A Semantic Analysis of Key Management Protocols for Wireless Sensor Networks☆

Damiano Macedonio, Massimo Merro

Dipartimento di Informatica, Università degli Studi di Verona, Italy

Abstract

Gorrieri and Martinelli's *timed Generalized Non-Deducibility on Compositions* (*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 three well-known *key management protocols* for wireless sensor networks: μTESLA, LEAP+ and LiSP.

Keywords: Wireless sensor network, key management protocol, security analysis, process calculus

1. Introduction

Wireless sensors are small and cheap devices powered by low-energy batteries, equipped with radio transceivers, and responding to physical stimuli, such as pressure, magnetism and motion, by emitting radio signals. Such devices are featured with *resource constraints* (involving power, storage and computation) and low transmission rates. *Wireless sensor networks* (WSNs) are large-scale networks of sensor nodes deployed in strategic areas to gather data. Sensor nodes collaborate using wireless communications with an asymmetric many-to-one data transfer model. Typically, they send their sensed events or data to a specific node, called sink node or base station, which collects the requested information. WSNs are primarily designed for monitoring environments that humans cannot easily reach (e.g., motion, target tracking, fire detection, chemicals, temperature); they are used as embedded systems (e.g., biomedical sensor engineering, smart homes) or mobile applications (e.g., when attached to robots, soldiers, or vehicles).

An important issue in WSNs is *network security*: Sensor nodes are vulnerable to several kinds of threats and risks. Unlike wired networks, wireless devices use radio frequency channels to broadcast their messages. An adversary can compromise a sensor node, alter the integrity of the data, eavesdrop on messages, inject fake messages, and waste network resource. Thus, one of the challenges in developing trustworthy WSNs is to provide high-security features with limited resources.

Generally, in order to have a secure communication between two (or more) parties, a secure association must be established by sharing a secret. This secret must be created, distributed and updated by one (or more) entity and it is often represented by the knowledge of a *cryptographic key*. The management of such cryptographic keys is the core of any security protocol. Due to resource limitations, all key management protocols for WSNs, such as μTESLA [1], LiSP [2], LEAP [3], PEBL [4] and INF [5], are based on *symmetric cryptography* rather than heavy public-key schemes, such as Diffie-Hellman [6] and RSA [7].

In this paper, we adopt a process calculus approach to formalise and verify real-world key management protocols for WSNs. A process calculus is a formal and concise language that allows us to express system behaviour in the form of a process term. 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. [8]): A global clock is supposed

☆The first author is supported by research fellowship n. AdR1601/11, funded by Dipartimento di Informatica Verona.

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* [8]. The labelled transition semantics is used to derive a (weak) *simulation theory* which can be easily *mechanised* by relying on well-known interactive theorem provers such as Isabelle/HOL [10] or Coq [11].

Based on our simulation theory, we generalise Gorrieri and Martinelli's *timed Generalized Non-Deducibility on Compositions* (*tGNDC*) schema [12, 13], 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 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 order to avoid the universal quantification over all possible attackers when proving *tGNDC* properties, we provide a *compositional* proof technique based on the notion of *the most powerful attacker*.

We use our calculus to provide a formal specification of three well-known key management protocols for WSNs: (i) μTESLA [1], which achieves *authenticated broadcast*; (ii) the *Localized Encryption and Authentication Protocol*, LEAP+ [3], intended for large-scale wireless sensor networks; (iii) the *Lightweight Security Protocol*, LiSP [2], that, through an efficient mechanism of re-keying, provides a good trade-off between resource consumption and network security.

We perform a *tGNDC*-based analysis on these three protocols. As a result of our analysis, we formally prove that the *authenticated-broadcast phase* of μTESLA enjoys both timed integrity and timed agreement. Then, we prove that the *single-hop pairwise shared key* mechanism of LEAP+ enjoys timed integrity but not timed agreement, due to the presence of a replay attack despite the security assessment of [3]. Finally, we prove that the LiSP protocol satisfies neither timed integrity nor timed agreement. Again, this is due to the presence of a replay attack. To our knowledge both attacks are new and they have not yet appeared in the literature.

We end this introduction with an outline of the paper. In Section 2, we provide syntax, operational semantics and behavioural semantics of `aTCWS`. In the same section we prove that our calculus enjoys time determinism, maximal progress and patience. In Section 3, we adapt Gorrieri and Martinelli's *tGNDC* framework to `aTCWS`. In Sections 4, 5 and 6 we provide a security analysis of the three key management protocols mentioned above. The paper ends with a section on conclusions, future and related work.

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 $w, w' \ldots$ range over *Msg*.

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

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

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 system, the *deduction function* $\mathcal{D} : 2^{Msg} \to 2^{Msg}$ associates a (finite) set ϕ of

[1] Time synchronisation relies on some clock synchronisation protocol [9].

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
	$\mid$	$[u_1 \dots u_n \vdash_r x]P; Q$	deduction
	$\mid$	$H\langle\tilde{u}\rangle$	guarded recursion

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 [14]). 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 ($\nu \subseteq Nds \setminus \{n\}$). In other words, ν contains all nodes in the transmission cell of n (except n itself), thus modelling the network topology.[2] For simplicity, when $\nu = \{m\}$ we will omit parentheses. Our wireless networks have a fixed topology as node mobility is not relevant to our analysis.

Processes are sequential and live within the nodes. We let *Prc* be the set of all possible processes. We write nil to denote the skip process. The sender process $!\langle w\rangle.P$ allows to broadcast the message w, the continuation being P. The process $\lfloor ?(x).P\rfloor Q$ denotes a receiver with timeout. The process $\lfloor \sum_{i\in I} \tau.P_i\rfloor Q$ denotes internal choice with timeout. The process $\sigma.P$ models sleeping for one time unit. The process $[w_1 = w_2]P; Q$ is the standard "if then else" construct. The process $[w_1 \dots w_k \vdash_{\mathrm{r}} x]P; Q$ tries to infer a message w from the premises $w_1 \dots w_k$ by an application of rule r.

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 $[w_1 \dots w_n \vdash_{\mathrm{r}} x]P$ 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. In order to deal with (guarded) recursion, we assume a set *PrcIds* of process identifiers ranged over by H, H', H_1, H_2. We write $H\langle w_1, \dots, w_k\rangle$ to denote a recursive process H defined via an equation $H(x_1, \dots, x_k) = P$, where (i) the tuple $x_1, \dots, x_k$ 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 write Prc_{wt} for the set of processes in which summations are finite-indexed and recursive definitions are time-guarded.

Remark 2.1 *The recursion construct allows us to define persistent listeners, i.e., receivers which wait indefinitely for an incoming message, as* $Rcv = \lfloor ?(x).P\rfloor Rcv$*; similarly, internal choice (without timeout) can be defined as* $Sum = \lfloor \sum_{i\in I} \tau.P_i\rfloor Sum$*.*

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 $I = \emptyset$. We write $\sigma^k.P$ as an abbreviation for $\sigma. \dots .\sigma.P$, where prefix σ appears k times.

[2] We could have represented the topology in terms of a restriction operator à la CCS on node names; we have preferred our notation to keep at hand the neighbours of a node.

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

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 2.2 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 where all nodes have distinct names, $\mathsf{nds}\,(M)$ 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. Basically, it (i) rules out networks containing two nodes with the same name; (ii) rules out self-neighbouring; (iii) imposes symmetric neighbouring relations (we recall that all nodes have the same transmission range); (iv) imposes network connectivity to allow clock synchronisation.

Definition 2.3 (Well-formedness) *M is said to be* well-formed *if*

- *$M \equiv N \mid m_1[P_1]^{\nu_1} \mid m_2[P_2]^{\nu_2}$ implies $m_1 \neq m_2$;*
- *$M \equiv N \mid m[P]^{\nu}$ implies $m \notin \nu$;*
- *$M \equiv N \mid m_1[P_1]^{\nu_1} \mid m_2[P_2]^{\nu_2}$, with $m_1 \in \nu_2$, implies $m_2 \in \nu_1$;*
- *for all $m, n \in \mathsf{nds}\,(M)$ there are $m_1, \ldots, m_k \in \mathsf{nds}\,(M)$, such that $m{=}m_1$, $n{=}m_k$, $m_i \in \mathsf{ngh}(m_{i+1}, M)$, for $1 \leq i \leq k{-}1$.*

We let Net be the set of well-formed networks.

Henceforth, we will always work with networks in *Net*.

2.1. Labelled transition semantics

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 every global synchronisation all nodes proceed asynchronously by performing actions with no duration, which represent either broadcast or input or internal actions. Transmission proceeds even if there are no listeners: sending 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 loses 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. Rule (Tau) models local computations within a node due to a nondeterministic internal choice. Rule (TauPar) propagates internal computations on parallel components. The remaining rules model the passage of time. Rule (Sleep) models sleeping for one time slot. Rules (σ-nil) and (σ-$\mathbf{0}$) are straightforward.

Table 2 LTS - Transmissions, internal actions and time passing

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

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

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

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

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

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

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

Table 3 LTS - Matching, recursion and deduction

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

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

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

Rule (σ-Rcv) models timeout on receivers, and similarly rule (σ-Sum) describes timeout on internal activities. Rule (σ-Par) models time synchronisation between parallel components. 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 2.4 *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 $N = N_1 \mid N_2$.*

As the topology of our networks is static it is easy to prove the following result.

Proposition 2.5 *Let M be well-formed. If $M \xrightarrow{\lambda} M'$ then M' is well-formed.*

Proof By induction on the derivation of the transition $M \xrightarrow{\lambda} M'$. $\square$

2.2. Time properties

Our calculus aTCWS enjoys some desirable time properties. Here, we outline the most significant ones. Proposition 2.6 formalises the deterministic nature of time passing: a network can reach at most one new state by executing a σ-action.

Proposition 2.6 (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 $M \xrightarrow{\sigma} M'$. $\square$

Patience guarantees that a process will wait indefinitely until it can communicate [8]. 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 2.7 (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. $\square$

The maximal progress property says that processes communicate as soon as a possibility of communication arises [8]. In other words, the passage of time cannot block transmissions.

Proposition 2.8 (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. $\square$

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 [15] 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 2.9 (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 $h \leq k$.*

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) directly. 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 attackers) can stop the passage of time.

The following proposition provides a criterion to check well-timedness. We recall that Prc_{wt} denotes the set of processes where summations are always finite-indexed and recursive definitions are always time-guarded.

Proposition 2.10 *Let $M = \prod_{i \in I} m_i[P_i]^{\nu_i}$ be a network. If $P_i \in Prc_{\mathrm{wt}}$, for all $i \in I$, then M satisfies well-timedness.*

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

2.3. Behavioural semantics

Based on the LTS of Section 2.1, we define a standard notion of *timed labelled similarity* for `aTCWS`. In general, a simulation describes how a term (in our case a network) can mimic the actions of another term. Here, we focus on weak relations, i.e., we abstract on internal actions of networks. Thus, 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:

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

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. Note also that in a derivation tree the rule (Obs) can only be applied at top-level.

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 $\xRightarrow{\alpha}$ denotes $\Rightarrow\xrightarrow{\alpha}\Rightarrow$; the relation $\xRightarrow{\hat{\alpha}}$ denotes $\Rightarrow$ if $\alpha = \tau$ and $\xRightarrow{\alpha}$ otherwise.

Definition 2.11 (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 $N \xRightarrow{\hat{\alpha}} N'$ 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 similarity between networks is a pre-congruence, as it is preserved by parallel composition.

Theorem 2.12 *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

In order to achieve a formal verification of key management protocols for WSNs, we adopt a general schema for the definition of timed security properties, called *timed Generalized Non-Deducibility on Compositions* (*tGNDC*) [12], a real-time generalisation of *Generalized Non-Deducibility on Compositions* (*GNDC*) [16]. The main idea is the following: a system M is $tGNDC^{\rho(M)}$ if for every attacker A

$$M \;\big|\; A \;\lesssim\; \rho(M)$$

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 3.1 *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 constraints.

The knowledge of a network is expressed by the set of messages that the network can manipulate. Thus, we write msg(P) to denote the set of the messages that appear in the process P. Formally, we follow [12] and we define $\mathrm{msg} : Prc \to 2^{Msg}$ as the least set (fixed point) satisfying the rules in Table 4. A straightforward generalisation of msg( ) to networks is the following:

$$\mathrm{msg}(\mathbf{0}) \stackrel{\text{def}}{=} \emptyset\,; \quad \mathrm{msg}(n[P]^{\nu}) \stackrel{\text{def}}{=} \mathrm{msg}(P)\,; \quad \mathrm{msg}(M_1 \mid M_2) \stackrel{\text{def}}{=} \mathrm{msg}(M_1) \cup \mathrm{msg}(M_2)\,.$$

Table 4 Function msg_S

$$\text{msg}(\mathsf{nil}) \stackrel{\text{def}}{=} \emptyset$$
$$\text{msg}(!\langle u\rangle.P) \stackrel{\text{def}}{=} get(u) \cup \text{msg}(P)$$
$$\text{msg}(\lfloor ?(x).P \rfloor Q) \stackrel{\text{def}}{=} \text{msg}(P) \cup \text{msg}(Q)$$
$$\text{msg}(\lfloor \textstyle\sum_{i\in I} \tau.P_i \rfloor Q) \stackrel{\text{def}}{=} \bigcup_{i\in I} msg(P_i) \cup \text{msg}(Q)$$
$$\text{msg}(\sigma.P) \stackrel{\text{def}}{=} \text{msg}(P)$$
$$\text{msg}([u_1 = u_2]P; Q) \stackrel{\text{def}}{=} get(u_1) \cup get(u_2) \cup \text{msg}(P) \cup \text{msg}(Q)$$
$$\text{msg}([u_1 \dots u_n \vdash_r x]P; Q) \stackrel{\text{def}}{=} \bigcup_{i=1}^{n} get(u_i) \cup \text{msg}(P) \cup \text{msg}(Q)$$
$$\text{msg}(H\langle u_1 \dots u_r\rangle) \stackrel{\text{def}}{=} \bigcup_{i=1}^{r} get(u_i) \cup \text{msg}(P) \quad \text{if } H(\tilde{x}) \stackrel{\text{def}}{=} P$$

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

$$get(a) \stackrel{\text{def}}{=} \{a\} \qquad \text{(basic message)}$$
$$get(x) \stackrel{\text{def}}{=} \emptyset \qquad \text{(variable)}$$
$$get(\,\mathrm{F}^i(u_1, \dots, u_{k_i})\,) \stackrel{\text{def}}{=} \begin{cases} \bigcup_{j=1}^{k_i} get(u_j) \cup \{\,\mathrm{F}^i(u_1, \dots, u_{k_i})\,\} & \text{if } \mathrm{F}^i(u_1 \dots u_{k_i}) \in Msg \\ \bigcup_{j=1}^{k_i} get(u_j) & \text{otherwise.} \end{cases}$$

Now, everything is in place to formally define our notion of attacker. For simplicity, in the rest of the paper, given a set of nodes $\mathcal{N}$ and a node n, we will write $\mathcal{N} \setminus n$ for $\mathcal{N} \setminus \{n\}$, and $\mathcal{N} \cup n$ for $\mathcal{N} \cup \{n\}$.

Definition 3.2 (Attacker) *Let M be a network, with $\mathsf{nds}\,(M) = \{m_1, ..., m_k\}$. Let $\mathcal{A} = \{a_1, \dots, a_k\}$ be a set of attacking nodes for M. We define the set of* attackers *of M at $\mathcal{A}$ with initial knowledge $\phi_0 \subseteq Msg$ as:*

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

Remark 3.3 *By Proposition 2.10, the requirement $Q_i \in Prc_{\text{wt}}$ in the definition of $\mathbb{A}^{\phi_0}_{\mathcal{A}/M}$ guarantees that our attackers respects well-timedness and hence cannot prevent the passage of time.*

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 3.4 *Let $M = \prod_{i=1}^{k} m_i[P_i]^{\nu_i}$. Given a set $\mathcal{A} = \{a_1, \dots, a_k\}$ 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 of properties as follows.

Definition 3.5 (*tGNDC* for wireless networks) *Given a network M, an initial knowledge ϕ_0, a set $O \subseteq \mathsf{nds}(M)$ of nodes under observation and an abstraction $\rho(M)$, representing a security property for M, we write $M \in tGNDC^{\rho(M)}_{\phi_0,O}$ if and only if for some set $\mathcal{A}$ of attacking nodes for M and for every $A \in \mathbb{A}^{\phi_0}_{\mathcal{A}/M}$ it holds that*

$$M^{\mathcal{A}}_{O} \ \big|\ A \ \lesssim\ \rho(M) \ .$$

It should be noticed that when showing that a system M is $tGNDC^{\rho(M)}_{\phi_0,O}$, 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 [12]. 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. To this purpose we need a formalisation of computation. For $\Lambda{=}\alpha_1 \dots \alpha_n$, we write $\stackrel{\Lambda}{\Longrightarrow}$ to denote $\Rightarrow \xrightarrow{\alpha_1} \Rightarrow \dots \Rightarrow \xrightarrow{\alpha_n} \Rightarrow$. 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 3.6 (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 $M^{\mathcal{A}}_{\mathsf{nds}(M)} \ \big|\ A \ \stackrel{\Lambda}{\Longrightarrow}\ M' \ \big|\ A'$, 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 a 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 $\mathcal{A} = \{a_1, \dots, a_k\}$ 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 ϕ_j.

Definition 3.7 (Top attacker) *Let M be a network, with $\mathcal{P} = \mathsf{nds}(M) = \{m_1, ..., m_k\}$. Let $\mathcal{A} = \{a_1, \dots, a_k\}$ be a set of attacking nodes for M, and $\{\phi_j\}_{j\geq 0}$ a sequence of knowledge. We define:*

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

Basically, the top attacker $\textsc{Top}^{\phi_j}_{\mathcal{A}/\mathcal{P}}$ can perform the following transitions:

- $\textsc{Top}^{\phi_j}_{\mathcal{A}/\mathcal{P}} \xRightarrow{a_i!w\triangleright m_i} \textsc{Top}^{\phi_j}_{\mathcal{A}/\mathcal{P}}$, for every $i \in \{1, \dots, k\}$ and $w \in \mathcal{D}(\phi_j)$
- $\textsc{Top}^{\phi_j}_{\mathcal{A}/\mathcal{P}} \xrightarrow{\sigma} \textsc{Top}^{\phi_{j+1}}_{\mathcal{A}/\mathcal{P}}$.

In particular, from the 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 3.2, 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.

Remark 3.8 *Notice that the top attacker ignores message causality within a single time unit. Thus, it knows all messages in ϕ_j already at the beginning of the time slot j. Notice also that, at each time slot, the top attacker acquires all information that may be transmitted by the protocol at that time independently whether the information is really transmitted or not.*

Remark 3.9 *The top attacker does not satisfy well-timedness (see Definition 2.9), 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 first compositional property that involves the top attacker is the following (the symbol $\uplus$ denotes *disjoint union*).

Lemma 3.10 *Let $M = M_1 \mid M_2$ 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 and M_2, respectively. Let $O_1 \subseteq \mathsf{nds}(M_1)$ and $O_2 \subseteq \mathsf{nds}(M_2)$. Then*

$$(M_1 \mid M_2)^{\mathcal{A}_1 \uplus \mathcal{A}_2}_{O_1 \uplus O_2} \;\Big|\; \mathrm{Top}^{\phi_0}_{\mathcal{A}_1 \uplus \mathcal{A}_2/\mathsf{nds}(M)} \;\lesssim\; M_1{}^{\mathcal{A}_1}_{O_1} \;\Big|\; M_2{}^{\mathcal{A}_2}_{O_2} \;\Big|\; \mathrm{Top}^{\phi_0}_{\mathcal{A}_1/\mathsf{nds}(M_1)} \;\Big|\; \mathrm{Top}^{\phi_0}_{\mathcal{A}_2/\mathsf{nds}(M_2)} \ .$$

The following theorems say that (i) the top attacker $\mathrm{Top}^{\phi_0}_{\mathcal{A}/\mathcal{P}}$ is strong enough for checking *tGNDC*, and that (ii) the notion of the most powerful attacker can be employed to reason in a compositional manner.

Theorem 3.11 (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 $O \subseteq \mathsf{nds}(M) = \mathcal{P}$. Then $M^{\mathcal{A}}_{O} \;\big|\; \mathrm{Top}^{\phi_0}_{\mathcal{A}/\mathcal{P}} \lesssim N$ implies $M \in tGNDC^{N}_{\phi_0,O}$.*

The notion of the most powerful attacker is eventually employed to obtain the compositional property outlined by the following proposition.

Theorem 3.12 (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 $O_i \subseteq \mathsf{nds}(M_i) = \mathcal{P}_i$, for $1 \leq i \leq k$. Then, $(M_i)^{\mathcal{A}_i}_{O_i} \big| \mathrm{Top}^{\phi_0}_{\mathcal{A}_i/\mathcal{P}_i} \lesssim N_i$, for $1 \leq i \leq k$, implies $M \in tGNDC^{N_1|\ldots|N_k}_{\phi_0, O_1 \cup \ldots \cup O_k}$.*

Proof By Theorem 2.12 we have

$$(M_1)^{\mathcal{A}_1}_{O_1} \Big| \ldots \Big| (M_k)^{\mathcal{A}_k}_{O_k} \Big| \mathrm{Top}^{\phi_0}_{\mathcal{A}_1/\mathsf{nds}(M_1)} \Big| \ldots \Big| \mathrm{Top}^{\phi_0}_{\mathcal{A}_k/\mathsf{nds}(M_k)} \;\lesssim\; N_1 \Big| \ldots \Big| N_k \ .$$

By applying Lemma 3.10 and Theorem 2.12 we obtain

$$(M_1 \mid \ldots \mid M_k)^{\mathcal{A}_1 \uplus \ldots \uplus \mathcal{A}_k}_{O_1 \uplus \ldots \uplus O_k} \;\Big|\; \mathrm{Top}^{\phi_0}_{\mathcal{A}_1 \uplus \ldots \uplus \mathcal{A}_k/\mathsf{nds}(M_1|\ldots|M_k)} \;\lesssim\; N_1 \Big| \ldots \Big| N_k \ .$$

Thus, by an application of Theorem 3.11 we can derive $M \in tGNDC^{N_1|\ldots|N_k}_{\phi_0, O_1 \uplus \ldots \uplus O_k}$. □

3.1. Two timed security properties

As in [12], we formalise two useful timed properties for security protocols as instances of $tGNDC^{\rho}_{\phi_0,O}$, by suitably defining the abstraction function ρ. We will focus on the two following timed properties:

- A timed notion of integrity, called *timed integrity*, which guarantees that only fresh packets are authenticated.
- A timed notion of authentication, called *timed agreement*, according to which if agreement is reached between two parties then this must happen within a certain deadline, otherwise authentication does not hold.

More precisely, fixed a delay δ, a protocol is said to enjoy the timed integrity property if, whenever a packet p_i is authenticated during the i-th time interval, then this packet was sent at most $i-\delta$ time intervals before. For verification reasons, when expressing time integrity in the *tGNDC* scheme, we will introduce in the protocol under examination a special message auth_i which is emitted only when the packet p_i is authenticated.

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 m has initiated the protocol, apparently with n, at most δ time intervals before, and the two agents agreed on some set of data. When expressing time agreement in the *tGNDC* scheme, we introduce in the protocol under examination a special message hello_i, which is emitted by the initiator at the i-th run of the protocol, and a special message end_i, emitted by the responder, representing the completion of the protocol launched at run i.

4. A security analysis of μTESLA

The μTESLA protocol was designed by Perrig et al. [17] to provide authenticated broadcast from a base station (BS) towards all nodes of a wireless network. The protocol is based on a delayed disclosure of symmetric keys, and it requires the network to be *loosely time synchronised*. The protocol computes a MAC for every packet to be broadcast, by using different keys. The transmission time is split into *time intervals* of Δ_{int} time units each, and each key is tied to one of them. The keys belongs to a key chain $k_0, k_1, \ldots, k_n$ generated by BS by means of a public *one-way* function F. In order to generate this chain, BS randomly chooses the last key k_n and repeatedly applies F to compute all the other keys, whereby $k_i := F(k_{i+i})$, for $0 \leq i \leq n-1$. The key-chain mechanism together with the one-way function F, provides two major advantages: (i) a key k_i can be used to generate the beginning of the chain $k_0, \ldots, k_{i-1}$, by simply applying F as many time as necessary, but it cannot be used to generate any of the subsequent keys; (ii) any of the keys $k_0, \ldots, k_{i-1}$ can be used to authenticate k_i. Each node m_j is pre-loaded with a *master key* $k_{\text{BS}:m_j}$ for unicast communications with BS.

The μTESLA protocol is constituted by two main phases: *bootstrapping new receivers* and *authenticated broadcast*. The former establishes the node's initial setting in order to start receiving the authenticated packets, the latter describes the transmission of authenticated information.

In the first phase, when a new node m wishes to join the network it sends a request message to the base station BS containing its name and a nonce n_j, where j counts the number of bootstrapping requests:

$$m \longrightarrow \text{BS} : n_j \mid m \ .^3$$

The base station replies with a message of initialisation of the following form:

$$\text{BS} \longrightarrow m : \Delta_{\text{int}} \mid i \mid k_l \mid l \mid \text{mac}(k_{\text{BS}:m}\, , \, (n_j \mid \Delta_{\text{int}} \mid i \mid k_l \mid l))$$

where i is the current time interval of BS, k_l is a key in the key chain, and l, with $l < i$, represents the time interval in which k_l was employed for packet encryption. The secret key $k_{\text{BS}:m}$ is used to authenticate unicast messages; the nonce n_j allows the node m to verify the freshness of the reply coming from BS.

In the authenticated-broadcast phase, at each time interval i, one or more packets p_i are deployed by BS, each one containing the payload and the MAC calculated with the key k_i bound to the i-th time interval. Thus, at time interval i the BS broadcasts the authenticated message:

$$\text{BS} \longrightarrow * : p_i \mid \text{mac}(p_i, k_i) \ .$$

In the same time interval i, the key tied to the previous time interval $i - 1$ is disclosed to all receivers, so that they can authenticate all the previously received packets:

$$\text{BS} \longrightarrow * : k_{i-1} \ .$$

Loose time synchronisation on the key disclosure time prevents 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. In this phase, the nodes exploit the two main advantages of the key chain and the one-way function F: (i) the last received key k_i can be authenticated by means of F and the last authenticated key k_l; (ii) lost keys can be recovered by applying F to the last received key k_i. For instance, suppose that BS has sent packet p_1 (containing a MAC with key k_1) in the first time interval, packet p_2 in the second time interval and packet p_3 in the third one. If the key k_1 is correctly received by a node m while keys k_2 and k_3 get lost, then m can only authenticate the packet p_1 but not p_2 or p_3. However, if m gets the key k_4 then m can authenticate k_4 by using k_1, and it can also recover the lost keys k_2 and k_3 to authenticate p_2 and p_3, respectively.

Our security analysis of μTESLA focuses on the authenticated-broadcast phase which represents the core of the protocol.

Encoding in `aTCWS`. In Table 5 we provide an encoding of the authenticated-broadcast phase of μTESLA. Our encoding contains a few simplifications with respect to the original protocol. We assume that the duration of the time interval Δ_{int} is fixed and it is already known by the nodes. In our encoding, this time interval corresponds to two σ-actions. We assume that in each time interval i the sender broadcasts alternately only one packet p_i and the key k_{i-1} of the previous time interval. Thus, we assume a sequence $q_1, q_2, \ldots$ of payloads to be authenticated by using the corresponding keys $k_1, k_2, \ldots$ Moreover, we do not model the recovery of lost keys, hence the payload q_i can only be

[3] Here, the "|" symbol denotes message concatenation.

Table 5 μTESLA: authenticated-broadcast phase.

Sender:		
$S_i \stackrel{\text{def}}{=}$	$[q_i\ k_i \vdash_{mac} u_i]$	build MAC with payload and key
	$[u_i\ q_i \vdash_{pair} p_i]$	build packet with mac and payload
	$!\langle p_i\rangle.\sigma.$	broadcast packet, synchronise
	$!\langle k_{i-1}\rangle.\sigma.$	broadcast previous key, synchronise
	S_{i+1}	and go to next sending state
Receiver:		
$R(i,l,r,k_l) \stackrel{\text{def}}{=}$	$\lfloor ?(p).\sigma.P\langle i,l,p,r,k_l\rangle\rfloor$	receive a pkt, synchronise, go to P
	$Q\langle i,l,r,k_l\rangle$	if timeout go to Q
$P(i,l,p,r,k_l) \stackrel{\text{def}}{=}$	$\lfloor ?(k).T\langle i,l,p,r,k_l,k\rangle\rfloor$	receive a key k and move to T
	$R\langle i{+}1,l,p,k_l\rangle$	if timeout go to next receiving state
$T(i,l,p,r,k_l,k) \stackrel{\text{def}}{=}$	$[F^{i-1-l}(k) = k_l]$	authenticate key k with F and k_l
	$[r \vdash_{\text{fst}} u]$	extract MAC from previous pkt r
	$[r \vdash_{\text{snd}} q]$	extract payload from r
	$[q\ k \vdash_{mac} u']$	build MAC for r with key k
	$[u = u']$	check MACs to authenticate r
	$\sigma.Z\langle i{+}1, i{-}1, p, r, k\rangle;$	
	$\sigma.R\langle i{+}1, i{-}1, p, k\rangle;$	
	$\sigma.R\langle i{+}1, i{-}1, p, k\rangle;$	
	$\sigma.R\langle i{+}1, l, p, k_l\rangle$	
$Z(i,l,p,r,k_l) \stackrel{\text{def}}{=}$	$R\langle i,l,p,k_l\rangle$	authenticated-broadcast succeeded
$Q(i,l,r,k_l) \stackrel{\text{def}}{=}$	$\lfloor ?(k).T\langle i,l,r,r,k_l,k\rangle\rfloor$	receive a key, synchronise, and
	$R\langle i{+}1,l,r,k_l\rangle$	go to next receiving state

authenticated by receiving the key k_i. This simplification yields a easier to read model which can be generalised to fulfil the original requirements of the protocol.

The encoding essentially defines two kind of processes: the senders S_i, and the receivers $R(i,l,r,k_l)$, where i is the current time interval, r is the last received packet, l is the time interval when the last key k_l was authenticated. Since we bind one packet to one key, i also refers to the index number of packets.

The authenticated-broadcast phase of μTESLA can be represented as follows:

$$\mu\text{TESLA} \stackrel{\text{def}}{=} \textsc{bs}[\sigma.S_1]^{\nu_{\textsc{bs}}} \ \big|\ m_1[\sigma.R\langle 1,-1,\perp,k_{\textsc{bs}}\rangle]^{\nu_{m_1}} \ \big|\ \dots \ \big|\ m_h[\sigma.R\langle 1,-1,\perp,k_{\textsc{bs}}\rangle]^{\nu_{m_h}}$$

where $m \in \nu_{\textsc{bs}}$ and $\textsc{bs} \in \nu_m$, for every $m \in \{m_1,\dots,m_h\}$. We use $\perp$ because at the beginning there is no packet to authenticate. We write $k_{\textsc{bs}}$ to denote the key transmitted by the base station $\textsc{bs}$ and authenticated at the node's site during the bootstrapping phase. Notice that $k_{\textsc{bs}}$ is associated to the time interval -1.

4.1. Timed integrity

In this section, we show that the authenticated-broadcast phase of μTESLA *enjoys timed integrity*. In particular, we prove that receivers authenticate only packets that have been sent $2\Delta_{\text{int}}$ time units before (that is, four σ-actions before) in the correct order, even in the presence of an intruder. The crucial point is that even if an attacker acquires the shared keys then it is "too late" to break integrity, i.e., to authenticate packets which are more than $2\Delta_{\text{int}}$ time units old.

We signal authentication of a packet r by broadcasting a special packet pair(auth, r). Thus, we replace the process $R(i,l,r,k_l)$ of Table 5 with $R'(i,l,r,k_l)$, where the process $Z(i,l,p,r,k_l)$ is replaced by

$$Z'(i,l,p,r,k_l) \stackrel{\text{def}}{=} [\mathsf{auth}\ r \vdash_{pair} t]!\langle t\rangle.R'\langle i,l,p,k_l\rangle\ .$$

The formalisation of the authenticated-broadcast phase for μTESLA becomes the following:

$$\mu\text{TESLA}' \stackrel{\text{def}}{=} \text{BS}[\sigma.S_1]^{\nu_{\text{BS}}} \;\big|\; m_1[\sigma.R'\langle 1,-1,\perp,k_{\text{BS}}\rangle]^{\nu_{m_1}} \;\big|\; \dots \;\big|\; m_h[\sigma.R'\langle 1,-1,\perp,k_{\text{BS}}\rangle]^{\nu_{m_h}} \;.$$

We define the *timed integrity* property as the following abstraction of the protocol μTESLA′:

$$\rho(\mu\text{TESLA}') \stackrel{\text{def}}{=} \text{BS}[\sigma.S_1]^{obs} \;\big|\; m_1[\sigma.\hat{R}_1]^{obs} \;\big|\; \dots \;\big|\; m_h[\sigma.\hat{R}_1]^{obs}$$

where S_1 is the process defined in Table 5, while $\hat{R}_i \stackrel{\text{def}}{=} \sigma.\lfloor\tau.\sigma.!\langle \mathsf{auth}_{i-1}\rangle.\hat{R}_{i+1}\rfloor\hat{R}_{i+1}$. The node *obs* is the observing node introduced in Section 3. Here, we abstract on receivers' behaviour: At time interval i+2 they may signal the authentication of the packet $p_i = \text{pair}(\text{mac}(k_i, q_i), q_i)$ by sending the special packet $\mathsf{auth}_i = \text{pair}(\mathsf{auth}, p_i)$.

The abstraction $\rho(\mu\text{TESLA}')$ is a faithful representation of the timed integrity property for the authenticated-broadcast phase of μTESLA.

Proposition 4.1 *Whenever* $\rho(\mu TESLA') \stackrel{\Lambda}{\Longrightarrow} \xrightarrow{!p_i \triangleright obs} \stackrel{\Omega}{\Longrightarrow} \xrightarrow{!\mathsf{auth}_i \triangleright obs} M$ *then* $\#^{\sigma}(\Omega) = 4$.

In order to show that μTESLA′ satisfies timed integrity, we will prove that

$$\mu\text{TESLA}' \in tGNDC^{\rho(\mu\text{TESLA}')}_{\phi_0,\{\text{BS},m_1,\dots,m_k\}}$$

for some appropriate ϕ_0. Notice that μTESLA′ is time-dependent stable with respect to the following sequence of knowledge:

$$\begin{array}{lcll} \phi_0 & \stackrel{\text{def}}{=} & \emptyset & \\ \phi_1 & \stackrel{\text{def}}{=} & \{p_1\} & \\ & \vdots & & \\ \phi_i & \stackrel{\text{def}}{=} & \phi_{i-1} \cup \{k_{j-1}\} & \text{if } j > 0 \text{ and } i = 2j \\ \phi_i & \stackrel{\text{def}}{=} & \phi_{i-1} \cup \{p_{j+1}, \mathsf{auth}_{j-1}\} & \text{if } j > 0 \text{ and } i = 2j+1 \;. \end{array} \tag{1}$$

We fix an attacking node a_j for each m_j, with $1 \le j \le h$, and an attacking node b for BS. By applying the compositional criterion of Theorem 3.12, it suffices to prove a simpler integrity result for each node in isolation composed with its corresponding top attacker.

Lemma 4.2 *Given an attacking node b for* BS *and the attacking nodes a_j for m_j, with $1 \le j \le h$, and fixed the sequence of knowledge $\{\phi_i\}_{i\ge 0}$ as in* (1)*, then the encoding in Table* 5 *satisfies the following:*

1. $\text{BS}[\sigma.S_1]^{\{b,obs\}} \;\big|\; \text{Top}^{\phi_0}_{b/\text{BS}} \lesssim \text{BS}[\sigma.S_1]^{obs}$
2. $m_j[\sigma.R'\langle 1,-1,\perp,\bar{k}\rangle]^{\{a_j,obs\}} \;\big|\; \text{Top}^{\phi_0}_{a_j/m_j} \lesssim m_j[\sigma.\hat{R}_1]^{obs}$*, for* $1 \le j \le h$.

Theorem 4.3 (μTESLA Timed integrity) *The protocol μTESLA′ satisfies timed integrity:*

$$\mu TESLA' \in tGNDC^{\rho(\mu TESLA')}_{\phi_0,\{\text{BS},m_1,\dots,m_k\}} \;.$$

Proof By applying Lemma 4.2 and Theorem 3.12. □

4.2. Timed agreement

The timed agreement property for the authenticated-broadcast phase μTESLA requires that when the receiver m_j completes the protocol, apparently with the initiator BS, then BS has initiated the protocol, apparently with m_j, at most $2\Delta_{\text{int}}$ time intervals before, and the two parties agree on the sent data. In other words, the packet p_i is authenticated by m_j exactly $2\Delta_{\text{int}}$ time units after it has been sent by BS. This says that any formulation of timed agreement for μTESLA would actually coincide with timed integrity. Thus, Proposition 4.1 demonstrates that $\rho(\mu\text{TESLA}')$ is also a faithful abstraction of timed agreement. As a consequence, Theorem 4.3 also says that μTESLA *satisfies timed agreement*.

5. A Security Analysis of LEAP+

The LEAP+ protocol [3] provides a keying mechanism to establish authenticated communications. The protocol is designed to establish four types of keys: an *individual key*, shared between a base station and a node, a *single-hop pair-wise key*, shared between two sensor nodes, a *cluster key*, shared between a node and all its neighbourhood, a *group key*, shared between a base station and all sensor nodes of the network.

In this section, we focus on the *single-hop pairwise key* mechanism as it is underlying to all other keying methods. This mechanism is aimed at establishing a pair-wise key between a sensor node and a neighbours in Δ_{leap} time units. In order to do that, LEAP+ exploits two peculiarities of sensor nodes: (i) the set of neighbours of a node is relatively static, and (ii) a sensor node that is being added to the network will discover most of its neighbours at the time of its initial deployment.

The single-hop pairwise shared key mechanism of LEAP+ consists of three phases.

Key pre-distribution. A network controller fixes an initial key k_{in} and a computational efficient pseudo-random function prf(). Both k_{in} and prf() are pre-loaded in each node, before deployment. Then, each node r derives its *master key*: $k_r := \text{prf}(k_{\text{in}}, r)$.

Neighbour discovery. As soon as a node m is scattered in the network area it tries to discover its neighbours by broadcasting a hello packet that contains its identity, m, and a freshly created nonce n_i, where i counts the number of attempts:

$$m \longrightarrow * : m \mid n_i \ .$$

Then each neighbour r replies with an ack message which includes its identity r, the corresponding MAC calculated by using r's master key k_r, to guarantee authenticity, and the nonce n_i, to guarantee freshness. Specifically:

$$r \longrightarrow m : r \mid \text{mac}(k_r, (r \mid n_i)) \ .$$

Pairwise Key Establishment. When m receives the packet q from r, it tries to authenticate it by using the last created nonce n_i and r's master key $k_r = \text{prf}(k_{\text{in}}, r)$. Notice that m can calculate k_r as k_{in} and prf have been pre-loaded in m, and r is contained in q. If the authentication succeeds, then both nodes proceed in calculating the pairwise key $k_{m:r} := \text{prf}(k_r, m)$. Any other message between m and r will be authenticated by using the pairwise key $k_{m:r}$. If m does not get an authenticated packet from the responder in due time, it sends a new hello packet with a fresh nonce.

Encoding in `aTCWS`. In Table 6, we provide an encoding of the single-hop pairwise shared key mechanism of LEAP+. For the sake of clarity, we assume that Δ_{leap} consists of two time slots, i.e. it takes two σ-actions. To yield an easier to read model, we consider only two nodes and we define

$$\text{LEAP+} \stackrel{\text{def}}{=} m[\sigma.S_1]^{\nu_m} \ \Big| \ r[\sigma.R]^{\nu_r}$$

where m is the initiator, r is the responder, with $m \in \nu_r$ and $r \in \nu_m$. Moreover, we assume that r has already computed its master key $k_r := \text{prf}(k_{\text{in}}, r)$. This simple model does not lose in generality with respect to the multiple nodes case.

5.1. Timed Agreement

The timed agreement property for LEAP+ requires that the responder r successfully completes the protocol initiated by m, with the broadcasting of a hello packet, in at most Δ_{leap} time units (i.e. two σ-actions). We will show that LEAP+ *does not satisfy* the timed agreement property. Intuitively, due to the presence of the attacker, r may wrongly believe it has established a pairwise key with m, whereas m will indefinitely send new hello packets. In some respect, this may be viewed as a kind of *denial-of-service attack*.

For our analysis, in order to make observable the completion of the protocol, we define LEAP′+ by replacing in LEAP+ the process R of Table 6 with the process R' defined as the same as R except for process R^6 which is replaced by

$$R^{6\prime} \stackrel{\text{def}}{=} \sigma.[\mathsf{end}\ n \vdash_{pair} e]!\langle e\rangle.\mathsf{nil} \ .$$

Table 6 LEAP+ specification

Sender at node m:	
$S_i \stackrel{\text{def}}{=} [n_{i-1}\ m \vdash_{prf}\ n_i]$	build a random nonce n_i
$[m\ n_i \vdash_{pair}\ t]$	build a pair t with m and the nonce n_i
$[\mathsf{hello}\ t \vdash_{pair}\ p]$	build hello packet using the pair t
$!\langle p\rangle.\sigma.P$	broadcast hello, synchronise and move to P
$P \stackrel{\text{def}}{=} \lfloor ?(q).P^1 \rfloor S_{i+1}$	wait for response from neighbours
$P^1 \stackrel{\text{def}}{=} [q \vdash_{fst}\ r]P^2; \sigma.S_{i+1}$	extract node name r from packet q,
$P^2 \stackrel{\text{def}}{=} [q \vdash_{\text{snd}}\ h]$	extract MAC h from packet q
$[r\ n_i \vdash_{pair}\ t']$	build a pair t' with r and current nonce n_i
$[k_{\text{in}}\ r \vdash_{prf}\ k_r]$	calculate r's master key k_r
$[k_r\ t' \vdash_{mac}\ h']$	calculate MAC h' with k_r and t'
$[h' = h]P^3; \sigma.S_{i+1}$	if it matches with the received one go to P^3, otherwise go to next time unit and restart
$P^3 \stackrel{\text{def}}{=} [k_r\ m \vdash_{prf}\ k_{m:r}]P^4$	calculate the pairwise key $k_{m:r}$
$P^4 \stackrel{\text{def}}{=} \sigma.\mathsf{nil}$	synchronise and conclude key establishment
Receiver at node r:	
$R \stackrel{\text{def}}{=} \lfloor ?(p).R^1 \rfloor \sigma.R$	wait for incoming hello packets
$R^1 \stackrel{\text{def}}{=} [p \vdash_{fst}\ p_1]R^2; \sigma.\sigma.R$	extract the first component
$R^2 \stackrel{\text{def}}{=} [p \vdash_{\text{snd}}\ p_2]$	extract the second component
$[p_1 = \mathsf{hello}]R^3; \sigma.\sigma.R$	check if p is a hello packet
$R^3 \stackrel{\text{def}}{=} [p_2 \vdash_{fst}\ m]R^4; \sigma.\sigma.R$	extract the sender name m
$R^4 \stackrel{\text{def}}{=} [p_2 \vdash_{\text{snd}}\ n]$	extract the nonce n
$[r\ n \vdash_{pair}\ t]$	build a pair t with n and r
$[k_r\ t \vdash_{mac}\ h]$	calculate MAC h on t with r's master key k_r
$[r\ h \vdash_{pair}\ q]$	build packet q with node name r and MAC h
$\sigma.!\langle q\rangle.R^5$	synchronise, broadcast q and go to R^5
$R^5 \stackrel{\text{def}}{=} [k_r\ m \vdash_{prf}\ k_{m:r}]R^6$	calculate pairwise key $k_{m:r}$
$R^6 \stackrel{\text{def}}{=} \sigma.\mathsf{nil}$	synchronise and conclude key establishment

We use the following abbreviations: $\mathsf{hello}_i \stackrel{\text{def}}{=} \mathrm{pair}(\mathsf{hello}, \mathrm{pair}(m, n_i))$ and $\mathsf{end}_i \stackrel{\text{def}}{=} \mathrm{pair}(\mathsf{end}, n_i)$.

The timed agreement property of LEAP+ is defined by the following abstraction:

$$\rho_{agr}(\text{LEAP}'+) \stackrel{\text{def}}{=} m[\sigma.\bar{S}_1]^{obs} \mid r[\sigma.\bar{R}_1]^{obs}$$

where $\bar{S}_i \stackrel{\text{def}}{=} !\langle \mathsf{hello}_i\rangle.\sigma\lfloor\tau.\sigma.\mathsf{nil}\rfloor\bar{S}_{i+1}$ and $\bar{R}_i \stackrel{\text{def}}{=} \lfloor\tau.\sigma!\langle q_i\rangle.\sigma.!\langle \mathsf{end}_i\rangle.\mathsf{nil}\rfloor\sigma.\bar{R}_{i+1}$, with $q_i = \mathrm{pair}(r, \mathrm{mac}(k_r, \mathrm{pair}(r, n_i)))$, as defined in Table 6.

The following statement says that the abstraction $\rho_{agr}(\text{LEAP}'+)$ expresses correctly the timed agreement property for LEAP+.

Proposition 5.1 *Whenever* $\rho_{agr}(\text{LEAP}'+) \stackrel{\Lambda}{\Longrightarrow} \xrightarrow{!\mathsf{hello}_i \triangleright obs} \stackrel{\Omega}{\Longrightarrow} \xrightarrow{!\mathsf{end}_i \triangleright obs}$ *then* $\#^{\sigma}(\Omega) = 2$.

Now, in order to prove timed agreement for LEAP+ we should show that

$$\text{LEAP}'+ \in tGNDC^{\rho_{agr}(\text{LEAP}'+)}_{\phi_0,\{m,r\}}$$

Table 7 Replay attack to LEAP+.

$m \longrightarrow * : \mathsf{hello}_1$	m starts the protocol, but hello_1 is grasped by a and missed by r
$\xrightarrow{\sigma}$	the system moves to the next time slot
$a \longrightarrow b : \mathsf{hello}_1$	a sends hello_1 to b
$\xrightarrow{\sigma}$	the system moves to the next time slot
$b \longrightarrow r : \mathsf{hello}_1$	b *replays* hello_1 to r
$m \longrightarrow * : \mathsf{hello}_2$	m broadcasts hello_2 (containing a fresh nonce n_2), which gets lost
$\xrightarrow{\sigma}$	the system moves to the next time slot
$r \longrightarrow m : q_1$	r replies by sending q_1 (which is discarded by m)
$\xrightarrow{\sigma}$	the system moves to the next time slot
$r \longrightarrow * : \mathsf{end}_1$	r signals the end of the protocol

for some appropriate ϕ_0. This would imply that all traces of the system composed by LEAP′+ in parallel with an attacker can be mimicked by ρ_{agr}(LEAP′+).

However, this is not the case, as stated by the following theorem.

Theorem 5.2 (Replay Attack to LEAP+)

$$\text{LEAP}'{+} \notin tGNDC^{\rho_{agr}(\text{LEAP}'+)}_{\emptyset,\{m,r\}} \quad .$$

Proof We define an attacker that delays agreement. Let us define the set of attacking nodes $\mathcal{A} = \{a, b\}$ for nds (LEAP′+). Let us fix the initial knowledge $\phi_0 = \emptyset$, so to deal with the most general situation. We set $\nu_a = \{m, b\}$ and $\nu_b = \{r, a\}$, and we assume all the nodes in nds (LEAP′+) are observable, thus $\nu_m = \{r, a, obs\}$ and $\nu_r = \{m, b, obs\}$. We give an intuition of the replay attack in Table 7. Basically, the attacker delays the reception of the packet p_1 at m which cannot complete the protocol within two time slots, but only after four time slots, thus breaking agreement. Formally, we define the attacker $A \in \mathbb{A}^{\phi_0}_{\mathcal{A}/\{m,r\}}$ as follows:

$$A = a[\sigma.X]^{\nu_a} \mid b[\sigma^2.X]^{\nu_b}$$

where $X = \lfloor ?(x).\sigma.!\langle x\rangle.\mathsf{nil}\rfloor \mathsf{nil}$. Now, we consider the system

$$(\text{LEAP}'{+})^{\mathcal{A}} \,\big|\, A \;=\; m[\sigma.S_1]^{\nu_m} \,\big|\, r[\sigma.R']^{\nu_r} \,\big|\, A$$

and we find that it admits the following execution trace

$$\sigma \,.\, !\mathsf{hello}_1 \triangleright obs \,.\, \sigma \,.\, \tau \,.\, \sigma \,.\, \tau \,.\, !\mathsf{hello}_2 \triangleright obs \,.\, \sigma \,.\, !q_1 \triangleright obs \,.\, \sigma \,.\, !\mathsf{end}_1 \triangleright obs$$

where the packet hello_1 and the corresponding packet end_1 are divided by four σ-actions (we report the corresponding computation in the Appendix). Proposition 5.1 says that this trace cannot be mimicked by the abstraction $\rho_{agr}(\mu\text{TESLA}'_{boot})$. As a consequence, the timed agreement property for LEAP+ does not hold. □

5.2. *Timed integrity*

The timed integrity property for LEAP+ says that hello messages and authentication messages with the same nonce must differ for at most Δ_{leap} time units. We show that LEAP+ *satisfies* the timed integrity property. For doing that, we slightly modify the specification of LEAP+ to make observable key authentication. We define

$$\text{LEAP}''{+} \stackrel{\text{def}}{=} m[\sigma.S''_1]^{\nu_m} \,\big|\, r[\sigma.R]^{\nu_r}$$

where the process S''_i is the same as process S_i of Table 6, except for process P^4 which is replaced by

$$P^{4''} \stackrel{\text{def}}{=} \sigma.[\mathsf{auth}\ t \vdash_{pair} a]!\langle a\rangle.\mathsf{nil} \quad .$$

For simplicity, we use the following abbreviation: $\mathsf{auth}_i = \mathrm{pair}(\mathsf{auth}, \mathrm{pair}(m, n_i))$.

In order to formally represent the timed integrity property, we define the following abstraction of the protocol:

$$\rho_{int}(\mathrm{LEAP}''\text{+}) \stackrel{\text{def}}{=} m[\sigma.\hat{S}_1]^{obs} \;\big|\; r[Tick]^{\emptyset}$$

where $\hat{S}_i \stackrel{\text{def}}{=} !\langle \mathsf{hello}_i\rangle.\sigma.\lfloor \tau.\sigma.!\langle \mathsf{auth}_i\rangle.\mathsf{nil}\rfloor \hat{S}_{i+1}$ and $Tick \stackrel{\text{def}}{=} \sigma.Tick$.

By construction, $\rho_{int}(\mathrm{LEAP}''\text{+})$ is a faithful representation of timed integrity for LEAP+ (we recall that in our encoding Δ_{leap} corresponds to two σ-actions).

Proposition 5.3 *Whenever* $\rho_{int}(\mathrm{LEAP}''\text{+}) \xRightarrow{\Lambda} \xrightarrow{!hello_i \triangleright obs} \xRightarrow{\Omega} \xrightarrow{!auth_i \triangleright obs} M$ *then* $\#^{\sigma}(\Omega) = 2$.

Now, we notice that LEAP″+ is time-dependent stable with respect to the sequence of knowledge $\{\phi_i\}_{i\geq 0}$, defined as follows:

$$\begin{array}{lcll} \phi_0 & \stackrel{\text{def}}{=} & \emptyset & \\ \phi_1 & \stackrel{\text{def}}{=} & \{\mathsf{hello}_1\} & \\ & \vdots & & \\ \phi_i & \stackrel{\text{def}}{=} & \phi_{i-1} \cup \{mac(k_r, \mathrm{pair}(r, n_j))\} & \text{if } j > 0 \text{ and } i = 2j \\ \phi_i & \stackrel{\text{def}}{=} & \phi_{i-1} \cup \{\mathsf{hello}_{j+1}, \mathsf{auth}_j\} & \text{if } j > 0 \text{ and } i = 2j+1 \ . \end{array} \tag{2}$$

Now, we pick two attacking nodes a and b, for m and r, respectively, and we focus on the observation of node m as it signals both the beginning and the end of the authentication protocol. Again, by applying Theorem 3.12 it suffices to prove a simpler result for each node in isolation composed with its corresponding top attacker.

Lemma 5.4 *Given two attacking nodes a and b, for m and r respectively, and fixed the sequence of knowledge* $\{\phi_i\}_{i\geq 0}$ *as in* (2)*, then*

1. $m[\sigma.S''_1]^{\{a,obs\}} \;\big|\; \mathrm{Top}^{\phi_0}_{a/m} \;\lesssim\; m[\sigma.\hat{S}_1]^{obs}$
2. $r[\sigma.R]^{\{b\}} \;\big|\; \mathrm{Top}^{\phi_0}_{b/r} \;\lesssim\; r[Tick]^{\emptyset}$.

Theorem 5.5 (LEAP+ Timed integrity) LEAP″+ *satisfies the timed integrity property:*

$$\mathrm{LEAP}''\text{+} \in tGNDC^{\rho_{int}(\mathrm{LEAP}''+)}_{\phi_0,\{m\}} \ .$$

Proof By applying Lemma 5.4 and Theorem 3.12. □

6. A security analysis of LiSP

In order to achieve a good trade-off between resource limitations and network security, Park et al. [2] have proposed a *Lightweight Security Protocol* (LiSP) for WSNs. LiSP provides (i) an efficient *key renewal* mechanism which avoids key retransmission, (ii) authentication for each key-disclosure, and (iii) the possibility of both recovering and detecting lost keys.

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

$$\textsc{ks} \longrightarrow m_j \ : \ \mathrm{enc}(k_{\textsc{ks}:m_j},\ (s \mid k_{s+i} \mid \Delta_{\text{refresh}})) \ \mid \ \mathrm{hash}(s \mid k_{s+i} \mid \Delta_{\text{refresh}}) \ .$$

The operator $\mathrm{enc}(k, p)$ represents the encryption of p by using the key of k, while $\mathrm{hash}(p)$ 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 $\mathsf{InitKey}$ packet, it computes the sequence of keys $k_{s+i-1}, k_{s+i-2}, \dots, k_i$ 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 $i \leq n$, $\textsc{ks}$ employs the active encryption key k_i to encode the key k_{s+i}. The resulting packet is broadcast as an $\mathsf{UpdateKey}$ packet:

$$\textsc{ks} \longrightarrow * \ : \ \mathrm{enc}(k_i,\ k_{s+i}) \ .$$

When a node receives an $\mathsf{UpdateKey}$ 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 $\mathsf{UpdateKey}$ 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 $\mathsf{RequestKey}$ packet in order to request the current configuration:

$$m_j \longrightarrow \textsc{ks} \ : \ \mathsf{RequestKey} \mid m_j \ .$$

Upon reception, node $\textsc{ks}$ performs authentication of m_j and, if successful, it sends the current configuration via an $\mathsf{InitKey}$ packet.

Encoding. In Table 8, 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 $k_0, \dots, k_n$ have already been computed by $\textsc{ks}$, (ii) both the buffer size s and the refresh interval Δ_{refresh} are known by each node. Thus, the $\mathsf{InitKey}$ packet can be simplified as follows:

$$\textsc{ks} \longrightarrow m_j \ : \ \mathrm{enc}(k_{\textsc{ks}:m_j},\ k_{s+i}) \ \mid \ \mathrm{hash}(k_{s+i}) \ .$$

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:

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

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

Table 8 The LiSP protocol

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

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 $R(k_{\mathrm{c}}, k_{\mathrm{L}}, l)$, 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, the LiSP protocol, in its initial configuration, can be represented as:

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

where for each node m_j, with $j \in J$, $m_j \in \nu_{\textsc{kd}} \cap \nu_{\textsc{kl}}$ and $\{\textsc{kd}, \textsc{kl}\} \subseteq \nu_{m_j}$.

Timed integrity. 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 will show that, due to the interference of an attacker, key authentication may take longer than Δ_{refresh} time units. In order to exhibit the attack 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 8 with the process

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

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

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

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

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

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 8;
- $\hat{L}_i \stackrel{\text{def}}{=} \lfloor \tau.\sigma.!\langle q_{i+1}\rangle.\sigma.\hat{L}_{i+1}\rfloor \sigma.\hat{L}_{i+1}$, and $q_i = \text{pair}(\mathsf{InitKey}\ r_i)$ with $r_i = \text{pair}(\text{enc}(k_{\textsc{ks}:m}, k_{s+i}), \text{hash}(k_{s+i}))$ as defined as in Table 8.

It is easy to see that $\rho(\text{LiSP}')$ is a correct abstraction of the timed integrity property of 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 6.1 $\rho(\text{LiSP}') \stackrel{\Lambda}{\Longrightarrow} \xrightarrow{!q_i \triangleright obs} \stackrel{\Omega}{\Longrightarrow} \xrightarrow{!\mathsf{auth}_i \triangleright obs}$ *implies* $\#^{\sigma}(\Omega) = 2$.

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

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

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

Theorem 6.2 (Replay attack to LiSP)

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

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 $\nu_b = \{\textsc{kl}, a\}$, and we assume that $O = \{\textsc{kl}, m\}$. We give an intuition of the replay attack in Table 9. Basically, an attacker may prevent the node m to receive the $\mathsf{InitKey}$ packet within Δ_{refresh} time units. As a consequence, m may 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 $A \in \mathbb{A}^{\phi_0}_{\mathcal{A}/\{\textsc{kl}, m\}}$ as $A = a[\sigma^3.X]^{\nu_a} \ \big| \ b[\sigma^2.X]^{\nu_b}$ where $X = \lfloor ?(x).\sigma.!\langle x\rangle.\mathsf{nil}\rfloor \mathsf{nil}$. We then consider the system $(\text{LiSP}')^{\mathcal{A}}_{O} \mid A$ 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 6.1, this trace cannot be matched by $\rho(\text{LiSP}')$. So, $(\text{LiSP}')^{\mathcal{A}}_{O} \ \big| \ A \ \not\lesssim \ \rho(\text{LiSP}')$. □

Table 9 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
$b \longrightarrow a : q_1$	b sends q_1 to a
$m \longrightarrow \text{KL} : r$	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
$m \longrightarrow * : \mathsf{auth}_1$	m authenticates q_1 and signals the end of the protocol

6.1. *Timed agreement*

The timed agreement property for LiSP requires that when a sensor node m completes the protocol, apparently with the initiator KL, then KL has initiated the protocol, apparently with m, at most Δ_{refresh} time units before, and the two parties agree on the transmitted data. In other words: the packet q_i must be received and authenticated by m exactly Δ_{refresh} time units after it has been sent by KL. This suggests that any formulation of timed agreement for LiSP would actually coincide with timed integrity. Thus, Proposition 6.1 demonstrates that $\rho(\text{LiSP}')$ is also a faithful abstraction of timed agreement. As a consequence, Theorem 6.2 also says that LiSP *does not satisfies timed agreement*.

7. Conclusions, Related and Future Work

We have proposed a timed broadcasting calculus, called `aTCWS`, to formalise and verify real-world key management protocols for WSNs. Our calculus comes with a well-defined operational semantics and a simulation-based behavioural semantics. Then, we have revised Gorrieri and Martinelli's *tGNDC* framework [12] in such a way that it can be applied to WSNs. We have used *tGNDC* to express two timed properties of wireless security protocols: timed integrity and timed agreement. A nice aspect of expressing a security property as a *tGNDC*-property is that when it does not hold then it is possible to exhibit an attack which invalidate the property. In order to prove *tGNDC*-properties in an effective manner, we have provided a compositional proof technique based on the notion of the most powerful attacker. Notice that, as pointed out in Remarks 3.8 and 3.9, the top attacker used in our proof technique is slightly more powerful than any process in `aTCWS`. As a consequence, our proof technique for *tGNDC* is sound but not complete. Nevertheless, when a protocol can be attacked by the top attacker there are good chances that the same attack can be perpetrated by a real attacker.

We have provided formal specifications in `aTCWS` of three well-known key management protocols for WSNs: μTESLA [1], LEAP+ [3] and LiSP [2]. Our specifications meet the requirements of Proposition 2.10, thus they all satisfy well-timedness. Then, we have formally proved that the single-hop pairwise shared key mechanism of LEAP+ enjoys timed integrity, and that the authenticated-broadcast phase of μTESLA enjoys both timed integrity and timed agreement. We have provided two different *replay attacks* to show that the single-hop pairwise shared key mechanism of LEAP+ do not enjoy timed agreement, and that LiSP satisfies neither timed integrity nor timed agreement. The kind of attack occurring in LiSP cannot be repeated in LEAP+ or μTESLA as these two protocols assume the presence of nonces which guarantee message freshness. We could use the same precaution in LiSP, by adding a nonce in all RequestKey and InitKey packets [18]. However, as seen for LEAP+ the addition of nonces in LiSP would fix timed integrity but not timed agreement.

The present work is the continuation and generalisation of [19], where a slight variant of the calculus was introduced, and an early security analysis for the authenticated-broadcast phase of μTESLA and the single-hop pairwise shared key mechanism of LEAP+ was performed. In [18] the calculus `aTCWS` has been used to analyse the LiSP protocol. The design of our calculus is strongly inspired by `tCryptoSPA` [12], a timed "cryptographic" variant of Milner's CCS [20], 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. The *tGNDC* schema for `tCryptoSPA`, has already been used by Gorrieri et al. [13] to study a number security protocols, for both wired and wireless networks. In particular, they studied the authenticated-broadcast phase of μTESLA, proving timed integrity. The formalisation for μTESLA we have proposed here is much less involved than the one of [13] thanks to the specific features of our calculus for broadcast communications.

Several process calculi for wireless systems have been recently proposed. Mezzetti and Sangiorgi [21] have introduced a calculus to describe interferences in wireless systems. Nanz and Hankin [22] have proposed a calculus for mobile ad hoc networks for specification and security analysis of communication protocols. They provide a decision procedure to check security against fixed intruders known in advance. Merro [23] has proposed a behavioural theory for mobile ad hoc networks. Godskesen [24] has proposed a calculus for mobile ad hoc networks with a formalisation of an attack on the cryptographic routing protocol ARAN. Singh et al. [25] have proposed the ω-calculus for modelling the AODV routing protocol. Ghassemi et al. [26, 27] have proposed a process algebra, provided with model checking and equational reasoning, which models topology changes implicitly in the semantics. Merro and Sibilio [28] have proposed a timed calculus for wireless systems focusing on the notion of communication collision. Godskesen and Nanz [29] have proposed a simple timed calculus for wireless systems to express a wide range of mobility models. Gallina and Rossi [30] have proposed a calculus for the analysis of energy-aware communications in mobile ad hoc networks. Song and Godskesen [31] have proposed the first probabilistic un-timed calculus for mobile wireless systems in which connection probabilities may change due to node mobility. Kouzapas and Philippou [32] have proposed a process calculus for dynamic networks which contains features for broadcasting at multiple transmission ranges and for viewing networks at different levels of abstraction.

Recently, Arnaud et al. [33] have proposed a calculus for modelling and reasoning about security protocols, including secure routing protocols, for a bounded number of sessions. They provide two NPTIME decision procedures for analysing routing protocols for any network topology, and apply their framework to analyse the protocol SRP [34] applied to DSR [35].

The AVISPA model checker [36] has been used in [37] for an analysis of TinySec [38], LEAP [39], and TinyPK [40], three wireless sensor network security protocols, and in [41] for an analysis of the Sensor Network Encryption Protocol SNEP [1]. In particular, in [37] the authors considered the communication between immediate neighbour nodes which use the pairwise shared key already established by LEAP. In this case AVISPA found a man-in-the-middle attack where the intruder may play at the same time the role of two nodes in order to obtain real information from one of them, thus loosing confidentiality.

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 [42], and evolutions of LEAP+, such as R-LEAP+ [43] and LEAP++ [44].

Acknowledgements. The referees have provided several useful comments.

References

[1] A. Perrig, R. Szewczyk, J. D. Tygar, V. Wen, D. Culler, SPINS: Security Protocols for Sensor Networks, Wireless Networks 8 (5) (2002) 521–534.

[2] T. Park, K. G. Shin, LiSP: A Lightweight Security Protocol for Wireless Sensor Networks, ACM Transactions on Embedded Computing Systems (TECS) 3 (3) (2004) 634–660.

[3] S. Zhu, S. Setia, S. Jajodia, LEAP+: Efficient Security Mechanisms For Large-Scale Distributed Sensor Networks, ACM Transactions on Sensor Networks 2 (4) (2006) 500–528.

[4] S. Basagni, K. Herrin, D. Bruschi, E. Rosti, Secure Pebblenets, in: ACM International Symposium on Mobile Ad Hoc Networking and Computing (MobiHoc), 156–163, 2001.

[5] R. J. Anderson, H. Chan, A. Perrig, Key Infection: Smart Trust for Smart Dust, in: IEEE International Conference on Network Protocols (ICNP), 206–215, 2004.

[6] W. Diffiel, M. E. Hellman, New Directions in Cryptography, IEEE Transactions on Information Theory 22 (6) (1976) 644–654.

[7] R. L. Rivest, A. Shamir, L. M. Adleman, A Method for Obtaining Digital Signatures and Public-Key Cryptosystems, Communications of the ACM 21 (2) (1978) 120–126.

[8] M. Hennessy, T. Regan, A Process Algebra for Timed Systems, Information and Computation 117 (2) (1995) 221–239.

[9] B. Sundararaman, U. Buy, A. D. Kshemkalyani, Clock Synchronization for Wireless Sensor Networks: a Survey, Ad Hoc Networks 3 (3) (2005) 281–323.

[10] T. Nipkow, L. C. Paulson, M. Wenzel, Isabelle/HOL - A Proof Assistant for Higher-Order Logic, vol. 2283 of *Lecture Notes in Computer Science*, Springer, 2002.

[11] Y. Bertot, A Short Presentation of Coq, in: International Conference on Theorem Proving in Higher Order Logics (TPHOLs 2007), vol. 5170 of *LNCS*, 12–16, 2008.

[12] R. Gorrieri, F. Martinelli, A Simple Framework for Real-Time Cryptographic Protocol Analysis with Compositional Proof Rules, Science of Computer Programming 50 (1-3) (2004) 23–49.

[13] R. Gorrieri, F. Martinelli, M. Petrocchi, Formal Models and Analysis of Secure Multicast in Wired and Wireless Networks, Journal of Automated Reasoning 41 (3-4) (2008) 325–364.

[14] S. Misra, I. Woungag, Guide to Wireless Ad Hoc Networks, Computer Communications and Networks, Springer London, ISBN 978-1-84800-327-9, 2009.

[15] X. Nicollin, J. Sifakis, An Overview and Synthesis on Timed Process Algebras, in: International Workshop on Computer Aided Verification (CAV), vol. 575 of *Lecture Notes in Computer Science*, Springer, 376–398, 1991.

[16] R. Focardi, F. Martinelli, A Uniform Approach for the Definition of Security Properties, in: World Congress on Formal Methods (FM), vol. 1708 of *Lecture Notes in Computer Science*, Springer, 794–813, 1999.

[17] A. Perrig, R. Szewczyk, V. Wen, D. E. Culler, J. D. Tygar, SPINS: Security Protocols for Sensor Netowrks, in: International Conference on Mobile Computing and Networking (MOBICOM), 189–199, 2001.

[18] D. Macedonio, M. Merro, A Semantics Analysis of Wireless Network Security Protocols, in: NASA Formal Methods Symposium (NFM), vol. 7226 of *Lecture Notes in Computer Science*, Springer, Heidelberg, 403–417, 2012.

[19] F. Ballardin, M. Merro, A Calculus for the Analysis of Wireless Network Security Protocols, in: Formal Aspects in Security and Trust (FAST), vol. 6561 of *Lecture Notes in Computer Science*, Springer, 206–222, 2010.

[20] R. Milner, Communication and Concurrency, Prentice Hall, 1989.

[21] I. Lanese, D. Sangiorgi, An Operational Semantics for a Calculus for Wireless Systems, Theoretical Computer Science 411 (2010) 1928–1948.

[22] S. Nanz, C. Hankin, A Framework for Security Analysis of Mobile Wireless Networks, Theoretical Computer Science 367 (1-2) (2006) 203–227.

[23] M. Merro, An Observational Theory for Mobile Ad Hoc Networks (full paper), Information and Computation 207 (2) (2009) 194–208.

[24] J. Godskesen, A Calculus for Mobile Ad Hoc Networks, in: Coordination Models and Languages (COORDINATION), vol. 4467 of *Lecture Notes in Computer Science*, Springer, 132–150, 2007.

[25] A. Singh, C. R. Ramakrishnan, S. A. Smolka, A Process Calculus for Mobile Ad Hoc Networks, in: Coordination Models and Languages (COORDINATION), vol. 5052 of *Lecture Notes in Computer Science*, Springer, 296–314, 2008.

[26] F. Ghassemi, W. Fokkink, A. Movaghar, Equational Reasoning on Mobile Ad Hoc Networks, Fundamenta Informaticae 105 (4) (2010) 375–415.

[27] F. Ghassemi, W. Fokkink, A. Movaghar, Verification of Mobile Ad Hoc Networks: an Algebraic Approach, Theoretical Computer Science 412 (28) (2011) 3262–3282.

[28] M. Merro, E. Sibilio, A Timed Calculus for Wireless Systems, in: Fundamentals of Software Engineering (FSEN), vol. 5961 of *Lecture Notes in Computer Science*, Springer, 228–243, 2010.

[29] J. C. Godskesen, S. Nanz, Mobility Models and Behavioural Equivalence for Wireless Networks, in: Coordination Models and Languages (COORDINATION), vol. 5521 of *Lecture Notes in Computer Science*, Springer, 106–122, 2009.

[30] L. Gallina, S. Rossi, A Calculus for Power-aware Multicast Communications in Ad Hoc Networks, in: Theoretical Computer Science - 6th IFIP TC 1/WG 2.2 International Conference (IFIP TCS), vol. 323 of *IFIP*, Springer, 20–31, 2010.

[31] L. Song, J. C. Godskesen, Probabilistic Mobility Models for Mobile and Wireless Networks, in: Theoretical Computer Science - 6th IFIP TC 1/WG 2.2 International Conference (IFIP TCS), vol. 323 of *IFIP*, Springer, 86–100, 2010.

[32] D. Kouzapas, A. Philippou, A Process Calculus for Dynamic Networks, in: Formal Techniques for Distributed Systems (FMOODS/FORTE), vol. 6722 of *Lecture Notes in Computer Science*, Springer, 213–227, 2011.

[33] M. Arnaud, V. Cortier, S. Delaune, Modeling and Verifying Ad Hoc Routing Protocols, in: IEEE Computer Security Foundations Symposium (CSF), IEEE Computer Society, 59–74, 2010.

[34] P. Papadimitratos, Z. J. Haas, Secure Link State Routing for Mobile Ad Hoc Networks, in: Symposium on Applications and the Internet Workshops (SAINT), ACM, 379–383, 2003.

[35] D. B. Johnson, D. A. Maltz, Dynamic Source Routing in ad Hoc Wireless Networks, in: Mobile Computing, Kluwer Academic Publishers, 153–181, 1996.

[36] A. Armando, D. A. Basin, Y. Boichut, Y. Chevalier, L. Compagna, J. Cuéllar, P. H. Drielsma, P.-C. Héam, O. Kouchnarenko, J. Mantovani, S. Mödersheim, D. von Oheimb, M. Rusinowitch, J. Santiago, M. Turuani, L. Viganò, L. Vigneron, The AVISPA Tool for the Automated Validation of Internet Security Protocols and Applications, in: International Conference on Computer Aided Verification (CAV), vol. 3576 of *Lecture Notes in Computer Science*, Springer, 281–285, 2005.

[37] M. L. Tobarra, D. Cazorla, F. Cuartero, G. Díaz, M.-E. Cambronero, Model Checking Wireless Sensor Network Security Protocols: TinySec + LEAP + TinyPK, Telecommunication Systems 40 (3-4) (2009) 91–99.

[38] C. Karlof, N. Sastry, D. Wagner, TinySec: a Link Layer Security Architecture for Wireless Sensor Networks, in: International Conference on Enbedded Networked Sensor Systems (SenSys), ACM, 162–175, 2004.

[39] S. Zhu, S. Setia, S. Jajodia, LEAP - Efficient Security Mechanisms for Large-Scale Distributed Sensor Networks, in: International Conference on Enbedded Networked Sensor Systems (SenSys), ACM, 308–309, 2003.

[40] R. J. Watro, D. Kong, S. fen Cuti, C. Gardiner, C. Lynn, P. Kruus, TinyPK: securing sensor networks with public key technology, in: ACM Workshop on Security of ad hoc and Sensor Networks (SASN), ACM, 59–64, 2004.

[41] M. L. Tobarra, D. Cazorla, F. Cuartero, Formal Analysis of Sensor Network Encryption Protocol (SNEP), in: IEEE International Conference on Mobile Adhoc and Sensor Systems (MASS), IEEE CS, 1–6, 2007.

[42] M. Luk, G. Mezzour, A. Perrig, V. D. Gligor, MiniSec: a Secure Sensor Network Communication Architecture, in: International Conference on Information Processing in Sensor Networks (IPSN), ACM, 479–488, 2007.

[43] S. Blackshear, R. Verma, R-LEAP+: Randomizing LEAP+ Key Distribution to Resist Replay and Jamming Attacks, in: ACM Symposium

on Applied Computing (SAC), ACM Press, 1985–1992, 2010.
[44] C. H. Lim, LEAP++: A Robust Key Establishment Scheme for Wireless Sensor Networks, in: IEEE International Conference on Distributed Computing Systems Workshops (ICDCS), IEEE Computer Society, 376–381, 2008.

Appendix A. Proofs

Proof of Proposition 2.4 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). $\square$

Proposition Appendix A.1 *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 $N \xRightarrow{m?w} N'$ 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 $N \xRightarrow{m?w} N'$. $\square$

Proof of Theorem 2.12 We prove that the relation

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

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 $\eta = \mu \setminus \mathsf{nds}(O)$. 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 $N \xRightarrow{!w\rhd\mu} N'$ with $M' \lesssim N'$. This implies that there exists $h \in \mathsf{nds}(N)$ such that $N \xRightarrow{h!w\rhd\mu'} N'$ 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 2.4(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 $N \mid O \xRightarrow{h!w\rhd\eta'} N' \mid O'$ 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 $N \mid O \xRightarrow{!w\rhd\nu} N' \mid O'$. 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 Appendix A.1 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. □

Proof of Lemma 3.10 We first note that a straightforward consequence of Definition 3.7 is:

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

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

$$(M_1 \mid M_2)^{\mathcal{A}_1\uplus\mathcal{A}_2}_{O_1\uplus O_2} \ \Big|\ \textsc{Top}^{\phi_0}_{\mathcal{A}_1\uplus\mathcal{A}_2/\mathsf{nds}(M)} \ \lesssim\ (M_1)^{\mathcal{A}_1}_{O_1} \ \Big|\ (M_2)^{\mathcal{A}_2}_{O_2} \ \Big|\ \textsc{Top}^{\phi_0}_{\mathcal{A}_1\uplus\mathcal{A}_2/\mathsf{nds}(M)} \ .$$

To improve readability, we consider the most general case, that is $O_1 = \mathsf{nds}(M_1)$ and $O_2 = \mathsf{nds}(M_2)$. 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:

- $(M_1 \mid M_2)^{\mathcal{A}_1\uplus\mathcal{A}_2} = m_1[P_1]^{\nu'_1} \mid m_2[P_2]^{\nu'_2}$ 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:

$$\begin{aligned}\mathcal{R} \stackrel{\text{def}}{=} \ \textstyle\bigcup_{j\geq 0} \{ &(m_1[Q_1]^{\nu'_1} \mid m_2[Q_2]^{\nu'_2} \mid N \,,\, m_1[Q_1]^{\nu''_1} \mid m_2[Q_2]^{\nu''_2} \mid \textsc{Top}^{\phi_j}_{\mathcal{A}/\mathcal{P}}) \\ &\text{such that } m_1[P_1]^{\nu'_1} \mid m_2[P_2]^{\nu'_2} \mid \textsc{Top}^{\phi_0}_{\mathcal{A}/\mathcal{P}} \xRightarrow{\Lambda} m_1[Q_1]^{\nu'_1} \mid m_2[Q_2]^{\nu'_2} \mid N \\ &\text{for some } \Lambda \text{ with } \#^{\sigma}(\Lambda) = j \} \ .\end{aligned}$$

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 $m_1[Q_1]^{\nu''_1} \mid m_2[Q_2]^{\nu''_2} \mid \textsc{Top}^{\phi_j}_{\mathcal{A}/\mathcal{P}} \xrightarrow{\sigma} m_1[\hat{Q}_1]^{\nu''_1} \mid m_2[\hat{Q}_2]^{\nu''_2} \mid \textsc{Top}^{\phi_{j+1}}_{\mathcal{A}/\mathcal{P}}$.

$\alpha = !w\triangleright\nu$. 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

$$m_1[Q_1]^{\nu''_1} \mid m_2[Q_2]^{\nu''_2} \mid \textsc{Top}^{\phi_j}_{\mathcal{A}/\mathcal{P}} \xRightarrow{a_2!w\triangleright\emptyset} m_1[Q_1]^{\nu''_1} \mid m_2[\hat{Q}_2]^{\nu''_2} \mid \textsc{Top}^{\phi_j}_{\mathcal{A}/\mathcal{P}}$$

and by rule (Shh) $m_1[Q_1]^{\nu''_1} \mid m_2[Q_2]^{\nu''_2} \mid \textsc{Top}^{\phi_j}_{\mathcal{A}/\mathcal{P}} \xRightarrow{\tau} m_1[Q_1]^{\nu''_1} \mid m_2[\hat{Q}_2]^{\nu''_2} \mid \textsc{Top}^{\phi_j}_{\mathcal{A}/\mathcal{P}}$. 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) both $m_2[\hat{Q}_2]^{\nu''_2} \xrightarrow{m_1?w} m_2[\hat{Q}_2]^{\nu''_2}$ and $\textsc{Top}^{\phi_j}_{\mathcal{A}/\mathcal{P}} \xrightarrow{m_1?w} \textsc{Top}^{\phi_j}_{\mathcal{A}/\mathcal{P}}$. Thus $m_1[Q_1]^{\nu''_1} \mid m_2[\hat{Q}_2]^{\nu''_2} \mid \textsc{Top}^{\phi_j}_{\mathcal{A}/\mathcal{P}} \xrightarrow{m_1!w\triangleright obs} m_1[\hat{Q}_1]^{\nu''_1} \mid m_2[\hat{Q}_2]^{\nu''_2} \mid \textsc{Top}^{\phi_j}_{\mathcal{A}/\mathcal{P}}$.

$\alpha = \tau$. 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 $\textsc{Top}^{\phi_j}_{\mathcal{A}/\mathcal{P}} \xRightarrow{a_i!w\triangleright m_i} \textsc{Top}^{\phi_j}_{\mathcal{A}/\mathcal{P}}$ and $m_l[Q_l]^{\nu''_l} \xrightarrow{a_i?w} m_l[\hat{Q}_l]^{\nu''_l}$, for $l = 1,2$. Thus $m_1[Q_1]^{\nu''_1} \mid m_2[Q_2]^{\nu''_2} \mid \textsc{Top}^{\phi_j}_{\mathcal{A}/\mathcal{P}} \xrightarrow{\tau} m_1[\hat{Q}_1]^{\nu''_1} \mid m_2[\hat{Q}_2]^{\nu''_2} \mid \textsc{Top}^{\phi_j}_{\mathcal{A}/\mathcal{P}}$. □

Lemma Appendix A.2 *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 $O \subseteq \mathsf{nds}(M)$ then*

$$M^{\mathcal{A}}_{O} \,\big|\, A \ \lesssim\ M^{\mathcal{A}}_{O} \,\big|\, \textsc{Top}^{\phi_0}_{\mathcal{A}/\mathsf{nds}(M)} \quad \textit{for every } A \in \mathbb{A}^{\phi_0}_{\mathcal{A}/\mathsf{nds}(M)} \ .$$

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

$$\mathcal{R} \stackrel{\text{def}}{=} \bigcup_{j\geq 0} \{ (M' \mid A',\ M' \mid \textsc{Top}^{\phi_j}_{\mathcal{A}/\mathsf{nds}(M)}) \text{ s.t. } M^{\mathcal{A}} \mid A \xRightarrow{\Lambda} M' \mid A' \text{ with } \mathsf{nds}(M') = \mathsf{nds}\left(M^{\mathcal{A}}\right) \text{ and } \#^{\sigma}(\Lambda) = j \}$$

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

$\alpha = m?w$. As for Lemma 3.10, this case is straightforward.

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

$\alpha = !w\triangleright\nu$. 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 $\textsc{Top}^{\phi_j}_{\mathcal{A}/\mathsf{nds}(M)} \xrightarrow{m?w} \textsc{Top}^{\phi_j}_{\mathcal{A}/\mathsf{nds}(M)}$ by rule (RcvEnb). Hence $M' \mid \textsc{Top}^{\phi_j}_{\mathcal{A}/\mathsf{nds}(M)} \xrightarrow{m!w\triangleright obs} M'' \mid \textsc{Top}^{\phi_j}_{\mathcal{A}/\mathsf{nds}(M)}$ by rule (Bcast) and the fact that $\mathsf{nds}(A') = \mathcal{A} = \mathsf{nds}\left(\textsc{Top}^{\phi_j}_{\mathcal{A}/\mathsf{nds}(M)}\right)$. Finally, by rule (Obs): $M' \mid \textsc{Top}^{\phi_j}_{\mathcal{A}/\mathsf{nds}(M)} \xrightarrow{!w\triangleright obs} M'' \mid \textsc{Top}^{\phi_j}_{\mathcal{A}/\mathsf{nds}(M)}$.

$\alpha = \tau$. The most significant case is when the transition 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 $\mathsf{nds}(M') = \mathsf{nds}\left(M^{\mathcal{A}}\right)$ 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 3.11 By Lemma Appendix A.2 we have $M^{\mathcal{A}}_{O} \mid A \lesssim M^{\mathcal{A}}O \mid \textsc{Top}^{\phi_0}_{\mathcal{A}/\mathsf{nds}(M)}$ for every $A \in \mathbb{A}^{\phi_0}_{\mathcal{A}/\mathsf{nds}(M)}$. Then by transitivity of $\lesssim$ we have $M^{\mathcal{A}}O \mid A \lesssim N$ for every $A \in \mathbb{A}^{\phi_0}_{\mathcal{A}/\mathsf{nds}(M)}$ and we conclude that M is $tGNDC^{N}_{\phi_0,O}$. □

Proof of Proposition 4.1 By induction on i we show that whenever $\textsc{bs}[\sigma.S_1]^{obs} \xRightarrow{\Lambda} \textsc{bs}[S_i]^{obs}$ or $m_j[\sigma.\hat{R}_1]^{obs} \xRightarrow{\Lambda} m_j[\hat{R}_i]^{obs}$ then $\#^{\sigma}(\Lambda) = 2i-1$. Moreover, we observe that $!p_i\triangleright obs$ can be performed exclusively as $\textsc{bs}[S_i]^{obs} \xrightarrow{!p_i\triangleright obs}$. While $!\mathsf{end}_i\triangleright obs$ can be performed exclusively as $m_j[\hat{R}_i]^{obs} \xRightarrow{\Omega} \xrightarrow{!\mathsf{auth}_i\triangleright obs}$ with $\#^{\sigma}\Omega = 2$. Hence we deduce that:

1. if $\textsc{bs}[\sigma.S_1]^{obs} \xRightarrow{\Lambda} \xrightarrow{!p_i\triangleright obs}$ then $\#^{\sigma}(\Lambda) = 2i - 1$;
2. if $m_j[\hat{R}_1]^{obs} \xRightarrow{\Lambda} \xrightarrow{!\mathsf{auth}_i\triangleright obs}$ for some $1 \le j \le k$, then $\#^{\sigma}(\Lambda) = 2(i-1) + 4$.

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

Proof of Lemma 4.2 We provide the proper simulation in both the cases.
Case 1: Base Station. We notice that any process S_i, along with its derivatives, cannot receive any message. Thus an attacker in b cannot affect the behaviour of $\textsc{bs}[\sigma.S_1]^{\{b,obs\}}$. Hence it is straightforward to prove that $\textsc{bs}[\sigma.S_1]^{\{b,obs\}} \mid \textsc{Top}^{\phi_0}_{b/\textsc{bs}} \lesssim \textsc{bs}[\sigma.S_1]^{obs}$.
Case 2: Nodes. We fix a node $m \in \{m_1, \ldots, m_h\}$, we let $a \in \{a_1, \ldots, a_h\}$ denote the corresponding attacking place and we show that

$$m[\sigma.R'\langle 1, -1, \perp, \bar{k}\rangle]^{\{a,obs\}} \mid \textsc{Top}^{\phi_0}_{a/m} \quad \lesssim \quad m[\sigma.\hat{R}_1]^{obs} .$$

To uniform the notation, we define $k_{-1} \stackrel{\text{def}}{=} \bar{k}$. We pick the indexes $i \geq 1$ and $-1 \geq l \geq i-2$, and the messages $\hat{r}$, $\hat{p}$, $\hat{k}$ and $\hat{q}$. Then we build the relation $\mathcal{R}^{l,\hat{r}}_{i}(\hat{p}, \hat{k}, \hat{q})$ which contains the pair

$$\left(m[R'\langle i, l, \hat{r}, k_l\rangle]^{\{a,obs\}} \mid \textsc{Top}^{\phi_{2(i-1)}}_{a/m} \ ,\ m[\hat{R}_i]^{obs}\right)$$

along with its derivatives which may be generated when m first receives $\hat{p}$ and then $\hat{k}$ from the attacker. To improve the readability: (i) we define $\nu'_m \stackrel{\text{def}}{=} \{a, obs\}$, (ii) we employ the structural congruence $\equiv$ to rewrite the process $\hat{R}_i$ as:

$$\hat{R}_i \stackrel{\text{def}}{=} \sigma.\hat{P}_i \qquad \hat{P}_i \stackrel{\text{def}}{=} \lfloor \tau.\sigma.\hat{Z}_{i+1} \rfloor \hat{R}_{i+1} \qquad \hat{Z}_i \stackrel{\text{def}}{=} !\langle \mathsf{auth}_{i-2}\rangle.\hat{R}_i \ .$$

Then we define

$$
\begin{aligned}
\mathcal{R}_i^{l,\hat r}(\hat p,\hat k,\hat q) \stackrel{\text{def}}{=} \Big\{ &\Big(m[R'\langle i,l,\hat r,k_l\rangle]^{\nu'_m} \mid \mathrm{Top}^{\phi_{2i-1}}_{a/m},\ m[\hat R_i]^{obs} \Big)\ , \\
&\Big(m[R'\langle i,l,\hat r,k_l\rangle]^{\nu'_m} \mid a[!\langle \hat p\rangle.\mathrm{T}_{\phi_{2i-1}}]^{m},\ m[\hat R_i]^{obs} \Big)\ , \\
&\Big(m[\sigma.P'\langle i,l,\hat p,\hat r,k_l\rangle]^{\nu'_m} \mid \mathrm{Top}^{\phi_{2i-1}}_{a/m},\ m[\hat R_i]^{obs} \Big)\ , \\
&\Big(m[\sigma.P'\langle i,l,\hat p,\hat r,k_l\rangle]^{\nu'_m} \mid a[!\langle \hat p\rangle.\mathrm{T}_{\phi_{2i-1}}]^{m},\ m[\hat R_i]^{obs} \Big)\ , \\
&\Big(m[P'\langle i,l,\hat p,\hat r,k_l\rangle]^{\nu'_m} \mid \mathrm{Top}^{\phi_{2i}}_{a/m},\ m[\hat P_i]^{obs} \Big)\ , \\
&\Big(m[P'\langle i,l,\hat p,\hat r,k_l\rangle]^{\nu'_m} \mid a[!\langle \hat k\rangle.\mathrm{T}_{\phi_{2i}}]^{m},\ m[\hat P_i]^{obs} \Big)\ , \\
&\Big(m[T'\langle i,l,\hat p,\hat r,k_l,\hat k\rangle]^{\nu'_m} \mid \mathrm{Top}^{\phi_{2i}}_{a/m},\ m[\hat P_i]^{obs} \Big)\ , \\
&\Big(m[T'\langle i,l,\hat p,\hat r,k_l,\hat k\rangle]^{\nu'_m} \mid a[!\langle \hat k\rangle.\mathrm{T}_{\phi_{2i}}]^{m},\ m[\hat P_i]^{obs} \Big)\ , \\
&\Big(m[Q'\langle i,l,\hat r,k_l\rangle]^{\nu'_m} \mid \mathrm{Top}^{\phi_{2i}}_{a/m},\ m[\hat P_i]^{obs} \Big)\Big\}\ , \\
&\Big(m[Q'\langle i,l,\hat r,k_l\rangle]^{\nu'_m} \mid a[!\langle \hat k\rangle.\mathrm{T}_{\phi_{2i}}]^{m},\ m[\hat P_i]^{obs} \Big)\Big\}\ , \\
&\Big(m[Z'\langle i{+}1,i{-}1,\hat p,\hat r,k_{i-1}\rangle]^{\nu'_m} \mid \mathrm{Top}^{\phi_{2i+1}}_{a/m},\ m[\hat Z_{i+1}]^{obs} \Big)\ , \\
&\Big(m[Z'\langle i{+}1,i{-}1,\hat p,\hat r,k_{i-1}\rangle]^{\nu'_m} \mid a[!\langle \hat q\rangle.\mathrm{T}_{\phi_{2i+1}}]^{m},\ m[\hat Z_{i+1}]^{obs} \Big)\Big\}\ .
\end{aligned}
$$

and we show that the required simulation is $(\, m[\sigma.R'\langle 1,-1,\bot,\bar k\rangle]^{\{a,obs\}} \mid \mathrm{Top}^{\phi_0}_{a/m}\ ,\ m[\sigma.\hat R_1]^{obs}\,) \cup \mathcal{R}$, where the relation $\mathcal{R}$ is defined as

$$
\mathcal{R} \stackrel{\text{def}}{=} \bigcup_{i\geq 1}\ \bigcup_{\substack{-1\leq l\leq i-2\\ \hat r\in\mathcal{D}(\phi_{2(i-2)})}}\ \bigcup_{\substack{\hat p\in\mathcal{D}(\phi_{2i-1})\\ \hat k\in\mathcal{D}(\phi_{2i})\\ \hat q\in\mathcal{D}(\phi_{2i+1})}} \mathcal{R}_i^{l,\hat r}(\hat p,\hat k,\hat q)
$$

We outline the 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 τ-actions generated by internal choices of the attacker.

In the pair $(\, m[R'\langle i,l,\hat r,k_l\rangle]^{\nu'_m} \mid \mathrm{Top}^{\phi_{2i-1}}_{a/m},\ m[\hat R_i]^{obs}\,)$ we have a significant action:

- $m[R'\langle i,l,\hat r,k_l\rangle]^{\nu'_m} \mid \mathrm{Top}^{\phi_{2i-1}}_{a/m} \xrightarrow{\sigma} m[Q'\langle i,l,\hat r,k_l\rangle]^{\nu'_m} \mid \mathrm{Top}^{\phi_{2i}}_{a/m}$, where m does not receive anything. Then $m[\hat R_i]^{obs} \stackrel{\sigma}{\Longrightarrow} m[\hat P_i]^{obs}$.

In the pair $(\, m[R'\langle i,l,\hat r,k_l\rangle]^{\nu'_m} \mid a[!\langle \hat p\rangle.\mathrm{T}_{\phi_{2i-1}}]^{m},\ m[\hat R_i]^{obs}\,)$ we have two significant actions:

- $m[R'\langle i,l,\hat r,k_l\rangle]^{\nu'_m} \mid a[!\langle \hat p\rangle.\mathrm{T}_{\phi_{2i-1}}]^{m} \xrightarrow{\tau} m[\sigma.P'\langle i,l,\hat p,\hat r,k_l\rangle]^{\nu'_m} \mid \mathrm{Top}^{\phi_{2i-1}}_{a/m}$, where m receives $\hat p$ from the attacker. Then $m[\hat R_i]^{obs} \Longrightarrow m[\hat R_i]^{obs}$.
- $m[R'\langle i,l,\hat r,k_l\rangle]^{\nu'_m} \mid a[!\langle \hat p\rangle.\mathrm{T}_{\phi_{2i-1}}]^{m} \xrightarrow{\tau} m[R'\langle i,l,\hat r,k_l\rangle]^{\nu'_m} \mid \mathrm{Top}^{\phi_{2i-1}}_{a/m}$, where $\hat p$ gets lost. Then $m[\hat R_i]^{obs} \Longrightarrow m[\hat R_i]^{obs}$.

In the pair $(\, m[\sigma.P'\langle i,l,\hat p,\hat r,k_l\rangle]^{\nu'_m} \mid \mathrm{Top}^{\phi_{2i-1}}_{a/m},\ m[\hat R_i]^{obs}\,)$ we have just a significant action:

- $m[\sigma.P'\langle i,l,\hat p,\hat r,k_l\rangle]^{\nu'_m} \mid \mathrm{Top}^{\phi_{2i-1}}_{a/m} \xrightarrow{\sigma} m[P'\langle i,l,\hat p,\hat r,k_l\rangle]^{\nu'_m} \mid \mathrm{Top}^{\phi_{2i}}_{a/m}$. Then the second network replies with $m[\hat R_i]^{obs} \stackrel{\sigma}{\Longrightarrow} m[\hat P_i]^{obs}$.

In the pair $(\, m[P'\langle i,l,\hat p,\hat r,k_l\rangle]^{\nu'_m} \mid \mathrm{Top}^{\phi_{2i}}_{a/m},\ m[\hat P_i]^{obs}\,)$ we have just a significant action:

- $m[P'\langle i,l,\hat p,\hat r,k_l\rangle]^{\nu'_m} \mid \mathrm{Top}^{\phi_{2i}}_{a/m} \xrightarrow{\sigma} m[R'\langle i{+}1,l,\hat p,k_l\rangle]^{\nu'_m} \mid \mathrm{Top}^{\phi_{2i+1}}_{a/m}$, where m does not receive anything. Then $m[\hat P_i]^{obs} \stackrel{\sigma}{\Longrightarrow} m[\hat R_{i+1}]^{obs}$.

In the pair $(\, m[P'\langle i,l,\hat p,\hat r,k_l\rangle]^{\nu'_m} \mid a[!\langle \hat k\rangle.\mathrm{T}_{\phi_{2i}}]^{m},\ m[\hat P_i]^{obs}\,)$ we have two significant actions:

- $m[P'\langle i,l,\hat{p},\hat{r},k_l\rangle]^{\nu'_m} \mid a[!\langle\hat{k}\rangle.\mathrm{T}_{\phi_{2i}}]^m \xrightarrow{\tau} m[T'\langle i,l,\hat{p},\hat{r},k_l,\hat{k}\rangle]^{\nu'_m} \mid \mathrm{Top}^{\phi_{2i}}_{a/m}$, where m receives $\hat{k}$. Then the second network replies with $m[\hat{P}_i]^{obs} \Longrightarrow m[\hat{P}_i]^{obs}$.
- $m[P'\langle i,l,\hat{p},\hat{r},k_l\rangle]^{\nu'_m} \mid a[!\langle\hat{k}\rangle.\mathrm{T}_{\phi_{2i}}]^m \xrightarrow{\tau} m[P'\langle i,l,\hat{p},\hat{r},k_l\rangle]^{\nu'_m} \mid \mathrm{Top}^{\phi_{2i}}_{a/m}$, where $\hat{k}$ gets lost. The second network replies with $m[\hat{P}_i]^{obs} \Longrightarrow m[\hat{P}_i]^{obs}$.

In the pair ($m[T'\langle i,l,\hat{p},\hat{r},k_l,\hat{k}\rangle]^{\nu'_m} \mid \mathrm{Top}^{\phi_{2i}}_{a/m},\ m[\hat{P}_i]^{obs}$) we have three significant actions:

- $m[T'\langle i,l,\hat{p},\hat{r},k_l,\hat{k}\rangle]^{\nu'_m} \mid \mathrm{Top}^{\phi_{2i}}_{a/m} \xrightarrow{\sigma} m[Z'\langle i{+}1,i{-}1,\hat{p},\hat{r},k_{i-1}\rangle]^{\nu'_m} \mid \mathrm{Top}^{\phi_{2i+1}}_{a/m}$ where m checks that $k_l = F^{i-1-l}(\hat{h})$ and authenticates $\hat{r} = p_{i-1}$. Then $m[\hat{P}_i]^{obs} \stackrel{\sigma}{\Longrightarrow} m[\hat{Z}_{i+1}]^{obs}$.
- $m[T'\langle i,l,\hat{p},\hat{r},k_l,\hat{k}\rangle]^{\nu'_m} \mid \mathrm{Top}^{\phi_{2i}}_{a/m} \xrightarrow{\sigma} m[R'\langle i{+}1,i{-}1,\hat{p},k_{i-1}\rangle]^{\nu'_m} \mid \mathrm{Top}^{\phi_{2i+1}}_{a/m}$ where m checks that $k_l = F^{i-l}(\hat{h})$ without but it does not authenticate $\hat{r}$. Then $m[\hat{P}_i]^{obs} \stackrel{\sigma}{\Longrightarrow} m[\hat{R}_{i+1}]^{obs}$.
- $m[T'\langle i,l,\hat{p},\hat{r},k_l,\hat{k}\rangle]^{\nu'_m} \mid \mathrm{Top}^{\phi_{2i}}_{a/m} \xrightarrow{\sigma} m[R'\langle i{+}1,l,\hat{p},k_l\rangle]^{\nu'_m} \mid \mathrm{Top}^{\phi_{2i+1}}_{a/m}$ where m verifies $k_l \neq F^{i-l}(\hat{h})$. Then again $m[\hat{P}_i]^{obs} \stackrel{\sigma}{\Longrightarrow} m[\hat{R}_{i+1}]^{obs}$ by timeout.

In the pair ($m[Q'\langle i,l,\hat{r},k_l\rangle]^{\nu'_m} \mid \mathrm{Top}^{\phi_{2i}}_{a/m},\ m[\hat{P}_i]^{obs}$) we have a significant action

- $m[Q'\langle i,l,\hat{r},k_l\rangle]^{\nu'_m} \mid \mathrm{Top}^{\phi_{2i}}_{a/m} \xrightarrow{\sigma} m[R\langle i{+}1,l,\hat{r},k_l\rangle]^{\nu'_m} \mid \mathrm{Top}^{\phi_{2i+1}}_{a/m}$, where m does not receive anything and thus performs a timeout. Then $m[\hat{P}_i]^{obs} \stackrel{\sigma}{\Longrightarrow} m[\hat{R}_{i+1}]^{obs}$.

In the pair ($m[Q'\langle i,l,\hat{r},k_l\rangle]^{\nu'_m} \mid a[!\langle\hat{k}\rangle.\mathrm{T}_{\phi_{2i}}]^m,\ m[\hat{P}_i]^{obs}$) the first network can perform two significant actions

- $m[Q'\langle i,l,\hat{r},k_l\rangle]^{\nu'_m} \mid a[!\langle\hat{k}\rangle.\mathrm{T}_{\phi_{2i}}]^m \xrightarrow{\tau} m[T\langle i,l,\hat{r},\hat{r},k_l,\hat{k}\rangle]^{\nu'_m} \mid \mathrm{Top}^{\phi_{2i}}_{a/m}$, where m receives $\hat{k}$. Then the second network replies $m[\hat{P}_i]^{obs} \Longrightarrow m[\hat{P}_i]^{obs}$.
- $m[Q'\langle i,l,\hat{r},k_l\rangle]^{\nu'_m} \mid a[!\langle\hat{k}\rangle.\mathrm{T}_{\phi_{2i}}]^m \xrightarrow{\tau} m[R\langle i{+}1,l,\hat{r},k_l\rangle]^{\nu'_m} \mid \mathrm{Top}^{\phi_{2i}}_{a/m}$, where $\hat{k}$ gets lost. Then the second network replies $m[\hat{P}_i]^{obs} \Longrightarrow m[\hat{P}_i]^{obs}$. □

Proof of Proposition 5.1 Similar to that of Proposition 4.1. □

Proof of Theorem 5.2 The system $(\mathrm{LEAP'{+}})^{\mathcal{A}} \mid A$ admits the following computation:

$$
\begin{array}{lr}
m[\sigma.S_1]^{\nu_m} \mid r[\sigma.R']^{\nu_r} \mid A & \xrightarrow{\sigma} \\
m[S_1]^{\nu_m} \mid r[R']^{\nu_r} \mid a[X]^{\nu_a} \mid b[\sigma.X]^{\nu_b} & \xrightarrow{!\mathsf{hello}_1 \triangleright obs} \\
m[\sigma.P]^{\nu_m} \mid r[R']^{\nu_r} \mid a[\sigma.!\langle\mathsf{hello}_1\rangle.\mathsf{nil}]^{\nu_a} \mid b[\sigma.X]^{\nu_b} & \xrightarrow{\sigma} \\
m[P]^{\nu_m} \mid r[\sigma.R']^{\nu_r} \mid a[!\langle\mathsf{hello}_1\rangle.\mathsf{nil}]^{\nu_a} \mid b[X]^{\nu_b} & \xrightarrow{\tau} \\
m[\{^{\mathsf{hello}_1}/_q\}P^1]^{\nu_m} \mid r[\sigma.R']^{\nu_r} \mid a[\mathsf{nil}]^{\nu_a} \mid b[\sigma.!\langle\mathsf{hello}_1\rangle.\mathsf{nil}]^{\nu_b} & \xrightarrow{\sigma} \\
m[S_2]^{\nu_m} \mid r[R']^{\nu_r} \mid a[\mathsf{nil}]^{\nu_a} \mid b[!\langle\mathsf{hello}_1\rangle.\mathsf{nil}]^{\nu_b} & \xrightarrow{\tau} \\
m[S_2]^{\nu_m} \mid r[\sigma.!\langle q_1\rangle.R^{8\prime}]^{\nu_r} \mid a[\mathsf{nil}]^{\nu_a} \mid b[\mathsf{nil}]^{\nu_b} & \xrightarrow{!\mathsf{hello}_2 \triangleright obs} \\
m[\sigma.P]^{\nu_m} \mid r[\sigma.!\langle q_1\rangle.R^{8\prime}]^{\nu_r} \mid a[\mathsf{nil}]^{\nu_a} \mid b[\mathsf{nil}]^{\nu_b} & \xrightarrow{\sigma} \\
m[P]^{\nu_m} \mid r[!\langle q_1\rangle.R^{8\prime}]^{\nu_r} \mid a[\mathsf{nil}]^{\nu_a} \mid b[\mathsf{nil}]^{\nu_b} & \xrightarrow{!q_1 \triangleright obs} \\
m[\{^{q_1}/_q\}P_1]^{\nu_m} \mid r[R^{8\prime}]^{\nu_r} \mid a[\mathsf{nil}]^{\nu_a} \mid b[\mathsf{nil}]^{\nu_b} & \xrightarrow{\sigma} \\
m[S^3]^{\nu_m} \mid r[!\langle\mathsf{end}_1\rangle.\mathsf{nil}]^{\nu_r} \mid a[\mathsf{nil}]^{\nu_a} \mid b[\mathsf{nil}]^{\nu_b} & \xrightarrow{!\mathsf{end}_1 \triangleright obs}
\end{array}
$$

Then agreement is not reached. □

Proof of Lemma 5.4 We prove this lemma by showing the appropriate simulations.

Case 1: Sender. We define $\nu'_m = \{a, obs\}$. We need to prove $m[\sigma.S_1]^{\nu'_m} \mid \textsc{Top}^{\phi_0}_{a/m} \lesssim m[\sigma.\hat{S}_1]^{obs}$. Thus we fix an index $i = 1, 2, \ldots$, we pick the messages $q' \in \mathcal{D}(\phi_{2i-1})$ and $\hat{q} \in \mathcal{D}(\phi_{2i})$, and we build the relation $\mathcal{R}_i(q', \hat{q})$ containing $(\, m[S''_i]^{\nu'_m} \mid \textsc{Top}^{\phi_{2i-1}}_{a/m},\ m[\hat{S}_i]^{obs}\,)$ along with its derivatives which may be generated when m receives $\hat{q}$ from the attacker.

$$
\begin{array}{rcl}
\mathcal{R}_i(q', \hat{q}) & \stackrel{\text{def}}{=} & \Big\{ \Big(m[S''_i]^{\nu'_m} \mid \textsc{Top}^{\phi_{2i-1}}_{a/m},\ m[\hat{S}_i]^{obs} \Big)\ , \\
& & \Big(m[S''_i]^{\nu'_m} \mid a[!\langle q' \rangle.\mathrm{T}_{\phi_{2i-1}}]^m,\ m[\hat{S}_i]^{obs} \Big)\ , \\
& & \Big(m[\sigma.P'']^{\nu'_m} \mid \textsc{Top}^{\phi_{2i-1}}_{a/m},\ m[\sigma.\lfloor \tau.\sigma.!\langle \mathsf{auth}_i \rangle.\mathsf{nil} \rfloor \hat{S}_{i+1}]^{obs} \Big)\ , \\
& & \Big(m[\sigma.P'']^{\nu'_m} \mid a[!\langle q' \rangle.\mathrm{T}_{\phi_{2i-1}}]^m,\ m[\sigma.\lfloor \tau.\sigma.!\langle \mathsf{auth}_i \rangle.\mathsf{nil} \rfloor \hat{S}_{i+1}]^{obs} \Big)\ , \\
& & \Big(m[P'']^{\nu'_m} \mid \textsc{Top}^{\phi_{2i}}_{a/m},\ m[\lfloor \tau.\sigma.!\langle \mathsf{auth}_i \rangle.\mathsf{nil} \rfloor \hat{S}_{i+1}]^{obs} \Big)\ , \\
& & \Big(m[P'']^{\nu'_m} \mid a[!\langle \hat{q} \rangle.\mathrm{T}_{\phi_{2i}}]^m,\ m[\lfloor \tau.\sigma.!\langle \mathsf{auth}_i \rangle.\mathsf{nil} \rfloor \hat{S}_{i+1}]^{obs} \Big)\ , \\
& & \Big(m[\{^{\hat{q}}/_{q}\}P^{1\prime\prime}]^{\nu'_m} \mid \textsc{Top}^{\phi_{2i}}_{a/m},\ m[\lfloor \tau.\sigma.!\langle \mathsf{auth}_i \rangle.\mathsf{nil} \rfloor \hat{S}_{i+1}]^{obs} \Big)\ , \\
& & \Big(m[\{^{\hat{q}}/_{q}\}P^{1\prime\prime}]^{\nu'_m} \mid a[!\langle \hat{q} \rangle.\mathrm{T}_{\phi_{2i}}]^m,\ m[\lfloor \tau.\sigma.!\langle \mathsf{auth}_i \rangle.\mathsf{nil} \rfloor \hat{S}_{i+1}]^{obs} \Big)\ , \\
& & \Big(m[!\langle \mathsf{auth}_i \rangle.\mathsf{nil}]^{\nu'_m} \mid \textsc{Top}^{\phi_{2i+1}}_{a/m},\ m[!\langle \mathsf{auth}_i \rangle.\mathsf{nil}]^{obs} \Big) \Big\}\ .
\end{array}
$$

Moreover, it is straightforward to prove that there exists a simulation $\overline{\mathcal{R}}_i$ containing the pair $(\, m[!\langle \mathsf{auth}_i \rangle.\mathsf{nil}]^{\nu'_m} \mid \textsc{Top}^{\phi_{2i+1}}_{a/m},\ m[!\langle \mathsf{auth}_i \rangle.\mathsf{nil}]^{obs}\,)$.

Then we show that the required simulation is given by the following relation

$$
\mathcal{R} \stackrel{\text{def}}{=} \Big\{ m[\sigma.S_1]^{\nu'_m} \mid \textsc{Top}^{\phi_0}_{a/m}, m[\sigma.\hat{S}_1]^{obs} \Big\} \cup \bigcup_{i \geq 1} \Big(\overline{\mathcal{R}}_i \ \cup \bigcup_{\substack{q' \in \mathcal{D}(\phi_{2i-1}) \\ \hat{q} \in \mathcal{D}(\phi_{2i})}} \mathcal{R}_i(q', \hat{q}) \Big)\ .
$$

As done for Lemma 4.2, we outline the most significant cases.

In the pair $(\, m[S''_i]^{\nu'_m} \mid \textsc{Top}^{\phi_{2i-1}}_{a/m},\ m[\hat{S}_i]^{obs}\,)$ we have a significant action:

- $m[S''_i]^{\nu'_m} \mid \textsc{Top}^{\phi_{2i-1}}_{a/m} \xrightarrow{!\mathsf{hello}_i \triangleright obs} m[\sigma.P'']^{\nu'_m} \mid \textsc{Top}^{\phi_{2i-1}}_{a/m}$, where m broadcasts the packet hello_i. Then second network answers with $m[\hat{S}_i]^{obs} \xRightarrow{!\mathsf{hello}_i \triangleright obs} m[\sigma.\lfloor \tau.\sigma.!\langle \mathsf{auth}_i \rangle.\mathsf{nil} \rfloor \hat{S}_{i+1}]^{obs}$.

In the pair $(\, m[\sigma.P'']^{\nu'_m} \mid \textsc{Top}^{\phi_{2i-1}}_{a/m}, m[\sigma.\lfloor \tau.\sigma.!\langle \mathsf{auth}_i \rangle.\mathsf{nil} \rfloor \hat{S}_{i+1}]^{obs}\,)$ we have a significant action:

- $m[\sigma.P'']^{\nu'_m} \mid \textsc{Top}^{\phi_{2i-1}}_{a/m} \xrightarrow{\sigma} m[P'']^{\nu'_m} \mid \textsc{Top}^{\phi_{2i}}_{a/m}$. Then $m[\sigma.\lfloor \tau.\sigma.!\langle \mathsf{auth}_i \rangle.\mathsf{nil} \rfloor \hat{S}_{i+1}]^{obs} \xRightarrow{\sigma} m[\lfloor \tau.\sigma.!\langle \mathsf{auth}_i \rangle.\mathsf{nil} \rfloor \hat{S}_{i+1}]^{obs}$.

In the pair $(\, m[P'']^{\nu'_m} \mid \textsc{Top}^{\phi_{2i}}_{a/m},\ m[\lfloor \tau.\sigma.!\langle \mathsf{auth}_i \rangle.\mathsf{nil} \rfloor \hat{S}_{i+1}]^{obs}\,)$ we have a significant action:

- $m[P'']^{\nu'_m} \mid \textsc{Top}^{\phi_{2i}}_{a/m} \xrightarrow{\sigma} m[S''_{i+1}]^{\nu'_m} \mid \textsc{Top}^{\phi_{2i+1}}_{a/m}$, where m does not receive anything and performs a timeout. Then $m[\lfloor \tau.\sigma.!\langle \mathsf{auth}_i \rangle.\mathsf{nil} \rfloor \hat{S}_{i+1}]^{obs} \xRightarrow{\sigma} m[\hat{S}_{i+1}]^{obs}$.

In the pair $(\, m[P'']^{\nu'_m} \mid a[!\langle \hat{q} \rangle.\mathrm{T}_{\phi_{2i}}]^m,\ m[\lfloor \tau.\sigma.!\langle \mathsf{auth}_i \rangle.\mathsf{nil} \rfloor \hat{S}_{i+1}]^{obs}\,)$ we consider two actions:

- $m[P'']^{\nu'_m} \mid a[!\langle \hat{q} \rangle.\mathrm{T}_{\phi_{2i}}]^m \xrightarrow{\tau} m[\{^{\hat{q}}/_{q}\}P^{1\prime\prime}]^{\nu'_m} \mid \textsc{Top}^{\phi_{2i}}_{a/m}$, where m receives $\hat{q}$. Then the second network replies $m[\lfloor \tau.\sigma.!\langle \mathsf{auth}_i \rangle.\mathsf{nil} \rfloor \hat{S}_{i+1}]^{obs} \Rightarrow m[\lfloor \tau.\sigma.!\langle \mathsf{auth}_i \rangle.\mathsf{nil} \rfloor \hat{S}_{i+1}]^{obs}$.

- $m[P'']^{\nu'_m} \mid a[!\langle \hat{q} \rangle.\mathrm{T}_{\phi_{2i}}]^m \xrightarrow{\tau} m[P'']^{\nu'_m} \mid \textsc{Top}^{\phi_{2i}}_{a/m}$, where $\hat{q}$ gets lost. Then $m[\lfloor \tau.\sigma.!\langle \mathsf{auth}_i \rangle.\mathsf{nil} \rfloor \hat{S}_{i+1}]^{obs} \Rightarrow m[\lfloor \tau.\sigma.!\langle \mathsf{auth}_i \rangle.\mathsf{nil} \rfloor \hat{S}_{i+1}]^{obs}$.

In $(\, m[\{^{\hat{q}}/_{q}\}P^{1\prime\prime}]^{\nu'_m} \mid \textsc{Top}^{\phi_{2i}}_{a/m},\ m[\lfloor \tau.\sigma.!\langle \mathsf{auth}_i \rangle.\mathsf{nil} \rfloor \hat{S}_{i+1}]^{obs}\,)$ we have two significant actions:

- $m[\{^{\hat{q}}/_{q}\}P^{1\prime\prime}]^{\nu'_m} \mid \textsc{Top}^{\phi_{2i}}_{a/m} \xrightarrow{\sigma} m[!\langle \mathsf{auth}_i\rangle.\mathsf{nil}]^{\nu'_m} \mid \textsc{Top}^{\phi_{2i+1}}_{a/m}$, where m verifies that $\hat{q}$ refers to the nonce n_i. Then $m[\lfloor \tau.\sigma.!\langle \mathsf{auth}_i\rangle.\mathsf{nil}\rfloor \hat{S}_{i+1}]^{obs} \stackrel{\sigma}{\Longrightarrow} m[!\langle \mathsf{auth}_i\rangle.\mathsf{nil}]^{obs}$.

- $m[\{^{\hat{q}}/_{q}\}P^{1\prime\prime}]^{\nu'_m} \mid \textsc{Top}^{\phi_{2i}}_{a/m} \xrightarrow{\sigma} m[S''_{i+1}]^{\nu'_m} \mid \textsc{Top}^{\phi_{2i+1}}_{a/m}$, where m verifies that $\hat{q}$ does not refer to n_i, or it finds out that $\hat{q}$ is corrupted. Then $m[\lfloor \tau.\sigma.!\langle \mathsf{auth}_i\rangle.\mathsf{nil}\rfloor \hat{S}_{i+1}]^{obs} \stackrel{\sigma}{\Longrightarrow} m[\hat{S}_{i+1}]^{obs}$.

Case 2: Receiver. To show that $r[\sigma.R]^{\{b\}} \mid \textsc{Top}^{\phi_0}_{b/r} \lesssim r[Tick]^{\emptyset}$ we define the following relation:

$$\mathcal{R} \stackrel{\text{def}}{=} \{(M, r[Tick]^{\emptyset}) \text{ such that } r[\sigma.R]^{\{b\}} \mid \textsc{Top}^{\phi_0}_{b/r} \stackrel{\Lambda}{\Longrightarrow} M\} \ .$$

We first note that for every $(M, r[Tick]^{\emptyset}) \in \mathcal{R}$ we have $\mathsf{Env}\,(M) = \emptyset$. Thus the most significant actions can only be $M \xrightarrow{\tau}$ or $M \xrightarrow{\sigma}$ or input actions that can be derived without applying rule (Rcv). Then it is straightforward to prove that $\mathcal{R}$ is a simulation. □

Proof of Proposition 6.1 Similar to that of Proposition 4.1. □

Proof of Theorem 6.2 The system $(\text{LiSP}')^{\mathcal{A}} \mid A$ admits the following computation:

$$\begin{array}{ll}
m[\sigma.Z']^{\nu_m} \mid \textsc{kl}[\sigma.L_0]^{\nu_{\textsc{kl}}} \mid A & \xrightarrow{\sigma} \\
m[Z']^{\nu_m} \mid \textsc{kl}[L_0]^{\nu_{\textsc{kl}}} \mid a[\sigma^2.X]^{\nu_a} \,\big|\, b[\sigma.X]^{\nu_b} & \xrightarrow{!r\triangleright obs} \\
m[\sigma.\lfloor ?(q).T'\rfloor Z']^{\nu_m} \mid \textsc{kl}[\{^{r}/_{r}\}I_1]^{\nu_{\textsc{kl}}} \mid a[\sigma^2.X]^{\nu_a} \,\big|\, b[\sigma.X]^{\nu_b} & \xrightarrow{\sigma} \\
m[\lfloor ?(q).T'\rfloor Z']^{\nu_m} \mid \textsc{kl}[!\langle q_1\rangle.\sigma.L_1]^{\nu_{\textsc{kl}}} \mid a[\sigma.X]^{\nu_a} \mid b[X]^{\nu_b} & \xrightarrow{!q_1\triangleright obs} \\
m[\lfloor ?(q).T'\rfloor Z']^{\nu_m} \mid \textsc{kl}[\sigma.L_1]^{\nu_{\textsc{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 \textsc{kl}[L_1]^{\nu_{\textsc{kl}}} \mid a[X]^{\nu_a} \mid b[!\langle q_1\rangle.\mathsf{nil}]^{\nu_b} & \xrightarrow{\tau} \\
m[Z']^{\nu_m} \mid \textsc{kl}[\{^{q_1}/_{r}\}I_2]^{\nu_{\textsc{kl}}} \mid a[\sigma.!\langle q_1\rangle.\mathsf{nil}]^{\nu_a} \mid b[\mathsf{nil}]^{\nu_b} & \xrightarrow{!r\triangleright obs} \\
m[\sigma.\lfloor ?(q).T'\rfloor Z']^{\nu_m} \mid \textsc{kl}[\{^{q_1}/_{r}\}I_2]^{\nu_{\textsc{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 \textsc{kl}[\sigma.L_2]^{\nu_{\textsc{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 \textsc{kl}[\sigma.L_2]^{\nu_{\textsc{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 \textsc{kl}[L_2]^{\nu_{\textsc{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 \textsc{kl}[\sigma.L_3]^{\nu_{\textsc{kl}}} \mid a[\mathsf{nil}]^{\nu_a} \mid b[\mathsf{nil}]^{\nu_b} & \xrightarrow{!\mathsf{auth}_1\triangleright obs}
\end{array}$$

Then m signals the correct reconfiguration based on an old packet. □