N_2$ *and* $N = N_1 \mid N_2$.

Proposition 2. *Let* M *be well-formed. If* $M \xrightarrow{\lambda} M'$ *then* M' *is well-formed.*

Based on the LTS of Section 2, we define a standard notion of *timed labelled similarity* for `aTCWS`. We distinguish between the transmissions which may be observed and those which may not be observed by the environment. We extend the set of rules of Table 2 with the following two rules:

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

Rule (Shh) models transmissions that cannot be observed because none of the potential receivers is in the environment. Rule (Obs) models transmissions that can be received (and hence observed) by those nodes of the environment contained in ν. Notice that the name of the transmitter is removed from the label. This is motivated by the fact that nodes may refuse to reveal their identities, e.g. for security reasons or limited sensory capabilities in perceiving these identities.

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

Definition 3 (Similarity). *A relation* $\mathcal{R}$ *over well-formed networks is a* simulation *if* $M\ \mathcal{R}\ N$ *and* $M \xrightarrow{\alpha} M'$ *imply there is* N' *such that* $N \overset{\hat{\alpha}}{\Longrightarrow} N'$ *and* $M'\ \mathcal{R}\ N'$. *We write* $M \lesssim N$, *if there is a simulation* $\mathcal{R}$ *such that* $M\ \mathcal{R}\ N$.

Our notion of of similarity between networks is a pre-congruences, as it is preserved by parallel composition.

Theorem 1. *Let* M *and* N *be two well-formed networks such that* $M \lesssim N$. *Then* $M \mid O \lesssim N \mid O$ *for all* O *such that* $M \mid O$ *and* $N \mid O$ *are well-formed.*

3 A *tGNDC* schema for Wireless Networks

Gorrieri and Martinelli [5] have proposed a general schema for the definition of timed security properties, called *timed Generalized Non-Deducibility on Compositions* (*tGNDC*). Basically, a system M is $tGNDC^{\rho(M)}$ if for any attacker A

$$M \mid A \lesssim \rho(M)$$

i.e. the composed system $M \mid A$ satisfies the abstraction $\rho(M)$.

A wireless protocol involves a set of nodes which may be potentially under attack, depending on the proximity to the attacker. This means that, in general, the *attacker* of a protocol M is a distinct network A of possibly colluding nodes. For the sake of compositionality, we assume that each node of the protocol is attacked by exactly one node of A.

Definition 4. *We say that $\mathcal{A}$ is a set of attacking nodes for the network M if and only if $|\mathcal{A}| = \mathsf{nds}(M)$ and $\mathcal{A} \cap (\mathsf{nds}(M) \cup \mathsf{Env}(M)) = \emptyset$.*

During the execution of the protocol an attacker may increase its initial knowledge by grasping messages sent by the parties, according to Dolev-Yao constrains. The knowledge of a network is expressed by the set of messages that the network can manipulate. Thus, we write $\mathrm{msg}(M)$ (resp. $\mathrm{msg}(P)$) to denote the set of the messages appearing in the network M (resp. in the process P). To ensure that attackers cannot prevent the passage of time, in the following definition we denote Prc_{wt} the set of processes in which summations are finite-indexed and recursive definitions are time-guarded.

Definition 5 (Attacker). *Let M be a network, with $\mathsf{nds}(M) = \{m_1, ..., m_k\}$. Let $\mathcal{A} = \{a_1, \ldots, a_k\}$ be a set of attacking nodes for M. We define the set of* attackers *of M with initial knowledge $\phi_0 \subseteq Msg$ as:*

$$\mathbb{A}^{\phi_0}_{\mathcal{A}/M} \stackrel{\text{def}}{=} \left\{ \prod_{i=1}^{k} a_i[Q_i]^{\mu_i} : Q_i \in Prc_{\mathrm{wt}},\, \mathrm{msg}(Q_i) \subseteq \mathcal{D}(\phi_0),\, \mu_i = (\mathcal{A} \setminus a_i) \cup m_i \right\}.$$

Sometimes, for verification reasons, we will be interested in observing part of the protocol M under examination. For this purpose, we assume that the environment contains a fresh node $obs \notin \mathsf{nds}(M) \cup \mathsf{Env}(M) \cup \mathcal{A}$, that we call the 'observer', unknown to the attacker. For convenience, the observer *cannot* transmit: it can only receive messages.

Definition 6. *Let $M = \prod_{i=1}^{k} m_i[P_i]^{\nu_i}$. Given a set $\mathcal{A} = \{a_1, \ldots, a_k\}$ of attacking nodes for M and fixed a set $\mathcal{O} \subseteq \mathsf{nds}(M)$ of nodes to be observed, we define:*

$$M^{\mathcal{A}}_{\mathcal{O}} \stackrel{\text{def}}{=} \prod_{i=1}^{k} m_i[P_i]^{\nu'_i} \quad \textit{where} \quad \nu'_i \stackrel{\text{def}}{=} \begin{cases} (\nu_i \cap \mathsf{nds}(M)) \cup a_i \cup obs & \textit{if } m_i \in \mathcal{O} \\ (\nu_i \cap \mathsf{nds}(M)) \cup a_i & \textit{otherwise.} \end{cases}$$

This definition expresses that (i) every node m_i of the protocols has a dedicated attacker located at a_i, (ii) network and attacker are considered in *isolation*, without any external interference, (iii) only *obs* can observe the behaviour of nodes in $\mathcal{O}$, (iv) node *obs* does not interfere with the protocol as it cannot transmit, (v) the behaviour of the nodes in $\mathsf{nds}(M) \setminus \mathcal{O}$ is not observable.

We can now formalise the *tGNDC* family properties as follows.

Definition 7 (*tGNDC* for wireless networks). *Given a network M, an initial knowledge ϕ_0, a set $\mathcal{O} \subseteq \mathsf{nds}(M)$ of nodes under observation and an abstraction $\rho(M)$, representing a security property for M, we write $M \in tGNDC^{\rho(M)}_{\phi_0,\mathcal{O}}$ if and only if for all sets $\mathcal{A}$ of attacking nodes for M and for every $A \in \mathbb{A}^{\phi_0}_{\mathcal{A}/M}$ it holds that $M^{\mathcal{A}}_{\mathcal{O}} \mid A \lesssim \rho(M)$.*

It should be noticed that when showing that a system M is $tGNDC^{\rho(M)}_{\phi_0,\mathcal{O}}$, the universal quantification on attackers required by the definition makes the proof quite involved. Thus, we look for a sufficient condition which does not make use of the universal quantification. For this purpose, we rely on a timed notion of term stability [5]. Intuitively, a network M is said to be *time-dependent stable* if the attacker cannot increase its knowledge in a indefinite way when M runs in the space of a time slot. Thus, we can predict how the knowledge of the attacker evolves at each time slot. First, we need a formalisation of computation. For $\Lambda{=}\alpha_1 \ldots \alpha_n$, we write $\stackrel{\Lambda}{\Longrightarrow}$ to denote $\Rightarrow \xrightarrow{\alpha_1} \Rightarrow \ \ldots \ \Rightarrow \xrightarrow{\alpha_n} \Rightarrow$. In order to count how many time slots embraces an execution trace Λ, we define $\#^{\sigma}(\Lambda)$ to be the number of occurrences of σ-actions in Λ.

Definition 8 (Time-dependent stability). *A network M is said to be* time-dependent stable *with respect to a sequence of knowledge $\{\phi_j\}_{j\geq 0}$ if whenever $M^{\mathcal{A}}_{\mathsf{nds}(M)} \,|\, A \stackrel{\Lambda}{\Longrightarrow} M' \,|\, A'$, where $\mathcal{A}$ is a set of attacking nodes for M, $\#^{\sigma}(\Lambda) = j$, $A \in \mathbb{A}^{\phi_0}_{\mathcal{A}/M}$ and $\mathsf{nds}\,(M') = \mathsf{nds}\,(M)$, then $\mathrm{msg}(A') \subseteq \mathcal{D}(\phi_j)$.*

The set of messages ϕ_j expresses the knowledge of the attacker at the end of the j-th time slot. Time-dependent stability is the crucial notion that allows us to introduce the notion of most general attacker. Intuitively, given a sequence of knowledge $\{\phi_j\}_{j\geq 0}$ and a network M, with $\mathcal{P} = \mathsf{nds}\,(M)$, we pick a set $\mathcal{A} = \{a_1, \ldots, a_k\}$ of attacking nodes for M and we define the top attacker $\textsc{Top}^{\phi_j}_{\mathcal{A}/\mathcal{P}}$ as the network which at (the beginning of) the j-th time slot is aware of the knowledge (derivable) from ϕ_j.

Definition 9 (Top Attacker). *Let M be a network with $\mathcal{P}{=}\mathsf{nds}\,(M){=}\bigcup_{i=1}^{k} m_i$. Let $\mathcal{A} = \{a_1, \ldots, a_k\}$ be a set of attacking nodes for M, and $\{\phi_j\}_{j\geq 0}$ a sequence of knowledge. We define:*

$$\textsc{Top}^{\phi_j}_{\mathcal{A}/\mathcal{P}} \stackrel{\text{def}}{=} \prod_{i=1}^{k} a_i[\mathrm{T}_{\phi_j}]^{m_i} \quad \textit{where} \quad \mathrm{T}_{\phi_j} \stackrel{\text{def}}{=} \lfloor \textstyle\sum_{w\in\mathcal{D}(\phi_j)} \tau.!\langle w\rangle.\mathrm{T}_{\phi_j} \rfloor \, \mathrm{T}_{\phi_{j+1}} \ .$$

Basically, from j-th time slot onwards, $\textsc{Top}^{\phi_j}_{\mathcal{A}/\mathcal{P}}$ can *replay* any message in $\mathcal{D}(\phi_j)$ to the network under attack. Moreover, every attacking node a_i can send messages to the corresponding node m_i, but, unlike the attackers of Definition 5, it does not need to communicate with the other nodes in $\mathcal{A}$ as it already owns the full knowledge of the system at time j.

Top attackers are strong enough to guarantee *tGNDC*.

Theorem 2 (Criterion for *tGNDC*). *Let M be time-dependent stable with respect to a sequence $\{\phi_j\}_{j\geq 0}$, $\mathcal{A}$ be a set of attacking nodes for M and $\mathcal{O} \subseteq \mathsf{nds}\,(M) = \mathcal{P}$. Then $M^{\mathcal{A}}_{\mathcal{O}} \mid \textsc{Top}^{\phi_0}_{\mathcal{A}/\mathcal{P}} \lesssim N$ implies $M \in tGNDC^{N}_{\phi_0,\mathcal{O}}$.*

Top attackers can be employed to reason in a compositional manner.

Theorem 3 (Compositionality). *Let $M = M_1 \mid \ldots \mid M_k$ be time-dependent stable with respect to a sequence of knowledge $\{\phi_j\}_{j\geq 0}$. Let $\mathcal{A}_1, \ldots, \mathcal{A}_k$ be disjoint sets of attacking nodes for $M_1, \ldots, M_k$, respectively. Let $\mathcal{O}_i \subseteq \mathsf{nds}\,(M_i) = \mathcal{P}_i$, for $1 \leq i \leq k$. Then, $(M_i)^{\mathcal{A}_i}_{\mathcal{O}_i} \mid \textsc{Top}^{\phi_0}_{\mathcal{A}_i/\mathcal{P}_i} \lesssim N_i$, for $1 \leq i \leq k$, implies $M \in tGNDC^{N_1|\ldots|N_k}_{\phi_0,\mathcal{O}_1\cup\ldots\cup\mathcal{O}_k}$.*

4 A security analysis of LiSP

LiSP [13] is a well-known key management protocol for *wireless sensor networks*. A LiSP network consists of a *Key Server* (KS) and a set of *sensor nodes* $m_1, \ldots, m_k$. The protocol assumes a *one way function* F, pre-loaded in every node of the system, and employs two different key families: (i) a set of *temporal keys* $k_0, \ldots, k_n$, computed by KS by means of F, and used by all nodes to encrypt/decrypt data packets; (ii) a set of *master keys* $k_{\text{KS}:m_j}$, one for each node m_j, for unicast communications between m_j and BS. The transmission time is split into *time intervals*, each of them is Δ_{refresh} time units long. Thus, each temporal key is tied to a time interval and renewed every Δ_{refresh} time units. At a time interval i, the temporal key k_i is shared by all sensor nodes and it is used for data encryption. Key renewal relies on *loose node time synchronisation* among nodes. Each node stores a subset of temporal keys in a *buffer* of a fixed size, say s with $s << n$.

The LiSP protocol consists of the following phases.

Initial Setup. At the beginning, KS randomly chooses a key k_n and computes a sequence of temporal keys $k_0, \ldots, k_n$, by using the function F, as $k_i := F(k_{i+1})$. Then, KS waits for reconfiguration requests from nodes. More precisely, when KS receives a reconfiguration request from a node m_j, at time interval i, it unicasts the packet InitKey:

$$\text{KS} \rightarrow m_j \ : \ \text{enc}(k_{\text{KS}:m_j},\ (s \mid k_{s+i} \mid \Delta_{\text{refresh}})) \ \mid \ \text{hash}(s \mid k_{s+i} \mid \Delta_{\text{refresh}}) \ .$$

The operator $\text{enc}(k, p)$ represents the encryption of p by using the key of k, while $\text{hash}(p)$ generates a message digest for p by means of a cryptographic hash function used to check the integrity of the packet p. When m_j receives the InitKey packet, it computes the sequence of keys $k_{s+i-1}, k_{s+i-2}, \ldots, k_i$ by several applications of the function F to k_{s+i}. Then, it activates k_i for data encryption and it stores the remaining keys in its local buffer; finally it sets up a *ReKeyingTimer* to expires after $\Delta_{\text{refresh}}/2$ time units (this value applies only for the first rekeying).

Re-Keying. At each time interval i, with $i \leq n$, KS employs the active encryption key k_i to encode the key k_{s+i}. The resulting packet is broadcast as an UpdateKey packet:

$$\text{KS} \rightarrow * \ : \ \text{enc}(k_i,\ k_{s+i}) \ .$$

When a node receives an UpdateKey packet, it tries to authenticate the key received in the packet; if the node succeeds in the authentication then it recovers all keys that have been possibly lost and updates its key buffer. When the time interval i elapses, every node discards k_i, activates the key k_{i+1} for data encryption, and sets up the *ReKeyingTimer* to expire after Δ_{refresh} time units for future key switching (after the first time, switching happens every Δ_{refresh} time units).

Authentication and Recovery of Lost Keys. The one-way function F is used to authenticate and recover lost keys. If l is the number of stored keys in a buffer

of size s, with $l \leq s$, then $s-l$ represents the number of keys which have been lost by the node. When a sensor node receives an UpdateKey packet carrying a new key k, it calculates $F^{s-l}(k)$ by applying $s-l$ times the function F. If the result matches with the last received temporal key, then the node stores k in its buffer and recovers all lost keys.

Reconfiguration. When a node m_j joins the network or misses more than s temporal keys, then its buffer is empty. Thus, it sends a RequestKey packet in order to request the current configuration:

$$m_j \rightarrow \text{KS} \ : \ \mathsf{RequestKey} \mid m_j \ .$$

Upon reception, node KS performs authentication of m_j and, if successful, it sends the current configuration via an InitKey packet.

Encoding In Table 4, we provide a specification in `aTCWS` of the entire LiSP protocol. We introduce some slight simplifications with respect to the original protocol. We assume that (i) the temporal keys $k_0, \ldots, k_n$ have already been computed by KS, (ii) both the buffer size s and the refresh interval Δ_{refresh} are known by each node. Thus, the InitKey packet can be simplified as follows:

$$\text{KS} \rightarrow m_j \ : \ \text{enc}(k_{\text{KS}:m_j}, \ k_{s+i}) \ \mid \ \text{hash}(k_{s+i}) \ .$$

Moreover, we assume that every σ-action models the passage of $\Delta_{\text{refresh}}/2$ time units. Therefore, every two σ-actions the key server broadcasts the new temporal key encrypted with the key tied to that specific interval. Finally, we do not model data encryption.

When giving our encoding in `aTCWS` we will require some new deduction rules to model an hash function and encryption/decryption of messages:

$$\text{(hash)} \ \frac{w}{\text{hash}(w)} \qquad \text{(enc)} \ \frac{w_1 \quad w_2}{\text{enc}(w_1, w_2)} \qquad \text{(dec)} \ \frac{w_1 \quad w_2}{\text{dec}(w_1, w_2)} \ .$$

The protocol executed by the key server is expressed by the following two threads: a key distributor D_i and a listener L_i waiting for reconfiguration requests from the sensor nodes, with i being the current time interval. Every Δ_{refresh} time units (that is, every two σ-actions) D_i broadcasts the new temporal key k_{s+i} encrypted with the key k_i of the current time interval i. The process L_i replies to reconfiguration requests by sending an initialisation packet.

At the beginning of the protocol, a sensor node runs the process Z, which broadcasts a request packet to KS, waits for a reconfiguration packet q, and then checks authenticity by verifying the hash code. If the verification is successful then the node starts the broadcasting new keys phase. This phase is formalised by the process $R(k_{\text{C}}, k_{\text{L}}, l)$, where k_{C} is the current temporal key, k_{L} is the last authenticated temporal key, and the integer l counts the number of keys that are actually stored in the buffer.

To simplify the exposition, we formalise the key server as a pair of nodes: a key disposer KD, which executes D_i, and a listener KL, which executes L_i. Thus,

Table 4 The LiSP protocol

Key Server:			
D_0	$\stackrel{\text{def}}{=}$	$\sigma.D_1$	synchronise and move to D_1
D_i	$\stackrel{\text{def}}{=}$	$[k_i\ k_{s+i} \vdash_{\text{enc}} t_i]$	for $i \geq 1$, encrypt k_{s+i} with k_i
		$[\mathsf{UpdateKey}\ t_i \vdash_{\text{pair}} u_i]$	build the UpdateKey packet u_i
		$!\langle u_i\rangle.\sigma.\sigma.D_{i+1}$	broadcast r_i, and move to D_{i+1}
L_i	$\stackrel{\text{def}}{=}$	$\lfloor ?(r).I_{i+1}\rfloor\sigma.L_{i+1}$	wait for request packets
I_i	$\stackrel{\text{def}}{=}$	$[r \vdash_{\text{fst}} r_1]I_i^1;\sigma.\sigma.L_i$	extract first component
I_i^1	$\stackrel{\text{def}}{=}$	$[r_1 = \mathsf{RequestKey}]I_i^2;\sigma.\sigma.L_i$	check if r_1 is a RequestKey
I_i^2	$\stackrel{\text{def}}{=}$	$[r \vdash_{\text{snd}} m]$	extract node name
		$[k_{\text{KS:}m}\ k_{s+i} \vdash_{\text{enc}} w_i]$	encrypt k_{s+i} with $k_{\text{KS:}m}$
		$[k_{s+i} \vdash_{\text{hash}} h_i]$	calculate hash code for k_{s+i}
		$[w_i\ h_i \vdash_{\text{pair}} r_i]$	build a pair r_i,
		$[\mathsf{InitKey}\ r_i \vdash_{\text{pair}} q_i]$	build a InitKey packet q_i,
		$\sigma.!\langle q_i\rangle.\sigma.L_i$	broadcast q_i, move to L_i
Receiver at node m:			
Z	$\stackrel{\text{def}}{=}$	$[\mathsf{RequestKey}\ m \vdash_{\text{pair}} r]$	send a RequestKey packet
		$!\langle r\rangle.\sigma.\lfloor ?(q).T\rfloor Z$	wait for a reconfig. packet
T	$\stackrel{\text{def}}{=}$	$[q \vdash_{\text{fst}} q']T^1;\sigma.Z$	extract fst component of q
T^1	$\stackrel{\text{def}}{=}$	$[q' = \mathsf{InitKey}]T^2;\sigma.Z$	check if q is a InitKey packet
T^2	$\stackrel{\text{def}}{=}$	$[q \vdash_{\text{snd}} q'']$	extract snd component of q
		$[q'' \vdash_{\text{fst}} w]T^3;\sigma.Z$	extract fst component of q''
T^3	$\stackrel{\text{def}}{=}$	$[q'' \vdash_{\text{snd}} h]$	extract snd component of q''
		$[k_{\text{KS:}m}\ w \vdash_{\text{dec}} k]T^3;\sigma.Z$	extract the key
T^4	$\stackrel{\text{def}}{=}$	$[k \vdash_{\text{hash}} h'][h = h']T^5;\sigma.Z$	verify hash codes
T^5	$\stackrel{\text{def}}{=}$	$\sigma.\sigma.R\langle F^{s-1}(k), k, s{-}1\rangle$	synchronise and move to R
$R(k_{\text{C}}, k_{\text{L}}, l)$	$\stackrel{\text{def}}{=}$	$\lfloor ?(u).E\rfloor F$	wait for incoming packets
E	$\stackrel{\text{def}}{=}$	$[u \vdash_{\text{fst}} u']E^1;\sigma.F$	extract fst component of u
E^1	$\stackrel{\text{def}}{=}$	$[u' = \mathsf{UpdateKey}]E^2;\sigma.F$	check UpdateKey packet
E^2	$\stackrel{\text{def}}{=}$	$[u \vdash_{\text{snd}} u'']$	extract snd component of u
		$[k_{\text{C}}\ u'' \vdash_{\text{dec}} k]E^3;\sigma.F$	decrypt u'' by using k_{C}
E^3	$\stackrel{\text{def}}{=}$	$[F^{s-l}(k) = k_{\text{L}}]E^4;\sigma.F$	authenticate k
E^4	$\stackrel{\text{def}}{=}$	$\sigma.\sigma.R\langle F^{s-1}(k), k, s{-}1\rangle$	synchronise and move to R
F	$\stackrel{\text{def}}{=}$	$[l = 0]Z;\sigma.R\langle F^{l-1}(k_{\text{L}}), k_{\text{L}}, l{-}1\rangle$	check if buffer key is empty

the LiSP protocol, in its initial configuration, can be represented as:

$$\text{LiSP} \stackrel{\text{def}}{=} \prod_{j\in J} m_j[\sigma.Z]^{\nu_{m_j}} \ \mid\ \text{KS}[\sigma.D_0]^{\nu_{\text{KS}}} \ \mid\ \text{KL}[\sigma.L_0]^{\nu_{\text{KL}}}$$

where for each node m_j, with $j \in J$, $m_j \in \nu_{\text{KD}} \cap \nu_{\text{KL}}$ and $\{\text{KD}, \text{KL}\} \subseteq \nu_{m_j}$.

Security Analysis In LiSP, a node should authenticate only keys sent by the key server in the previous Δ_{refresh} time units. Otherwise, a node needing a reconfiguration would authenticate an obsolete temporal key and it would not be synchronised with the rest of the network. Here, we show that key authentication may take longer than Δ_{refresh} time units, as a consequence of an attack.

For our analysis, without loss of generality, it suffices to focus on a part of the protocol composed by the KL node of the key server and a single sensor node m. Moreover, in order to make observable a successful reconfiguration, we replace the process T^4 of Table 4 with the process

$$T^{4\prime} \stackrel{\text{def}}{=} \sigma.\sigma.[\mathsf{auth}\ k \vdash_{\text{pair}} a]!\langle a\rangle.R\langle F^{s-1}(k), k, s{-}1\rangle \ .$$

Thus, the part of the protocol under examination can be defined as follows:

$$\text{LiSP}' \stackrel{\text{def}}{=} m[\sigma.Z']^{\nu_m} \mid \text{KL}[\sigma.L_0]^{\nu_{\text{KL}}} \ .$$

Our freshness requirement on authenticated keys can be expressed by the following abstraction of the protocol:

$$\rho(\text{LiSP}') \stackrel{\text{def}}{=} m[\sigma.\hat{Z}_0]^{obs} \mid \text{KL}[\sigma.\hat{L}_0]^{obs}$$

where

- $\hat{Z}_i \stackrel{\text{def}}{=} !\langle r\rangle.\sigma.\lfloor\tau.\sigma.\sigma.!\langle\mathsf{auth}_{i+1}\rangle.R(k_{i+1}, k_{s+i}, s-1)\rfloor\hat{Z}_{i+1}$,
  with $r = \text{pair}(\mathsf{RequestKey},\ m)$ and $\mathsf{auth}_i = \text{pair}(\mathsf{auth}, k_{s+i})$ as in Table 4;
- $\hat{L}_i \stackrel{\text{def}}{=} \lfloor\tau.\sigma.!\langle q_{i+1}\rangle.\sigma.\hat{L}_{i+1}\rfloor\sigma.\hat{L}_{i+1}$, and q_i defined as in Table 4:
  $q_i = \text{pair}(\mathsf{InitKey}\ r_i)$ with $r_i = \text{pair}(\text{enc}(k_{\text{KS}:m}, k_{s+i}), \text{hash}(k_{s+i}))$.

It is easy to see that $\rho(\text{LiSP}')$ is a correct abstraction of key authentication within the protocol, as the action auth_i occurs exactly Δ_{refresh} time units (that is, two σ-actions) after the disclosure of key k_{s+i} through packet q_i.

Proposition 3. $\rho(\text{LiSP}') \stackrel{\Lambda}{\Longrightarrow} \xrightarrow{!q_i \triangleright obs} \stackrel{\Omega}{\Longrightarrow} \xrightarrow{!\mathsf{auth}_i \triangleright obs}$ *implies* $\#^{\sigma}(\Omega) = 2$.

In order to show that LiSP$'$ satisfies our security analysis, we should prove that

$$\text{LiSP}' \ \in \ tGNDC^{\rho(\text{LiSP}')}_{\phi_0,\mathcal{O}}$$

for $\mathcal{O} = \mathsf{nds}\left(\text{LiSP}'\right)$ and initial knowledge $\phi_0 = \emptyset$. However, this is not the case.

Theorem 4 (Replay attack to LiSP).

$$\text{LiSP}' \ \notin \ tGNDC^{\rho(\text{LiSP}')}_{\emptyset,\{\text{KL},m\}} \ .$$

Proof Let us define the set of attacking nodes $\mathcal{A} = \{a, b\}$ for LiSP$'$. Let us fix the initial knowledge of the attacker $\phi_0 = \emptyset$. We set $\nu_a = \{m, b\}$ and $\nu_b = \{\text{KL}, a\}$, and we assume that $\mathcal{O} = \{\text{KL}, m\}$. We give an intuition of the replay attack in Table 5. Basically, an attacker may prevent the node m to receive the $\mathsf{InitKey}$ packet within Δ_{refresh} time units. As a consequence, m may

Table 5 Replay attack to LiSP

$m \longrightarrow \text{KL} : r$	m sends a RequestKey and KL correctly receives the packet
$\xrightarrow{\sigma}$	the system moves to the next time slot
$\text{KL} \longrightarrow m : q_1$	KL replies with an InitKey which is lost by m and grasped by b
$\xrightarrow{\sigma}$	the system moves to the next time slot
$b \rightarrow a : q_1$	b sends q_1 to a
$m \rightarrow \text{KL} : r$	m sends a new RequestKey which gets lost
$\xrightarrow{\sigma}$	the system moves to the next time slot
$a \rightarrow m : q_1$	*a replays* q_1 to m
$\xrightarrow{\sigma}\xrightarrow{\sigma}$	after Δ_{refresh} time units
$m \rightarrow * : \mathsf{auth}_1$	m authenticates q_1 and signals the end of the protocol

complete the protocol only after $2\Delta_{\text{refresh}}$ time units (that is, four σ-actions), so authenticating an old key. Formally, we define the attacker $A \in \mathbb{A}^{\phi_0}_{\mathcal{A}/\{\text{KL},m\}}$ as $A = a[\sigma.\sigma.\sigma.X]^{\nu_a} \mid b[\sigma.\sigma.X]^{\nu_b}$ where $X = \lfloor ?(x).\sigma.!\langle x\rangle.\mathsf{nil}\rfloor\mathsf{nil}$. We then consider the system $(\text{LiSP}')^{\mathcal{A}}_{\mathcal{O}} \mid A$ which admits the following execution trace:

$$\sigma \,.\, !r \triangleright obs \,.\, \sigma \,.\, !q_1 \triangleright obs \,.\, \sigma \,.\, \tau \,.\, !r \triangleright obs \,.\, \sigma \,.\, \tau \,.\, \sigma \,.\, \sigma \,.\, !\mathsf{auth}_1 \triangleright obs$$

containing four σ-actions between the packets q_1 and auth_1. By Proposition 3, this trace cannot be matched by $\rho(\text{LiSP}')$. So, $(\text{LiSP}')^{\mathcal{A}}_{\mathcal{O}} \mid A \not\lesssim \rho(\text{LiSP}')$. □

4.1 LiSP with nonces

Replay attacks as those described above appears also in other key management protocols, such as μTESLA [14] and LEAP+ [17]. These protocols have been amended by adding nonces to guarantee freshness. We propose to do the same in LiSP. For this purpose, we extend our inference system with a new deduction rule to model a *pseudo-random function*: The application $\text{prf}(m, w_i)$ returns a pseudo-random value w_{i+1} associated to a node m and the last generated value w_i. In our amended specification of LiSP, we add a nonce to the RequestKey packet. The nonce is then included in the corresponding InitKey packet to guarantee the freshness of the reply. These changes affect only those processes which model the key request at the node side and the reply at the server side. We modify these processes as shown in Table 6. The requesting nodes run the process Z_j, where j is the number associated to the current key request. At each request j, the receiver generates a nonce n_j which will be used to check the freshness of the received key. The process $\bar{L}_i$, running at the key server, now includes the received nonce in the InitKey packet. Notice that, as done before, the process T^7_j signals a successful reconfiguration. Again, for our analysis, it suffices to analyse the following fragment of the protocol:

$$\text{LiSP}'' \stackrel{\text{def}}{=} m[\sigma.Z_1]^{\nu_m} \mid \text{KL}[\sigma.\bar{L}_0]^{\nu_{\text{KL}}} \ .$$

Table 6 LiSP with nonces

Key Server:			
$\bar{L}_i$	$\stackrel{\text{def}}{=}$	$\lfloor ?(r).\bar{I}_{i+1} \rfloor \sigma.\bar{L}_{i+1}$	wait for request packets
$\bar{I}_i$	$\stackrel{\text{def}}{=}$	$[r \vdash_{\text{fst}} r_1]\bar{I}_i^1; \sigma.\sigma.\bar{L}_{i+1}$	extract first component
$\bar{I}_i^1$	$\stackrel{\text{def}}{=}$	$[r_1 = \mathsf{RequestKey}]\bar{I}_i^2; \sigma.\sigma.\bar{L}_{i+1}$	check if r_1 is a RequestKey
$\bar{I}_i^2$	$\stackrel{\text{def}}{=}$	$[r \vdash_{\text{snd}} t]$	extract second component
		$[t \vdash_{\text{fst}} m]\bar{I}_i^3; \sigma.\sigma.\bar{L}_{i+1}$	extract node name
$\bar{I}_i^3$	$\stackrel{\text{def}}{=}$	$[t \vdash_{\text{snd}} n]$	extract nonce
		$[k_{s+i}\ n \vdash_{\text{pair}} p]$	build a pair
		$[k_{\text{KS}:m}\ p \vdash_{\text{enc}} w_i]$	encrypt p with $k_{\text{KS}:m}$
		$[k_{s+i} \vdash_{\text{hash}} h_i]$	calculate hash code for k_{s+i}
		$[w_i\ h_i \vdash_{\text{pair}} r_i]$	build a pair r_i,
		$[\mathsf{InitKey}\ r_i \vdash_{\text{pair}} q_i]$	build a InitKey packet q_i,
		$\sigma.!\langle q_i \rangle.\sigma.\bar{L}_{i+1}$	broadcast q_i, move to $\bar{L}_{i+1}$
Receiver at node m:			
Z_j	$\stackrel{\text{def}}{=}$	$[m\ n_{j-1} \vdash_{\text{prf}} n_j]$	build a random nonce n_j
		$[m\ n_j \vdash_{\text{pair}} t]$	build a pair t with name m and nonce n_j
		$[\mathsf{RequestKey}\ t \vdash_{\text{pair}} r]$	send a RequestKey packet
		$!\langle r \rangle.\sigma.\lfloor ?(q).T_j \rfloor Z_{j+1}$	wait for a reconfig. packet
T_j	$\stackrel{\text{def}}{=}$	$[q \vdash_{\text{fst}} q']T_j^1; \sigma.Z_{j+1}$	extract fst component of q
T_j^1	$\stackrel{\text{def}}{=}$	$[q' = \mathsf{InitKey}]T_j^2; \sigma.Z_{j+1}$	check if q is a InitKey packet
T_j^2	$\stackrel{\text{def}}{=}$	$[q \vdash_{\text{snd}} q'']$	extract snd component of q
		$[q'' \vdash_{\text{fst}} w]T_j^3; \sigma.Z_{j+1}$	extract fst component of q''
T_j^3	$\stackrel{\text{def}}{=}$	$[q'' \vdash_{\text{snd}} h]$	extract snd component of q''
		$[k_{\text{KS}:m}\ w \vdash_{\text{dec}} p]T_j^4; \sigma.Z_{j+1}$	decript w
T_j^4	$\stackrel{\text{def}}{=}$	$[p \vdash_{\text{fst}} k]T_j^5; \sigma.Z_{j+1}$	extract the key
T_j^5	$\stackrel{\text{def}}{=}$	$[p \vdash_{\text{snd}} n][n = n_j]T_j^6; \sigma.Z_{j+1}$	verify nonces
T_j^6	$\stackrel{\text{def}}{=}$	$[k \vdash_{\text{hash}} h'][h = h']T_j^7; \sigma.Z_{j+1}$	verify hash codes
T_j^7	$\stackrel{\text{def}}{=}$	$\sigma.\sigma.[\mathsf{auth}\ k \vdash_{\text{pair}} a]!\langle a \rangle.\mathsf{nil}$	reaching of synchronisation

According to Definition 8, the system LiSP$''$ is time-dependent stable with respect to the following sequence of knowledge:

$$\begin{array}{lcll} \phi_0 & \stackrel{\text{def}}{=} & \emptyset & \\ \phi_1 & \stackrel{\text{def}}{=} & \{r_1\} & \\ \phi_i & \stackrel{\text{def}}{=} & \phi_{i-1} \cup \{q_j\} & \text{if } j > 0 \text{ and } i = 2j \\ \phi_i & \stackrel{\text{def}}{=} & \phi_{i-1} \cup \{\mathsf{auth}_j, r_{j+1}\} & \text{if } j > 0 \text{ and } i = 2j+1 \end{array} \tag{1}$$

where

$$\begin{aligned} \mathsf{auth}_j &= \text{pair}(\mathsf{auth}, k_{s+j}) \\ r_j &= \text{pair}(\mathsf{RequestKey}, \text{pair}(m, n_j)) \\ q_j &= \text{pair}(\,\mathsf{InitKey},\ \text{pair}(\,\text{enc}(k_{\text{KS}:m}, \text{pair}(k_{s+j}, n_j)),\ \text{hash}(k_{s+j})\,)\,)\ . \end{aligned}$$

Intuitively, ϕ_i consists of ϕ_{i-1} together with the set of messages an intruder can get by eavesdropping on a run of the protocol during the time slot i.

With the introduction of nonces, the abstraction expressing key authentication within Δ_{refresh} time units becomes the following:

$$\rho(\text{LiSP}'') \stackrel{\text{def}}{=} m[\sigma.\hat{Z}'_1]^{obs} \mid \text{KL}[\sigma.\hat{L}'_0]^{obs}$$

where

- $\hat{Z}'_i \stackrel{\text{def}}{=} [m\ n_{i-1} \vdash_{\text{prf}} n_i][m\ n_i \vdash_{\text{pair}} t][\mathsf{RequestKey}\ t \vdash_{\text{pair}} r]!\langle r\rangle.\sigma.\lfloor\tau.\sigma.\sigma.!\langle \mathsf{auth}_{i+1}\rangle.\mathsf{nil}\rfloor \hat{Z}'_{i+1}$
- $\hat{L}'_i \stackrel{\text{def}}{=} \lfloor \sum_{v\in\mathcal{D}(\phi_{2i+1})} \tau.\sigma.!\langle q^v_{i+1}\rangle.\sigma.\hat{L}'_{i+1}\rfloor \sigma.\hat{L}'_{i+1}$
  with $q^v_i = \text{pair}(\mathsf{InitKey}\ \ \text{pair}(\text{enc}(k_{\text{KS}:m}, \text{pair}(k_{s+i}, v)), \text{hash}(k_{s+i})))$.

In $\rho(\text{LiSP}'')$ keys are authenticated after Δ_{refresh} time units (two σ-actions).

Proposition 4. $\rho(\text{LiSP}'') \stackrel{\Lambda}{\Longrightarrow} \xrightarrow{!q^v_i \rhd obs} \stackrel{\Omega}{\Longrightarrow} \xrightarrow{!\mathsf{auth}_i \rhd obs} M$ *implies* $\#^\sigma(\Omega)$=2.

Now, everything is in place to prove the safety of the LiSP protocol with nonces.

Lemma 1. *Given two attacking nodes a and b, for m and* KL *respectively, and fixed the sequence of knowledge $\{\phi_i\}_{i\geq 0}$ as in* (1)*, then*

1. $\text{KL}[\sigma.\bar{L}_0]^{\{b,obs\}} \mid \text{TOP}^{\phi_0}_{b/\text{KL}} \lesssim \text{KL}[\sigma.\hat{L}'_0]^{obs}$
2. $m[\sigma.Z_1]^{\{a,obs\}} \mid \text{TOP}^{\phi_0}_{a/m} \lesssim m[\sigma.\hat{Z}'_1]^{obs}$.

Theorem 5 (Safety of LiSP with nonces). $\text{LiSP}'' \in tGNDC^{\rho(\text{LiSP}'')}_{\emptyset,\mathsf{nds}(\text{LiSP}'')}$.

Proof By an application of Lemma 1 and Theorem 3. □

References

1. Ballardin, F., Merro, M.: A calculus for the analysis of wireless network security protocols. In: FAST. LCNS, vol. 6561, pp. 206–222. Springer (2010)
2. Ghassemi, F., Fokkink, W., Movaghar, A.: Equational Reasoning on Mobile Ad Hoc Networks. Fundamentae Informaticae 105(4):375–415 (2010)
3. Godskesen, J.C.: A Calculus for Mobile Ad Hoc Networks. In: COORDINATION. LNCS, vol. 4467, pp. 132–150. Springer (2007)
4. Godskesen, J.C., Nanz, S.: Mobility Models and Behavioural Equivalence for Wireless Networks. In:COORDINATION. LNCS, vol.5521, pp.106–122. Springer (2009)
5. Gorrieri, R., Martinelli, F.: A simple framework for real-time cryptographic protocol analysis with compositional proof rules. Sc. of Com. Prog. 50, 23–49 (2004)
6. Gorrieri, R., Martinelli, F., Petrocchi, M.: Formal models and analysis of secure multicast in wired and wireless networks. J. Aut. Reasoning 41(3-4), 325–364 (2008)
7. Hennessy, M., Regan, T.: A Process Algebra for Timed Systems. Information and Computation 117(2), 221–239 (1995)
8. Lanese, I., Sangiorgi, D.: An Operational Semantics for a Calculus for Wireless Systems. Theoretical Computer Science 411, 1928–1948 (2010)
9. Merro, M.: An Observational Theory for Mobile Ad Hoc Networks (full paper). Information and Computation 207(2), 194–208 (2009)

10. Merro, M., Ballardin, F., Sibilio, E.: A Timed Calculus for Wireless Systems. Theoretical Computer Science 412(47), 6585–6611 (2011)
11. Misra, S., Woungag, I.: Guide to Wireless Ad Hoc Networks. Computer Communications and Networks, Springer (2009)
12. Nanz, S., Hankin, C.: A Framework for Security Analysis of Mobile Wireless Networks. Theoretical Computer Science 367(1-2), 203–227 (2006)
13. Park, T., Shin, K.G.: LiSP: A lightweight security protocol for wireless sensor networks. ACM Trans. Embedded Comput. Syst. 3(3), 634–660 (2004)
14. Perrig, A., Szewczyk, R., Tygar, J.D., Wen, V., Culler, D.: SPINS: Security Protocols for Sensor Networks. Wireless Networks 8(5), 521–534 (2002)
15. Singh, A., Ramakrishnan, C.R., Smolka, S.A.: A Process Calculus for Mobile Ad Hoc Networks. In: COORDINATION. LNCS, vol. 5052, pp. 296–314. (2008)
16. Sundararaman, B., Buy, U., Kshemkalyani, A.D.: Clock synchronization for wireless sensor networks: a survey. Ad Hoc Networks 3(3), 281–323 (2005)
17. Zhu, S., Setia, S., Jajodia, S.: Leap+: Efficient security mechanisms for large-scale distributed sensor networks. ACM Trans. on Sensor Networks 2(4), 500–528 (2006)

A Appendix

A.1 Proofs of Section 2

Proof of Proposition 1 We single out each item of the proposition.
Item 1. The forward direction is an instance of rule (RcvEnb), the converse is proved by a straightforward rule induction.

Item 2. The forward direction follows by noticing that only rules (RcvEnb) and (RcvPar) are suitable for deriving the action $m?w$ from $M_1 \mid M_2$; in the case of rule (RcvEnb) we just apply rule (RcvEnb) both on M_1 and on M_2, in the case of rule (RcvPar) the premises require both M_1 and M_2 to perform an action $m?w$ and to move to N_1 and N_2 with $N = N_1 \mid N_2$. The converse is an instance of rule (σ-Par).

Item 3. The result is a consequence of the combination of rules (Snd) and (Bcast) and it is proved by a straightforward rule induction.

Item 4. Again, the proof is done by a straightforward rule induction.

Item 5. The forward direction follows by noticing that the only rule for deriving the action σ from $M_1 \mid M_2$ is (σ-Par) which, in the premises, requires both M_1 and M_2 to perform an action σ. The converse is an instance of rule (σ-Par). □

Proof of Proposition 2 The property is a consequence of the fact that the topology of networks is static. □

A.2 Time properties

Our calculus `aTCWS` enjoys some desirable time properties. Here, we outline the most significant ones. Proposition 5 formalises the deterministic nature of time passing: a network can reach at most one new state by executing a σ-action.

Proposition 5 (Time Determinism). *If M is a well-formed network with $M \xrightarrow{\sigma} M'$ and $M \xrightarrow{\sigma} M''$, then M' and M'' are syntactically the same.*

Proof By induction on the length of the proof of $M \xrightarrow{\sigma} M'$. □

Patience guarantees that a process will wait indefinitely until it can communicate [7]. In our setting, this means that

Proposition 6 (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 $m_i[P_i]^{\nu_i} \not\equiv m_i[!\langle w\rangle.Q_i]^{\nu_i}$, then there is a network N such that $M \xrightarrow{\sigma} N$.*

Proof By induction on the structure of M. □

The maximal progress property says that processes communicate as soon as a possibility of communication arises [7]. In other words, the passage of time cannot block transmissions.

Proposition 7 (Maximal Progress). *Let M be a well-formed network. If $M \equiv m[!\langle w\rangle.P]^{\nu} \mid N$ then $M \xrightarrow{\sigma} M'$ for no network M'.*

Proof By inspection on the rules that can be used to derive $M \xrightarrow{\sigma} M'$, because sender nodes cannot perform σ-actions. □

Basically, time cannot pass unless the specification itself explicitly asks for it. This approach provides a lot of power to the specification, which can precisely handle the flowing of time. Such an extra expressive power leads, as a drawback, to the possibility of abuses. For instance, infinite loops of broadcast actions or internal computations prevent time passing. The *well-timedness* (or *finite variability*) property puts a limitation on the number of instantaneous actions that can fire between two contiguous σ-actions. Intuitively, well-timedness says that time passing never stops: Only a finite number of instantaneous actions can fire between two subsequent σ-actions.

Definition 10 (Well-Timedness). *A network M satisfies well-timedness if there exists an upper bound $k \in \mathbb{N}$ such that whenever $M \xrightarrow{\lambda_1} \cdots \xrightarrow{\lambda_h}$ where λ_j is not directly derived by an application of (RcvEnb) and $\lambda_j \neq \sigma$ (for $1 \leq j \leq h$) then $k \leq h$.*

The above definition takes into account only transitions denoting an active involvement of the network, that is why we have left out those transitions which can be derived by applying rule (RcvEnb). However, as `aTCWS` is basically a specification language, there is no harm in allowing specifications which do not respect well-timedness. Of course, when using our language to give a protocol implementation, then one must verify that the implementation satisfies well-timedness: No real-world service (even a attackers) can stop the passage of time.

The following proposition provides a criterion to check well-timedness. We recall that recursion is *time-guarded* if P contains only time-guarded occurrences of the process identifiers. We write Prc_{wt} for the set of processes in which summations are finite-indexed and recursive definitions are time-guarded.

Proposition 8. *Let $M = \prod_{i\in I} m_i[P_i]^{\nu_i}$ be a network. If for all $i \in I$ we have $P_i \in Prc_{\mathrm{wt}}$ then M satisfies well-timedness.*

Proof First notice that without an application of (RcvEnb) the network M can perform only a finite number of transitions. Then proceed by induction on the structure of M. □

Remark 1. By Proposition 8, the requirement $Q_i \in Prc_{\mathrm{wt}}$ in the definition of $\mathbb{A}^{\phi_0}_{\mathcal{A}/\mathcal{P}}$ guarantees that our attackers respects well-timedness and hence cannot prevent the passage of time.

Remark 2. Notice that the top attacker does not satisfy well-timedness (see Definition 10), as the process identifiers involved in the recursive definition are not time-guarded. However, this is not a problem as we are looking for a sufficient condition which ensures tGNDC with respect to well-timed attackers.

A.3 Proofs of Section 2

Proposition 9. *If* $M \lesssim N$ *then* $\mathsf{nds}(N) \subseteq \mathsf{nds}(M)$.

Proof By contradiction. Assume there exists a node m such that $m \in \mathsf{nds}(N)$ and $m \notin \mathsf{nds}(M)$. Then, by rule (RcvEnb), $M \xrightarrow{m?w} M$. Since $M \lesssim N$ there must be N' such that $N \stackrel{m?w}{\Longrightarrow} N'$ with $M' \lesssim N'$. However, since $m \in \mathsf{nds}(N)$, by inspection on the transition rules, there is no way to deduce a weak transition of the form $N \stackrel{m?w}{\Longrightarrow} N'$. □

Proof of Theorem 1 We prove that the relation

$$\mathcal{R} = \left\{ \left(M \mid O,\ N \mid O\right) \text{ s.t. } M \lesssim N \text{ and } M \mid O,\ N \mid O \text{ are well-formed} \right\}$$

is a simulation. We proceed by case analysis on why $M \mid O \xrightarrow{\alpha} Z$. The interesting cases are when the transition is due to an interaction between M and O. The remaining cases are more elementary.

Let $M \mid O \xrightarrow{!w \rhd \nu} M' \mid O'$ ($\nu \neq \emptyset$) by an application of rule (Obs), because $M \mid O \xrightarrow{m!w \rhd \eta} M' \mid O'$, by an application of rule (Bcast) with $\nu \subseteq \eta$. There are two possible ways to derive this transition, depending on where the sender node is located in the network.

1. $M \xrightarrow{m!w \rhd \mu} M'$ and $O \xrightarrow{m?w} O'$, with $m \in \mathsf{nds}(M)$ and $\eta = \mu \backslash \mathsf{nds}(O)$. By an application of rule (Obs) we obtain that $M \xrightarrow{!w \rhd \mu} M'$. Since $M \lesssim N$, it follows that there is N' such that $N \stackrel{!w \rhd \mu}{\Longrightarrow} N'$ with $M' \lesssim N'$. This implies that there exists $h \in \mathsf{nds}(N)$ such that $N \stackrel{h!w \rhd \mu'}{\Longrightarrow} N'$ with $\mu \subseteq \mu'$. Moreover:
   (a) $h \notin \mathsf{nds}(O)$, as $N \mid O$ is well-formed and it cannot contain two nodes with the same name;
   (b) $\mu' \subseteq \mathsf{ngh}(h, N)$, by Proposition 1(3);
   (c) If $k \in \mu' \cap \mathsf{nds}(O)$ then $h \in \mathsf{ngh}(k, O)$, as the neighbouring relation is symmetric.

   Now, in case $O \xrightarrow{m?w} O'$ exclusively by rule (RcvEnb) then also $O \xrightarrow{h?w} O'$ by rule (RcvEnb) and item (a). In case the derivation of $O \xrightarrow{m?w} O'$ involves some applications of the rule (Rcv) then the concerned nodes have the form

$k[\lfloor ?(x).P \rfloor Q]^{\pi}$ with $k \in \mu$, hence $h \in \mathsf{ngh}(k, O)$ by item (c), and so we can derive $O \xrightarrow{h?w} O'$ by applying the rules (RcvEnb) and (RcvPar).

Thus we have $O \xrightarrow{h?w} O'$ in any case. Then by an application of rule (Bcast) and several applications of rule (TauPar) we have $N \mid O \xRightarrow{h!w \triangleright \eta'} N' \mid O'$ with $\eta' = \mu' \setminus \mathsf{nds}(O)$. Now, since $\mu \subseteq \mu'$ we have $\mu \setminus \mathsf{nds}(O) \subseteq \mu' \setminus \mathsf{nds}(O)$ hence $\nu \subseteq \eta \subseteq \eta'$. As $\nu \neq \emptyset$, by an application of rule (Obs) and several applications of rule (TauPar) it follows that $N \mid O \xRightarrow{!w \triangleright \nu} N' \mid O'$. Since $M' \lesssim N'$, we obtain $(M' \mid O',\ N' \mid O') \in \mathcal{R}$.

2. $M \xrightarrow{m?w} M'$ and $O \xrightarrow{m!w \triangleright \mu} O'$, with $m \in \mathsf{nds}(O)$ and $\eta = \mu \setminus \mathsf{nds}(M)$. Since $M \lesssim N$, it follows that there is N' such that $N \xRightarrow{m?w} N'$ with $M' \lesssim N'$. By an application of rule (Bcast) and several applications of rule (TauPar) we have $N \mid O \xRightarrow{m!w \triangleright \eta'} N' \mid O'$, with $\eta' = \mu \setminus \mathsf{nds}(N)$. Since $M \lesssim N$, by Proposition 9 we have $\eta \subseteq \eta'$. Thus $\nu \subseteq \eta'$ and by an application of rule (Obs) and several applications of rule (TauPar) it follows that $N \mid O \xRightarrow{!w \triangleright \nu} N' \mid O'$. Since $M' \lesssim N'$, we obtain $(M' \mid O',\ N' \mid O') \in \mathcal{R}$.

Let $M \mid O \xrightarrow{\tau} M' \mid O'$ by an application of rule (Shh) because $M \mid O \xrightarrow{m!w \triangleright \emptyset} M' \mid O'$. This case is similar to the previous one.

Let $M \mid O \xrightarrow{m?w} M' \mid O'$ by an application of rule (RcvPar) because $M \xrightarrow{m?w} M'$ and $O \xrightarrow{m?w} O'$. Since $M \lesssim N$, it follows that there is N' such that $N \xRightarrow{m?w} N'$ with $M' \lesssim N'$. By an application of rule (RcvPar) and several applications of rule (TauPar) we have $N \mid O \xRightarrow{m?w} N' \mid O'$. Since $M' \lesssim N'$, we obtain $(M' \mid O',\ N' \mid O') \in \mathcal{R}$.

Let $M \mid O \xrightarrow{\sigma} M' \mid O'$ by an application of rule (σ-Par) because $M \xrightarrow{\sigma} M'$ and $O \xrightarrow{\sigma} O'$. This case is similar to the previous one. □

A.4 Proofs of Section 3

We define $\mathrm{msg}(P)$ as $\mathrm{msg}_{\emptyset}(P)$, where $\mathrm{msg}_S : Prc \to 2^{Msg}$, for $S \subseteq PrcIds$, is defined in Table 7 along the lines of [5]. Intuitively, msg_S is a function that visits recursively the sub-terms of P and the body of the recursive definitions referred by P. The index S is used to guarantee that the unwinding of every recursive definition is performed exactly once. A generalisation of msg( ) to networks is straightforward.

In the sequel, we will use the symbol $\uplus$ to denote *disjoint union*. Moreover, to ease the notation, whenever $\mathcal{O} = \mathsf{nds}(M)$ we will write $M^{\mathcal{A}}$ instead of $M^{\mathcal{A}}_{\mathsf{nds}(M)}$.

Lemma 2. *Let $M_1 \mid M_2$ be time-dependent stable with respect to a sequence of knowledge $\{\phi_j\}_{j \geq 0}$. Let $\mathcal{A}_1$ and $\mathcal{A}_2$ be disjoint sets of attacking nodes for M_1*

Table 7 Function msg_S

$$
\begin{aligned}
\mathrm{msg}_S(\mathsf{nil}) &\stackrel{\text{def}}{=} \emptyset \\
\mathrm{msg}_S(!\langle u\rangle.P) &\stackrel{\text{def}}{=} get(u) \cup \mathrm{msg}_S(P) \\
\mathrm{msg}_S(\lfloor ?(x).P \rfloor Q) &\stackrel{\text{def}}{=} \mathrm{msg}_S(P) \cup \mathrm{msg}_S(Q) \\
\mathrm{msg}_S(\lfloor \textstyle\sum_{i\in I} \tau.P_i \rfloor Q) &\stackrel{\text{def}}{=} \bigcup_{i\in I} msg_S(P_i) \cup \mathrm{msg}_S(Q) \\
\mathrm{msg}_S(\sigma.P) &\stackrel{\text{def}}{=} \mathrm{msg}_S(P) \\
\mathrm{msg}_S([u_1 = u_2]P; Q) &\stackrel{\text{def}}{=} get(u_1) \cup get(u_2) \cup \mathrm{msg}_S(P) \cup \mathrm{msg}_S(Q) \\
\mathrm{msg}_S([u_1 \ldots u_n \vdash_{\mathrm{r}} x]P; Q) &\stackrel{\text{def}}{=} \textstyle\bigcup_{i=1}^{n} get(u_i) \cup \mathrm{msg}_S(P) \cup \mathrm{msg}_S(Q) \\
\mathrm{msg}_S(H\langle u_1 \ldots u_r\rangle) &\stackrel{\text{def}}{=} \begin{cases} \bigcup_{i=1}^{r} get(u_i) \cup \mathrm{msg}_{S\cup\{H\}}(P) & \text{if } H(\tilde{x}) \stackrel{\text{def}}{=} P \text{ and } H \notin S \\ \bigcup_{i=1}^{r} get(u_i) & \text{otherwise} \end{cases}
\end{aligned}
$$

where $get : Val \to 2^{Msg}$ is defined as follows:

$$
\begin{aligned}
get(a) &\stackrel{\text{def}}{=} \{a\} \qquad \text{(basic message)} \\
get(x) &\stackrel{\text{def}}{=} \emptyset \qquad \text{(variable)} \\
get(\,\mathrm{F}^i(u_1, \ldots, u_{k_i})\,) &\stackrel{\text{def}}{=} \begin{cases} \{\mathrm{F}^i(u_1, \ldots, u_{k_i})\} \cup \{u_1 \ldots u_{k_i}\} & \text{if } \mathrm{F}^i(u_1 \ldots u_{k_i}) \in Msg \\ get(u_1) \cup \ldots \cup get(u_{k_i}) & \text{otherwise.} \end{cases}
\end{aligned}
$$

and M_2, respectively. Let $\mathcal{O}_1 \subseteq \mathsf{nds}(M_1)$ and $\mathcal{O}_2 \subseteq \mathsf{nds}(M_2)$. Then

$$
(M_1 \mid M_2)^{\mathcal{A}_1 \uplus \mathcal{A}_2}_{\mathcal{O}_1 \uplus \mathcal{O}_2} \mid \mathrm{TOP}^{\phi_0}_{\mathcal{A}_1 \uplus \mathcal{A}_2/\mathsf{nds}(M)} \lesssim (M_1)^{\mathcal{A}_1}_{\mathcal{O}_1} \mid (M_2)^{\mathcal{A}_2}_{\mathcal{O}_2} \mid \mathrm{TOP}^{\phi_0}_{\mathcal{A}_1/\mathsf{nds}(M_1)} \mid \mathrm{TOP}^{\phi_0}_{\mathcal{A}_2/\mathsf{nds}(M_2)} \ .
$$

Proof We first note that a straightforward consequence of Definition 9 is:

$$
\mathrm{TOP}^{\phi_0}_{(\mathcal{A}_1 \uplus \mathcal{A}_2)/\mathsf{nds}(M)} = \mathrm{TOP}^{\phi_0}_{\mathcal{A}_1/\mathsf{nds}(M_1)} \mid \mathrm{TOP}^{\phi_0}_{\mathcal{A}_2/\mathsf{nds}(M_2)} \ .
$$

Then, in order to prove the result, we just need to show that

$$
\big(M_1 \mid M_2\big)^{\mathcal{A}_1 \uplus \mathcal{A}_2}_{\mathcal{O}_1 \uplus \mathcal{O}_2} \mid \mathrm{TOP}^{\phi_0}_{\mathcal{A}_1 \uplus \mathcal{A}_2/\mathsf{nds}(M)} \lesssim \big(M_1\big)^{\mathcal{A}_1}_{\mathcal{O}_1} \mid \big(M_2\big)^{\mathcal{A}_2}_{\mathcal{O}_2} \mid \mathrm{TOP}^{\phi_0}_{\mathcal{A}_1 \uplus \mathcal{A}_2/\mathsf{nds}(M)} \ .
$$

To improve readability, we consider the most general case, that is $\mathcal{O}_1 = \mathsf{nds}(M_1)$ and $\mathcal{O}_2 = \mathsf{nds}(M_2)$. Moreover, we assume $M_1 = m_1[P_1]^{\nu_1}$, $M_2 = m_2[P_2]^{\nu_2}$ and therefore $\mathcal{A}_1 = \{a_1\}$, $\mathcal{A}_2 = \{a_2\}$. The generalisation is straightforward. Then we have:

- $\big(M_1 \mid M_2\big)^{\mathcal{A}_1 \uplus \mathcal{A}_2} = m_1[P_1]^{\nu_1'} \mid m_2[P_2]^{\nu_2'}$
  with $\{a_1, obs\} \subseteq \nu_1' \subseteq \{a_1, m_2, obs\}$ and $\{a_2, obs\} \subseteq \nu_2' \subseteq \{a_2, m_1, obs\}$;
- $M_1^{\mathcal{A}_1} = m_1[P_1]^{\nu_1''}$ with $\nu_1'' = \{a_1, obs\}$;
- $M_2^{\mathcal{A}_2} = m_2[P_2]^{\nu_2''}$ with $\nu_2'' = \{a_2, obs\}$.

We define $\mathcal{P} = \{m_1, m_2\}$ and $\mathcal{A} = \{a_1, a_2\}$. We need to prove

$$m_1[P_1]^{\nu_1'} \mid m_2[P_2]^{\nu_2'} \mid \textsc{Top}^{\phi_0}_{\mathcal{A}/\mathcal{P}} \lesssim m_1[P_1]^{\nu_1''} \mid m_2[P_2]^{\nu_2''} \mid \textsc{Top}^{\phi_0}_{\mathcal{A}/\mathcal{P}} \ .$$

We prove that the following binary relation is a simulation:

$$\mathcal{R} \stackrel{\text{def}}{=} \bigcup_{j\geq 0} \Big\{ \big(m_1[Q_1]^{\nu_1'} \mid m_2[Q_2]^{\nu_2'} \mid N \,,\, m_1[Q_1]^{\nu_1''} \mid m_2[Q_2]^{\nu_2''} \mid \textsc{Top}^{\phi_j}_{\mathcal{A}/\mathcal{P}} \big) \text{ s.t. } m_1[P_1]^{\nu_1'} \mid m_2[P_2]^{\nu_2'} \mid \textsc{Top}^{\phi_0}_{\mathcal{A}/\mathcal{P}} \stackrel{\Lambda}{\Longrightarrow} m_1[Q_1]^{\nu_1'} \mid m_2[Q_2]^{\nu_2'} \mid N \text{ for some } \Lambda \text{ with } \#^{\sigma}(\Lambda) = j \Big\} \ .$$

We consider $(m_1[Q_1]^{\nu_1'} \mid m_2[Q_2]^{\nu_2'} \mid N \,,\, m_1[Q_1]^{\nu_1''} \mid m_2[Q_2]^{\nu_2''} \mid \textsc{Top}^{\phi_j}_{\mathcal{A}/\mathcal{P}}) \in \mathcal{R}$ and we proceed by case analysis on why $m_1[Q_1]^{\nu_1'} \mid m_2[Q_2]^{\nu_2'} \mid N \xrightarrow{\alpha} m_1[\hat{Q}_1]^{\nu_1'} \mid m_2[\hat{Q}_2]^{\nu_2'} \mid \hat{N}$.

- $\alpha = m?w$. This case is straightforward. In fact, the environment of the system contains exclusively the node *obs* which cannot transmit; thus the rule (Rcv) cannot be applied. We can consider just the rules (RcvEnb) and (RcvPar), which do not modify the network.
- $\alpha = \sigma$. Then $m_i[Q_i]^{\nu_i'} \xrightarrow{\sigma} m_i[\hat{Q}_i]^{\nu_i'}$ (for $i = 1, 2$) and $N \xrightarrow{\sigma} \hat{N}$. Now also $\textsc{Top}^{\phi_j}_{\mathcal{A}/\mathcal{P}} \xrightarrow{\sigma} \textsc{Top}^{\phi_{j+1}}_{\mathcal{A}/\mathcal{P}}$, hence we have $m_1[Q_1]^{\nu_1''} \mid m_2[Q_2]^{\nu_2''} \mid \textsc{Top}^{\phi_j}_{\mathcal{A}/\mathcal{P}} \xrightarrow{\sigma} m_1[\hat{Q}_1]^{\nu_1''} \mid m_2[\hat{Q}_2]^{\nu_2''} \mid \textsc{Top}^{\phi_{j+1}}_{\mathcal{A}/\mathcal{P}}$.
- $\alpha = !w{\triangleright}\nu$. We observe: (i) the environment of the system contains just the node *obs* and (ii) $\mathsf{Env}\,(N) = \{m_1, m_2\}$. Thus there exists $i \in \{1, 2\}$ such that the transition has been derived just by rule (Obs) from the following premise

  $$m_1[Q_1]^{\nu_1'} \mid m_2[Q_2]^{\nu_2'} \mid N \xrightarrow{m_i!w{\triangleright}obs} m_1[\hat{Q}_1]^{\nu_1'} \mid m_2[\hat{Q}_2]^{\nu_2'} \mid \hat{N} \ .$$

  Without loss of generality we assume $i = 1$, then we have $m_1[Q_1]^{\nu_1'} \xrightarrow{m_1!w{\triangleright}\nu_1'} m_1[\hat{Q}_1]^{\nu_1'}$, $m_2[Q_2]^{\nu_2'} \xrightarrow{m_1?w} m_2[\hat{Q}_2]^{\nu_2'}$ and $N \xrightarrow{m_1?w} \hat{N}$. Now, to prove the similarity, we need to simulate the $m_1?w$-action at the node $m_2[Q_2]^{\nu_2''}$ which cannot actually receive packets from $m_1 \notin \nu_2''$. We first observe that the message w can be eavesdropped by an attacker at the time interval j, thus $w \in \mathcal{D}(\phi_j)$ thanks to time-dependent stability. Then $\textsc{Top}^{\phi_j}_{\mathcal{A}/\mathcal{P}} \xRightarrow{a_2!w{\triangleright}m_2} \textsc{Top}^{\phi_j}_{\mathcal{A}/\mathcal{P}}$. Since $a_2 \in \nu_2''$ we have $m_2[Q_2]^{\nu_2''} \xrightarrow{a_2?w} m_2[\hat{Q}_2]^{\nu_2''}$. Finally, $m_1[Q_1]^{\nu_1''} \xrightarrow{a_2?w} m_1[Q_1]^{\nu_1''}$ by rule (RcvEnb). Thus, by applying rule (Bcast) we obtain

  $$m_1[Q_1]^{\nu_1''} \mid m_2[Q_2]^{\nu_2''} \mid \textsc{Top}^{\phi_j}_{\mathcal{A}/\mathcal{P}} \xRightarrow{a_2!w{\triangleright}\emptyset} m_1[Q_1]^{\nu_1''} \mid m_2[\hat{Q}_2]^{\nu_2''} \mid \textsc{Top}^{\phi_j}_{\mathcal{A}/\mathcal{P}} \ .$$

  By rule (Shh) $m_1[Q_1]^{\nu_1''} \mid m_2[Q_2]^{\nu_2''} \mid \textsc{Top}^{\phi_j}_{\mathcal{A}/\mathcal{P}} \stackrel{\tau}{\Longrightarrow} m_1[Q_1]^{\nu_1''} \mid m_2[\hat{Q}_2]^{\nu_2''} \mid \textsc{Top}^{\phi_j}_{\mathcal{A}/\mathcal{P}}$. Now, $m_1[Q_1]^{\nu_1''} \xrightarrow{m_1!w{\triangleright}\nu_1''} m_1[\hat{Q}_1]^{\nu_1''}$ and by rule (RcvEnb) we

have both $m_2[\hat{Q}_2]^{\nu_2''} \xrightarrow{m_1?w} m_2[\hat{Q}_2]^{\nu_2''}$ and $\text{TOP}^{\phi_j}_{\mathcal{A}/\mathcal{P}} \xrightarrow{m_1?w} \text{TOP}^{\phi_j}_{\mathcal{A}/\mathcal{P}}$. Thus

$$m_1[Q_1]^{\nu_1''} \mid m_2[\hat{Q}_2]^{\nu_2''} \mid \text{TOP}^{\phi_j}_{\mathcal{A}/\mathcal{P}} \xrightarrow{m_1!w\triangleright obs} m_1[\hat{Q}_1]^{\nu_1''} \mid m_2[\hat{Q}_2]^{\nu_2''} \mid \text{TOP}^{\phi_j}_{\mathcal{A}/\mathcal{P}}.$$

$\alpha = \tau$. The most significant case is an application of rule (Shh), from the premise $m_1[Q_1]^{\nu_1'} \mid m_2[Q_2]^{\nu_2'} \mid N \xrightarrow{m_1!w\triangleright\emptyset} m_1[\hat{Q}_1]^{\nu_1'} \mid m_2[\hat{Q}_2]^{\nu_2'} \mid \hat{N}$. Since $obs \in \nu_1' \cap \nu_2'$, the broadcast action must be performed by N; thus there exists $i \in \{1,2\}$ such that $N \xrightarrow{a_i!w\triangleright m_i} \hat{N}$ and $m_l[Q_l]^{\nu_l'} \xrightarrow{a_i?w} m_l[\hat{Q}_l]^{\nu_l'}$, for $l = 1,2$. Now also $\text{TOP}^{\phi_j}_{\mathcal{A}/\mathcal{P}} \xrightarrow{a_i!w\triangleright m_i} \text{TOP}^{\phi_j}_{\mathcal{A}/\mathcal{P}}$ and $m_l[Q_l]^{\nu_l''} \xrightarrow{a_i?w} m_l[\hat{Q}_l]^{\nu_l''}$, for $l = 1,2$. Thus $m_1[Q_1]^{\nu_1''} \mid m_2[Q_2]^{\nu_2''} \mid \text{TOP}^{\phi_j}_{\mathcal{A}/\mathcal{P}} \xrightarrow{\tau} m_1[\hat{Q}_1]^{\nu_1''} \mid m_2[\hat{Q}_2]^{\nu_2''} \mid \text{TOP}^{\phi_j}_{\mathcal{A}/\mathcal{P}}$. □

Lemma 3. *If M is time-dependent stable with respect to a sequence of knowledge $\{\phi_j\}_{j\geq 0}$, $\mathcal{A}$ is a set of attacking nodes for M and $\mathcal{O} \subseteq \mathsf{nds}(M)$ then*

$$M^{\mathcal{A}}_{\mathcal{O}} \mid A \lesssim M^{\mathcal{A}}_{\mathcal{O}} \mid \text{TOP}^{\phi_0}_{\mathcal{A}/\mathsf{nds}(M)} \quad \textit{for every } A \in \mathbb{A}^{\phi_0}_{\mathcal{A}/\mathsf{nds}(M)} \ .$$

Proof We prove the lemma in the most general case, that is $\mathcal{O} = \mathsf{nds}(M)$. Then we fix an arbitrary $A \in \mathbb{A}^{\phi_0}_{\mathcal{A}/\mathsf{nds}(M)}$ and we define the proper simulation as follows:

$$\mathcal{R} \stackrel{\text{def}}{=} \bigcup_{j\geq 0} \Big\{ \big(M' \mid A',\ M' \mid \text{TOP}^{\phi_j}_{\mathcal{A}/\mathsf{nds}(M)}\big) \text{ s.t. } M^{\mathcal{A}} \mid A \stackrel{\Lambda}{\Longrightarrow} M' \mid A' \text{ with } \mathsf{nds}(M') = \mathsf{nds}\big(M^{\mathcal{A}}\big) \text{ and } \#^{\sigma}(\Lambda) = j \Big\}$$

We let $\big(M' \mid A',\ M' \mid \text{TOP}^{\phi_j}_{\mathcal{A}/\mathsf{nds}(M)}\big) \in \mathcal{R}$. We make a case analysis on why $M' \mid A' \xrightarrow{\alpha} N$.

$\alpha = m?w$. As for Lemma 2, this case is straightforward.

$\alpha = \sigma$. Then $N = M'' \mid A''$ with $M' \xrightarrow{\sigma} M''$ and $A' \xrightarrow{\sigma} A''$. Now also $\text{TOP}^{\phi_j}_{\mathcal{A}/\mathsf{nds}(M)} \xrightarrow{\sigma} \text{TOP}^{\phi_{j+1}}_{\mathcal{A}/\mathsf{nds}(M)}$ by rule (σ-Sum), hence by rule (σ-Par) we have $M' \mid \text{TOP}^{\phi_j}_{\mathcal{A}/\mathsf{nds}(M)} \xrightarrow{\sigma} M'' \mid \text{TOP}^{\phi_{j+1}}_{\mathcal{A}/\mathsf{nds}(M)}$.

$\alpha = !w\triangleright\nu$. Since the environment of the system contains just the node *obs*, the transition has to be derived by the rule (Obs) whose premise is $M' \mid A' \xrightarrow{m!w\triangleright obs} N$. Since $obs \notin \mathsf{Env}(A')$ then $m \in \mathsf{nds}(M')$ and $N = M'' \mid A''$ with $M' \xrightarrow{m!w\triangleright\nu'} M''$, $\{obs\} = \nu' \setminus \mathsf{nds}(A')$ and $A' \xrightarrow{m?w} A''$. Now we have $\text{TOP}^{\phi_j}_{\mathcal{A}/\mathsf{nds}(M)} \xrightarrow{m?w} \text{TOP}^{\phi_j}_{\mathcal{A}/\mathsf{nds}(M)}$ by rule (RcvEnb). Hence $M' \mid \text{TOP}^{\phi_j}_{\mathcal{A}/\mathsf{nds}(M)} \xrightarrow{m!w\triangleright obs} M'' \mid \text{TOP}^{\phi_j}_{\mathcal{A}/\mathsf{nds}(M)}$ by rule (Bcast) and the fact that $\mathsf{nds}(A') = \mathcal{A} = \mathsf{nds}\Big(\text{TOP}^{\phi_j}_{\mathcal{A}/\mathsf{nds}(M)}\Big)$. Finally, by rule (Obs): $M' \mid \text{TOP}^{\phi_j}_{\mathcal{A}/\mathsf{nds}(M)} \xrightarrow{!w\triangleright obs} M'' \mid \text{TOP}^{\phi_j}_{\mathcal{A}/\mathsf{nds}(M)}$.

$\alpha = \tau$. The most significant case is when τ is derived by an application of rule (Shh), then we have $M' \mid A' \xrightarrow{a!w\triangleright\emptyset} N$ and $a \in \mathsf{nds}(A') = \mathcal{A}$ since the broadcast from any of the nodes in $\mathsf{nds}(M') = \mathsf{nds}(M^{\mathcal{A}})$ can be observed by the node obs. In this case we have $M' \xrightarrow{a?w} M''$ and $A' \xrightarrow{a!w\triangleright m} A''$ where m is the single node of M attacked by a. Now also $\textsc{Top}^{\phi_j}_{\mathcal{A}/\mathsf{nds}(M)} \xrightarrow{\tau} \xrightarrow{a!w\triangleright m} \textsc{Top}^{\phi_j}_{\mathcal{A}/\mathsf{nds}(M)}$ by rules (Tau) and (Snd) since the attacking node associated to m does not change and $\mathrm{msg}(A') \subseteq \mathcal{D}(\phi_j)$. Hence, by rule (Bcast): $M' \mid \textsc{Top}^{\phi_j}_{\mathcal{A}/\mathsf{nds}(M)} \xRightarrow{a!w\triangleright\emptyset} M'' \mid \textsc{Top}^{\phi_j}_{\mathcal{A}/\mathsf{nds}(M)}$. Thus $M' \mid \textsc{Top}^{\phi_j}_{\mathcal{A}/\mathsf{nds}(M)} \xRightarrow{\tau} M'' \mid \textsc{Top}^{\phi_j}_{\mathcal{A}/\mathsf{nds}(M)}$ by rule (Shh). □

Proof of Theorem 2 By Lemma 3 we have $M^{\mathcal{A}}_{\mathcal{O}} \mid A \lesssim M^{\mathcal{A}}\mathcal{O} \mid \textsc{Top}^{\phi_0}_{\mathcal{A}/\mathsf{nds}(M)}$ for every $A \in \mathbb{A}^{\phi_0}_{\mathcal{A}/\mathsf{nds}(M)}$. Then by transitivity of $\lesssim$ we have $M^{\mathcal{A}}\mathcal{O} \mid A \lesssim N$ for every $A \in \mathbb{A}^{\phi_0}_{\mathcal{A}/\mathsf{nds}(M)}$ and we conclude that M is $tGNDC^{N}_{\phi_0,\mathcal{O}}$. □

Proof of Theorem 3 By Theorem 1 we have

$$(M_1)^{\mathcal{A}_1}_{\mathcal{O}_1} \mid \ldots \mid (M_k)^{\mathcal{A}_k}_{\mathcal{O}_k} \mid \textsc{Top}^{\phi_0}_{\mathcal{A}_1/\mathsf{nds}(M_1)} \mid \ldots \mid \textsc{Top}^{\phi_0}_{\mathcal{A}_k/\mathsf{nds}(M_k)} \lesssim N_1 \mid \ldots \mid N_k \ .$$

By applying Lemma 2 and Theorem 1 we obtain

$$(M_1 \mid \ldots \mid M_k)^{\mathcal{A}_1\uplus\ldots\uplus\mathcal{A}_k}_{\mathcal{O}_1\uplus\ldots\uplus\mathcal{O}_k} \mid \textsc{Top}^{\phi_0}_{\mathcal{A}_1\uplus\ldots\uplus\mathcal{A}_k/\mathsf{nds}(M_1\mid\ldots\mid M_k)} \lesssim N_1 \mid \ldots \mid N_k \ .$$

Thus, by an application of Theorem 2 we can derive $M \in tGNDC^{N_1\mid\ldots\mid N_k}_{\phi_0,\mathcal{O}_1\uplus\ldots\uplus\mathcal{O}_k}$. □

A.5 Proofs of Section 4

Proof of Proposition 3 By induction on i we show that whenever $\textsc{kl}[\hat{L}_0]^{\nu_{\textsc{kl}}} \xRightarrow{\Lambda} \textsc{kl}[\hat{L}_i]^{obs}$ or $m[\hat{Z}_0]^{obs} \xRightarrow{\Lambda} m[\hat{Z}_i]^{obs}$ then $\#^{\sigma}(\Lambda) = 2i$. Moreover, for every $i \geq 1$:

- action $!q_i\triangleright obs$ can be performed exclusively because
$$\textsc{kl}[\hat{L}_{i-1}]^{obs} \xrightarrow{\tau} \xrightarrow{\sigma} \xrightarrow{!q_i\triangleright obs}$$
- action $!\mathsf{auth}_i\triangleright obs$ can be performed exclusively because
$$m[\hat{Z}_{i-1}]^{obs} \xRightarrow{\Lambda} \xrightarrow{!\mathsf{auth}_i\triangleright obs}$$
with $\#^{\sigma}\Lambda = 3$.

Hence we deduce that:

1. if $\textsc{kl}[\hat{L}_0]^{obs} \xRightarrow{\Lambda} \xrightarrow{!q_i\triangleright obs}$ then $\#^{\sigma}(\Lambda) = 2i+1$.
2. if $m[\hat{Z}_1]^{obs} \xRightarrow{\Lambda} \xrightarrow{!\mathsf{auth}_i\triangleright obs}$ then $\#^{\sigma}(\Lambda) = 2i+3$.

Now, the result is a straightforward consequence of these two properties. □

Proof of Theorem 4 Let $\nu'_m = \{\text{KL}, a, obs\}$ and $\nu'_{\text{KL}} = \{m, a, obs\}$. The system $(\text{LiSP}')^{\mathcal{A}} \mid A$ performs the following computation:

$$
\begin{array}{ll}
(\text{LiSP}')^{\mathcal{A}} \mid A & \xrightarrow{\sigma} \\
m[Z']^{\nu'_m} \mid \text{KL}[L_0]^{\nu'_{\text{KL}}} \mid a[\sigma.\sigma.X]^{\nu_a} \mid b[\sigma.X]^{\nu_b} & \xrightarrow{!r \rhd obs} \\
m[\sigma.\lfloor ?(q).T' \rfloor Z']^{\nu'_m} \mid \text{KL}[\{^{r}/_{r}\} I_1]^{\nu'_{\text{KL}}} \mid a[\sigma.\sigma.X]^{\nu_a} \mid b[\sigma.X]^{\nu_b} & \xrightarrow{\sigma} \\
m[\lfloor ?(q).T' \rfloor Z']^{\nu'_m} \mid \text{KL}[!\langle q_1 \rangle.\sigma.L_1]^{\nu'_{\text{KL}}} \mid a[\sigma.X]^{\nu_a} \mid b[X]^{\nu_b} & \xrightarrow{!q_1 \rhd obs} \\
m[\lfloor ?(q).T' \rfloor Z']^{\nu'_m} \mid \text{KL}[\sigma.L_1]^{\nu'_{\text{KL}}} \mid a[\sigma.X]^{\nu_a} \mid b[\sigma.!\langle q_1 \rangle.\mathsf{nil}]^{\nu_b} & \xrightarrow{\sigma} \\
m[Z']^{\nu'_m} \mid \text{KL}[L_1]^{\nu'_{\text{KL}}} \mid a[X]^{\nu_a} \mid b[!\langle q_1 \rangle.\mathsf{nil}]^{\nu_b} & \xrightarrow{\tau} \\
m[Z']^{\nu'_m} \mid \text{KL}[\{^{q_1}/_{r}\} I_2]^{\nu'_{\text{KL}}} \mid a[\sigma.!\langle q_1 \rangle.\mathsf{nil}]^{\nu_a} \mid b[\mathsf{nil}]^{\nu_b} & \xrightarrow{!r \rhd obs} \\
m[\sigma.\lfloor ?(q).T' \rfloor Z']^{\nu'_m} \mid \text{KL}[\{^{q_1}/_{r}\} I_2]^{\nu'_{\text{KL}}} \mid a[\sigma.!\langle q_1 \rangle.\mathsf{nil}]^{\nu_a} \mid b[\mathsf{nil}]^{\nu_b} & \xrightarrow{\sigma} \\
m[\lfloor ?(q).T' \rfloor Z']^{\nu'_m} \mid \text{KL}[\sigma.L_2]^{\nu'_{\text{KL}}} \mid a[!\langle q_1 \rangle.\mathsf{nil}]^{\nu_a} \mid b[\mathsf{nil}]^{\nu_b} & \xrightarrow{\tau} \\
m[\{^{q_1}/_{q}\} T']^{\nu'_m} \mid \text{KL}[\sigma.L_2]^{\nu'_{\text{KL}}} \mid a[\mathsf{nil}]^{\nu_a} \mid b[\mathsf{nil}]^{\nu_b} & \xrightarrow{\sigma} \\
m[\sigma.!\langle \mathsf{auth}_1 \rangle.R(k_2, k_{s+1}, s-1)]^{\nu'_m} \mid \text{KL}[L_2]^{\nu'_{\text{KL}}} \mid a[\mathsf{nil}]^{\nu_a} \mid b[\mathsf{nil}]^{\nu_b} & \xrightarrow{\sigma} \\
m[!\langle \mathsf{auth}_1 \rangle.R(k_2, k_{s+1}, s-1)]^{\nu'_m} \mid \text{KL}[\sigma.L_3]^{\nu'_{\text{KL}}} \mid a[\mathsf{nil}]^{\nu_a} \mid b[\mathsf{nil}]^{\nu_b} & \xrightarrow{!\mathsf{auth}_1 \rhd obs}
\end{array}
$$

Then m signals the correct reconfiguration based on an old packet. Hence timed integrity property does not hold. □

Proof of Proposition 4 Similar to the proof of Proposition 3. □

Proof of Lemma 1 We provide the proper simulation in both cases.
Case 1: Key Server. To show that $\text{KL}[\sigma.\bar{L}_0]^{\{b,obs\}} \mid \text{Top}^{\phi_0}_{b/\text{KL}} \lesssim \text{KL}[\sigma.\hat{L}'_0]^{obs}$ we define the relation $\mathcal{R}_i(v, n, w)$:

$$
\begin{array}{l}
\Big\{ \Big(\text{KL}[\bar{L}'_i]^{\{b,obs\}} \mid \text{Top}^{\phi_{2i+1}}_{b/\text{KL}},\ \text{KL}[\hat{L}'_i]^{obs} \Big) , \\
\Big(\text{KL}[\bar{L}'_i]^{\{b,obs\}} \mid b[!\langle v \rangle.\text{T}_{\phi_{2i+1}}]^{\text{KL}},\ \text{KL}[\hat{L}'_i]^{obs} \Big) , \\
\Big(\text{KL}[\{^{v}/_{r}\} \bar{I}'_{i+1}]^{\{b,obs\}} \mid \text{Top}^{\phi_{2i+1}}_{b/\text{KL}},\ \text{KL}[\hat{L}'_i]^{obs} \Big) , \\
\Big(\text{KL}[\bar{L}'_i]^{\{b,obs\}} \mid b[!\langle v \rangle.\text{T}_{\phi_{2i+1}}]^{\text{KL}},\ \text{KL}[\hat{L}'_i]^{obs} \Big) , \\
\Big(\text{KL}[!\langle q^n_{i+1} \rangle.\sigma.\bar{L}'_{i+1}]^{\{b,obs\}} \mid \text{Top}^{\phi_{2(i+1)}}_{b/\text{KL}},\ \text{KL}[!\langle q^n_{i+1} \rangle.\sigma.\hat{L}'_{i+1}]^{obs} \Big) , \\
\Big(\text{KL}[!\langle q^n_{i+1} \rangle.\sigma.\bar{L}'_{i+1}]^{\{b,obs\}} \mid b[!\langle w \rangle.\text{T}_{\phi_{2(i+1)}}]^{\text{KL}},\ \text{KL}[!\langle q^n_{i+1} \rangle.\sigma.\hat{L}'_{i+1}]^{obs} \Big) , \\
\Big(\text{KL}[\sigma.\bar{L}'_{i+1}]^{\{b,obs\}} \mid \text{Top}^{\phi_{2(i+1)}}_{b/\text{KL}},\ \text{KL}[\sigma.\hat{L}'_{i+1}]^{obs} \Big) , \\
\Big(\text{KL}[\sigma.\bar{L}'_{i+1}]^{\{b,obs\}} \mid b[!\langle w \rangle.\text{T}_{\phi_{2(i+1)}}]^{\text{KL}},\ \text{KL}[\sigma.\hat{L}'_{i+1}]^{obs} \Big) \Big\} \ .
\end{array}
$$

Then we define

$$
\mathcal{R} \ \stackrel{\text{def}}{=} \ \bigcup_{i \geq 0} \ \bigcup_{\substack{v,\, n \,\in\, \mathcal{D}(\phi_{2i+1}) \\ w \,\in\, \mathcal{D}(\phi_{2(i+1)})}} \mathcal{R}_i(v, n, w) \ .
$$

It is now straightforward to check that the following relation is a simulation:

$$\mathcal{R} \quad \cup \quad \left\{ \, \mathrm{KL}[\sigma.\bar{L}_0]^{\{b,obs\}} \;\middle|\; \mathrm{Top}^{\phi_0}_{b/\mathrm{KL}},\ \mathrm{KL}[\sigma.\hat{L}'_0]^{obs} \, \right\} .$$

We outline the two most significant cases. We omit input actions since the environment contains exclusively the node *obs* which cannot transmit, thus all input actions can be derived just by combining rules (RcvEnb) and (RcvPar). We also omit internal choices of the attacker.

The pair $\left(\mathrm{KL}[\bar{L}'_i]^{\{b,obs\}} \mid b[!\langle v\rangle.\mathrm{T}_{\phi_{2i+1}}]^{\mathrm{KL}},\ \mathrm{KL}[\hat{L}'_i]^{obs}\right)$ has two significant actions:

- $\mathrm{KL}[\bar{L}'_i]^{\{b,obs\}} \mid b[!\langle v\rangle.\mathrm{T}_{\phi_{2i+1}}]^{\mathrm{KL}} \xrightarrow{\tau} \mathrm{KL}[\{^v/_r\}\bar{I}'_{i+1}]^{\{b,obs\}} \mid \mathrm{Top}^{\phi_{2i+1}}_{b/\mathrm{KL}},\ \mathrm{KL}[\hat{L}'_i]^{obs}$ where KL receives v. Then $\mathrm{KL}[\hat{L}'_i]^{obs} \Longrightarrow \mathrm{KL}[\hat{L}'_i]^{obs}$.
- $\mathrm{KL}[\bar{L}'_i]^{\{b,obs\}} \mid b[!\langle v\rangle.\mathrm{T}_{\phi_{2i+1}}]^{\mathrm{KL}} \xrightarrow{\tau} \mathrm{KL}[\bar{L}'_i]^{\{b,obs\}} \mid \mathrm{Top}^{\phi_{2i+1}}_{b/\mathrm{KL}}$ where v gets lost. Then the second network $\mathrm{KL}[\hat{L}'_i]^{obs} \Longrightarrow \mathrm{KL}[\hat{L}'_i]^{obs}$.

The pair $\left(\mathrm{KL}[\{^v/_r\}\bar{I}'_{i+1}]^{\{b,obs\}} \mid \mathrm{Top}^{\phi_{2i+1}}_{b/\mathrm{KL}},\ \mathrm{KL}[\hat{L}'_i]^{obs}\right)$ has two significant actions:

- $\mathrm{KL}[\{^v/_r\}\bar{I}'_{i+1}]^{\{b,obs\}} \mid \mathrm{Top}^{\phi_{2i+1}}_{b/\mathrm{KL}} \xrightarrow{\sigma} \mathrm{KL}[!\langle q^n_{i+1}\rangle.\sigma.\bar{L}'_{i+1}]^{\{b,obs\}} \mid \mathrm{Top}^{\phi_{2(i+1)}}_{b/\mathrm{KL}}$ where KL checks that v represents a correct RequestKey packet and n is as a possible nonce. Then $\mathrm{KL}[\hat{L}'_i]^{obs} \overset{\sigma}{\Longrightarrow} \mathrm{KL}[!\langle q^n_{i+1}\rangle.\sigma.\hat{L}'_{i+1}]^{obs}$.
- $\mathrm{KL}[\{^v/_r\}\bar{I}'_{i+1}]^{\{b,obs\}} \mid \mathrm{Top}^{\phi_{2i+1}}_{b/\mathrm{KL}} \xrightarrow{\sigma} \mathrm{KL}[\sigma.\bar{L}'_{i+1}]^{\{b,obs\}} \mid \mathrm{Top}^{\phi_{2(i+1)}}_{b/\mathrm{KL}}$ when v is not a correct RequestKey packet. Then $\mathrm{KL}[\hat{L}'_i]^{obs} \overset{\sigma}{\Longrightarrow} \mathrm{KL}[\sigma.\hat{L}'_{i+1}]^{obs}$.

Case 2: Node. To show that $m[\sigma.Z_1]^{\{a,obs\}} \mid \textsc{Top}^{\phi_0}_{a/m} \lesssim m[\sigma.\hat{Z}'_1]^{obs}$ we define the relation $\mathcal{R}_i(v_0, v_1, v_2, v_3)$:

$$\Big\{ \Big(m[Z_i]^{\{a,obs\}} \mid \textsc{Top}^{\phi_{2i-1}}_{a/m},\ m[\hat{Z}'_i]^{obs} \Big) ,$$
$$\Big(m[Z_i]^{\{a,obs\}} \mid a[!\langle v_0\rangle.\mathrm{T}_{\phi_{2i-1}}]^m,\ m[\hat{Z}'_i]^{obs} \Big) ,$$
$$\Big(m[\sigma.\lfloor ?(q).T_i \rfloor Z_{i+1}]^{\{a,obs\}} \mid \textsc{Top}^{\phi_{2i-1}}_{a/m},\ m[\sigma.\lfloor \tau.\sigma.\sigma.!\langle \mathsf{auth}_{i+1}\rangle.\mathsf{nil} \rfloor \hat{Z}'_{i+1}]^{obs} \Big) ,$$
$$\Big(m[\sigma.\lfloor ?(q).T_i \rfloor Z_{i+1}]^{\{a,obs\}} \mid a[!\langle v_0\rangle.\mathrm{T}_{\phi_{2i-1}}]^m,\ m[\sigma.\lfloor \tau.\sigma.\sigma.!\langle \mathsf{auth}_{i+1}\rangle.\mathsf{nil} \rfloor \hat{Z}'_{i+1}]^{obs} \Big) ,$$
$$\Big(m[\lfloor ?(q).T_i \rfloor Z_{i+1}]^{\{a,obs\}} \mid \textsc{Top}^{\phi_{2i}}_{a/m},\ m[\lfloor \tau.\sigma.\sigma.!\langle \mathsf{auth}_{i+1}\rangle.\mathsf{nil} \rfloor \hat{Z}'_{i+1}]^{obs} \Big) ,$$
$$\Big(m[\lfloor ?(q).T_i \rfloor Z_{i+1}]^{\{a,obs\}} \mid a[!\langle v_1\rangle.\mathrm{T}_{\phi_{2i}}]^m,\ m[\lfloor \tau.\sigma.\sigma.!\langle \mathsf{auth}_{i+1}\rangle.\mathsf{nil} \rfloor \hat{Z}'_{i+1}]^{obs} \Big) ,$$
$$\Big(m[\{^{v_1}/_q\}T_i]^{\{a,obs\}} \mid \textsc{Top}^{\phi_{2i}}_{a/m},\ m[\lfloor \tau.\sigma.\sigma.!\langle \mathsf{auth}_{i+1}\rangle.\mathsf{nil} \rfloor \hat{Z}'_{i+1}]^{obs} \Big) ,$$
$$\Big(m[\{^{v_1}/_q\}T_i]^{\{a,obs\}} \mid a[!\langle v_1\rangle.\mathrm{T}_{\phi_{2i}}]^m,\ m[\lfloor \tau.\sigma.\sigma.!\langle \mathsf{auth}_{i+1}\rangle.\mathsf{nil} \rfloor \hat{Z}'_{i+1}]^{obs} \Big) ,$$
$$\Big(m[\sigma.!\langle \mathsf{auth}_{i+1}\rangle.\mathsf{nil}]^{\{a,obs\}} \mid \textsc{Top}^{\phi_{2i+1}}_{a/m},\ m[\sigma.!\langle \mathsf{auth}_{i+1}\rangle.\mathsf{nil}]^{obs} \Big) ,$$
$$\Big(m[\sigma.!\langle \mathsf{auth}_{i+1}\rangle.\mathsf{nil}]^{\{a,obs\}} \mid a[!\langle v_2\rangle.\mathrm{T}_{\phi_{2i+1}}]^m,\ m[\sigma.!\langle \mathsf{auth}_{i+1}\rangle.\mathsf{nil}]^{obs} \Big) ,$$
$$\Big(m[!\langle \mathsf{auth}_{i+1}\rangle.\mathsf{nil}]^{\{a,obs\}} \mid \textsc{Top}^{\phi_{2i+2}}_{a/m},\ m[!\langle \mathsf{auth}_{i+1}\rangle.\mathsf{nil}]^{obs} \Big) ,$$
$$\Big(m[!\langle \mathsf{auth}_{i+1}\rangle.\mathsf{nil}]^{\{a,obs\}} \mid a[!\langle v_3\rangle.\mathrm{T}_{\phi_{2i+2}}]^m,\ m[!\langle \mathsf{auth}_{i+1}\rangle.\mathsf{nil}]^{obs} \Big) ,$$
$$\Big(m[\mathsf{nil}]^{\{a,obs\}} \mid \textsc{Top}^{\phi_{2i+2}}_{a/m},\ m[\mathsf{nil}]^{obs} \Big) ,$$
$$\Big(m[\mathsf{nil}]^{\{a,obs\}} \mid a[!\langle v_3\rangle.\mathrm{T}_{\phi_{2i+2}}]^m,\ m[\mathsf{nil}]^{obs} \Big) \Big\} \ .$$

Then we define

$$\mathcal{R} \ \stackrel{\text{def}}{=} \ \bigcup_{i\geq 0} \ \bigcup_{\substack{v_j \in \mathcal{D}(\phi_{(2i-1)+j}) \\ 0 \leq j \leq 3}} \mathcal{R}_i(v_0, v_1, v_2, v_3) \ .$$

It is now straightforward to check that the following relation is a simulation:

$$\mathcal{R} \ \cup \ \{ \, m[\sigma.\bar{L}_0]^{\{b,obs\}} \mid \textsc{Top}^{\phi_0}_{a/m},\ m[\sigma.\hat{L}'_0]^{obs} \, \} \ .$$

Again, we outline the most significant case.
The pair $\Big(m[\{^{v_1}/_q\}T_i]^{\{a,obs\}} \mid \textsc{Top}^{\phi_{2i}}_{a/m},\ m[\lfloor \tau.\sigma.\sigma.!\langle \mathsf{auth}_{i+1}\rangle.\mathsf{nil} \rfloor \hat{Z}'_{i+1}]^{obs} \Big)$ has two significant actions:

- $m[\{^{v_1}/_q\}T_i]^{\{a,obs\}} \mid \textsc{Top}^{\phi_{2i}}_{a/m} \xrightarrow{\ \sigma\ } m[\sigma.!\langle \mathsf{auth}_{i+1}\rangle.\mathsf{nil}]^{\{a,obs\}} \mid \textsc{Top}^{\phi_{2i+1}}_{a/m}$ where m checks that v is a correct $\mathsf{InitKey}$ packet and it contains the current nonce n_i. Since v is encrypted and contains n_i it can only be generated by KS just a σ action before, thus it contains the key k_{s+i+1}. Then the second network $m[\lfloor \tau.\sigma.\sigma.!\langle \mathsf{auth}_{i+1}\rangle.\mathsf{nil} \rfloor \hat{Z}'_{i+1}]^{obs} \stackrel{\sigma}{\Longrightarrow} m[\sigma.!\langle \mathsf{auth}_{i+1}\rangle.\mathsf{nil}]^{obs}$.
- $m[\{^{v_1}/_q\}T_i]^{\{a,obs\}} \mid \textsc{Top}^{\phi_{2i}}_{a/m} \xrightarrow{\ \sigma\ } m[Z_{i+1}]^{\{b,obs\}} \mid \textsc{Top}^{\phi_{2i+1}}_{a/m}$ where m cannot verify that v contains the current nonce. Then $m[\hat{L}'_i]^{obs} \stackrel{\sigma}{\Longrightarrow} m[\hat{Z}'_{i+1}]^{obs}$. □