A complete and decidable security-specialised logic and its application to the TESLA protocol

Alessio Lomuscio and Bożena Woźna[*]
Department of Computer Science, UCL
Gower Street, London WC1E 6BT,UK
email: {B.Wozna,A.Lomuscio}@cs.ucl.ac.uk

ABSTRACT

We examine a logic to reason about security protocols by means of temporal and epistemic concepts. We report results on completeness and decidability of the formalism as well as its expressiveness. As a case study we apply the formalism in the analysis of TESLA, a secure stream multi-cast protocol.

Categories and Subject Descriptors

F.3.1 [**Specifying and Verifying and Reasoning about Programs**]: Specification techniques; D.2.4 [**Software/Program Verification**]: Model checking; I.2.4 [**Knowledge Representation Formalisms and Methods**]: Modal logic; I.2.11 [**Distributed Artificial Intelligence**]: Multiagent systems

General Terms

Verification, Theory, Security

Keywords

Epistemic logic, security protocols, TESLA

1. INTRODUCTION

There is a long tradition in the use of formal tools (process algebras, theorem proving, model checking, formal logics, etc) to analyse cryptographic protocols. In particular, since the late 80s there has been an attention to modelling security protocols in terms of the evolution of the states of knowledge of the principals. The work on BAN logic [6] was the seminal contribution in this line. BAN logic provides primitives to represent and reason about principals' beliefs, reception of messages, an array of cryptographic-specific concepts such as freshness, keys, etc, as well as axioms and rules tailored to the particular protocol under analysis governing the evolution of these.

It is nowadays customary ([18, 4], etc) to point out that BAN logic lacks several essential technical features (no representation of temporal flows, no well-defined semantics, no models for the intruder, etc) thereby making it impossible to use it consistently to prove that a particular protocol is correct or otherwise. Advances and improvements over the original formulation [1, 12, 15] have not solved this crucial concern.

Given that the emphasis in BAN logic was analysing whether particular protocols established *authentication* of the principals participating in it, one crucial assumption was the benevolence of principals. On the contrary, more recently attention has shifted to analysing whether or not a particular protocol can be exploited by an *adversary*, or *intruder*, achieving a property originally not intended by the protocol designer. In this line perfect encryption is normally assumed, and the intruder follows the Dolev-Yao model [9], i.e. it can read and compose messages but cannot break encrypted messages. This analysis is often carried out in combination with verification techniques, notably model checking, and epistemic notions are not analysed.

Recently, attention has been given again to modelling epistemic states of the principals in a protocol. In [14] a notion of algorithmic knowledge has been proposed. The semantics of the logic assumes that adversaries use algorithms to compute their knowledge. A related paper [20] explores an epistemic logic where the notion of explicit knowledge is linked to deductions in a logical framework. Similarly, in [2] a logic for awareness (originally introduced in [10]) is augmented with security-specialised predicates.

The referenced papers above do not treat temporal concepts explicitly. In [8] a linear time temporal epistemic logic is proposed, where properties of a protocol may be shown via resolution, although an explicit model of the adversary is not provided. In [21] it is shown that the verification problem for a linear temporal epistemic logic augmented with the past operators and security-specific predicates, and *tagged protocols* is decidable.

Many of the papers discussed above [1, 6, 12, 15], do not provide completeness results for the language and/or do not provide a grounded semantics [22] for the modalities. The main aim of the present paper is to propose a decidable security-specialised logic to reason about security concepts, that:

[*]The authors acknowledge support from the EPSRC (grant GR/S49353), the Nuffield Foundation (grant NAL/690/G), and The Royal Society (grant 2004/R3 EU).

- has an intuitive, computationally-grounded semantics,
- offers a model of the adversary,
- enables the representation of epistemic and temporal concepts,
- enjoys soundness and completeness.

The logic presented in this paper is a combination of CTL [7], and a standard epistemic logic [11] augmented to include an additional epistemic operator representing explicit knowledge. It is further shown that the expressivity of the resulting system enables us to represent a non-standard notion of deductive knowledge. In addition we show how to apply the logic to model properties of the TESLA protocol [19], a protocol so far analysed only via process algebras [5] and Lynch-Vaandrager automata [3].

The rest of the paper is organised as follows. In Section 2 we present our security-specialised logic. Section 3 describes the standard Dolev-Yao adversary model in the logic of Section 2. In Section 4 we describe a scheme of the TESLA protocol, and analyse it formally in Section 5. We conclude in Section 6 with some final remarks.

2. A TDL LOGIC

In this section we present the syntax and semantics of a multi-modal temporal epistemic logic with security-specialised primitives, and we show that this logic is finitely axiomatisable. We call the logic *a temporal deductive logic* (*TDL*).

2.1 Syntax

We assume familiarity with the basic concepts of security protocols and we take $\mathbb{K} = \{k_1, k_2, \ldots\}$ to be a set of symmetric and asymmetric keys, $\mathbb{N}$ to be a set of nonces, $\mathbb{T} = \{t_1, t_2, \ldots\}$ to be a set of plain-texts, and $\mathbb{F} = \{f(k) \mid k \in \mathbb{K}$ and $f : \mathbb{K} \to \{0, 1, \ldots\}$ is a pseudo-random function$\}$ to be a set of commitments to keys. The commitment to a key k is an integer value that is computed by applying a pseudo-random function f to key k. It is assumed that f is effectively impossible to invert, so key k cannot be computed from the commitment to k. We assume that keys, nonces, texts, and commitments can be distinguished, that is, $\mathbb{K}$, $\mathbb{N}$, $\mathbb{T}$ and $\mathbb{F}$ are disjoint sets. A set of *messages* $\mathbb{M}$ is defined inductively by the following grammar:

$$m := t \mid k \mid n \mid f(k) \mid m \cdot m \mid \{m\}_k \mid \texttt{MAC}(k, m)$$

where $t \in \mathbb{T}$, $k \in \mathbb{K}$, $n \in \mathbb{N}$, $f(k) \in \mathbb{F}$, m is a generic message, and $\texttt{MAC} : \mathbb{K} \times \mathbb{M} \to \{0, 1, \ldots\}$ is a pseudo-random function, called *message authentication code*. It is assumed that MAC is effectively impossible to invert, so key k cannot be computed from the MAC value.

We write $m \cdot m'$ for the concatenation of m and m', $\{m\}_k$ for encryption of m with the key k, and $\texttt{MAC}(k, m)$ for the message authentication code of m and k. We assume that the set $\mathbb{K}$ is closed under inverses, i.e. for a given key $k \in \mathbb{K}$ there is an inverse key $k^{-1} \in \mathbb{K}$ such that $\{\{m\}_k\}_{k^{-1}} = m$. If the cryptosystem uses symmetric keys, then $k = k^{-1}$; for the public cryptosystem k and k^{-1} are different. We also define a *submessage* binary relation $\sqsubseteq$ on $\mathbb{M}$ as the smallest reflexive and transitive relation satisfying the following conditions: (1) $m \sqsubseteq m \cdot m'$, (2) $m \sqsubseteq m' \cdot m$, (3) $m \sqsubseteq \{m\}_k$.

Let $\mathcal{PV}$ be a set of propositional variables. We assume that $\mathcal{AG} = \{1, \ldots, N\}$ is a finite set of agents. The set $\mathcal{WF}(TDL)$ of well-formed TDL formulas is defined by the following grammar:

$$\varphi := p \mid has_i(m) \mid sent_i(m) \mid received_i(m) \mid faked_i(m) \mid dropped_i(m) \mid \neg\varphi \mid \varphi \vee \varphi \mid \mathrm{E}\bigcirc\varphi \mid \mathrm{E}(\varphi \mathcal{U} \varphi) \mid \mathrm{A}(\varphi \mathcal{U} \varphi) \mid \mathcal{K}_i\varphi \mid \mathcal{X}_i\varphi \mid \mathcal{A}_i\varphi$$

where $p \in \mathcal{PV}$, $i \in \mathcal{AG}$, $m \in \mathbb{M}$. The terms $has_i(m)$, $sent_i(m)$, $received_i(m)$, $dropped_i(m)$, and $faked_i(m)$ are security-specialised propositional variables, which, as one would expect, are read as "*agent i has message m*", "*agent i sent message m*", "*agent i received message m*", "*agent i dropped message m*, and "*agent i faked message m*", respectively. $has_i(m)$ means that agent i has in his explicit possession the message m. What "explicit possession" means depends on the application, the capabilities of the principals, and the protocol they are running. In Section 3 we provide a concrete implementation of the $has_i(m)$ that captures Dolev-Yao capabilities.

We would like to emphasise that the above security-specialised propositional variables are *propositions*, not predicates. This is because we assume that the message m, which is carried by $has_i(m)$, $sent_i(m)$, $received_i(m)$, $dropped_i(m)$, and $faked_i(m)$, is not a value, but rather an intrinsic part of the propositions.

The above syntax extends CTL [7] with a standard epistemic modality $\mathcal{K}_i$ as well as operators for explicit knowledge ($\mathcal{X}_i$) and awareness ($\mathcal{A}_i$) as in [11]. The formula $\mathcal{X}_i\alpha$ is read as "*agent i knows explicitly that* α", $\mathcal{A}_i\alpha$ is read as "*agent i is aware of* α", and $\mathcal{K}_i\alpha$ is read as "*agent i knows (implicitly) that* α". The formula $\mathrm{E}\bigcirc\alpha$ is read as "*there exists a computation path such that at the next step of the path* α *holds*", $\mathrm{E}(\alpha\mathcal{U}\beta)$ is read as "*there exists a computation path such that* β *eventually occurs and* α *continuously holds until then*", and $\mathrm{A}(\alpha\mathcal{U}\beta)$ is read as "*for all computation paths* β *eventually occurs and* α *continuously holds until then*". We shall further use the shortcut $\mathcal{D}_i\alpha$ to represent $\mathrm{E}(\mathcal{K}_i\alpha\mathcal{U}\mathcal{X}_i\alpha)$. The formula $\mathcal{D}_i\varphi$ is read as "*agent i may deduce* α *(by some computational process)*". Note that TDL differs from [16] only in a security-specialised subset of the set $\mathcal{PV}$.

2.2 Interpreted Systems

In this section we will briefly review the multi-agent framework from [11]; especially we will focus on *interpreted systems*, over which a semantics for TDL will be given.

A multi-agent system (MAS) consists of n agents and an environment, each of which is in some particular local state at a given point in time. We assume that an agent's local state encapsulates all the information the agent has access to. In the security settings, the local state of an agent might include some initial information about keys, the messages that the agent has sent and received so far, and so on. The local states of the environment describe information that is relevant to the system but that is not included in any local agent's state. Note also that the environment can be viewed as just another agent, as we will do here.

A multi-agent system is not a static entity. Its computations are usually defined by means of runs (see [11]), where a *run* is a function from the natural numbers to the set of all the possible global states. Thus, in these settings, an *interpreted system* for a multi-agent system is defined as a set of runs together with a valuation function for the propositional variables of the language under consideration. More formally, let each agent $i \in \mathcal{AG}$ be associated with a set of local states L_i, and the environment be associated with a set of local states L_e. Then, an *interpreted system* is a tuple $IS = (S, T, \sim_1, \ldots, \sim_n, \mathcal{V})$, where $S \subseteq \prod_{i=1}^{n} L_i \times L_e$ is a set of global states; $T \subseteq S \times S$ is a serial (temporal) relation on S; for each agent $i \in \mathcal{AG}$, $\sim_i \subseteq S \times S$ is an equivalence (epistemic) relation defined by: $s \sim_i s'$ iff $l_i(s') = l_i(s)$, where $l_i : S \rightarrow L_i$ is a function that returns the local state of agent i from a global state; $\mathcal{V} : S \longrightarrow 2^{\mathcal{PV}}$ is a valuation function. $\mathcal{V}$ assigns to each state a set of proposition variables that are assumed to be true at that state. For more details we refer to [11].

In order to give a semantics to TDL we extend the above definition by means of local awareness functions, used to indicate the facts that agents are aware of. As in [11], we are not very specific with the notion of awareness, but by being aware we intutively mean "*to be able to figure out the truth*", or "*to be able to compute the truth within time T*", etc.

Definition 1 (Model). *Given $\mathcal{AG} = \{1, \ldots, n\}$, a* model *is a tuple $M = (S, T, \sim_1, \ldots, \sim_n, \mathcal{V}, \mathbb{A}_1, \ldots, \mathbb{A}_n)$, where S, T, $\sim_i$, and $\mathcal{V}$ are defined as above, and $\mathbb{A}_i : L_i \longrightarrow 2^{\mathcal{WF}(TDL)}$ is an awareness function assigning a set of formulas to each state, for each $i \in \mathcal{AG}$.*

Note that the set of formulas that the agent is aware of can be arbitrary and may not be closed under sub-formulas. Note also that the definition of the model is an extension of the awareness structure (see [11]) by a temporal relation. Moreover, it restricts the standard awareness function to be defined over local states only.

In the paper we focus on a specific class of multi-agent systems, appropriate to modelling security protocols. These are message-passing systems in which one or more of the agents is an adversary controling the communication channel.

We assume that the local state of an agent is a sequence of events of the form $(e_0, \ldots, e_m)$, where e_0 is the initial event, and for $i \in \{1, \ldots, m\}$, e_i is a term of the form $sent(i, m)$ or $recv(m)$, where m is a message and i is an agent. The term $sent(i, m)$ stands for the agent has sent message m to agent i. Similarly the term $recv(m)$ represents that the agent has received message m. Note that in $recv(m)$ the sender is not specified. This is because the receiver will not in general be able to determine the sender of a message he has received.

2.3 Satisfaction

A *path* in M is an infinite sequence $\pi = (s_0, s_1, \ldots)$ of global states such that $(s_i, s_{i+1}) \in T$ for each $i \in \mathbb{N}$. For a path $\pi = (s_0, s_1, \ldots)$, we take $\pi(k) = s_k$. By $\Pi(s)$ we denote the set of all the paths starting at $s \in S$.

Definition 2 (Satisfaction). *Let M be a model, s a state, and α, β TDL formulas. The satisfaction relation $\models$, indicating truth of a formula in model M at state s, is defined inductively as follows:*

$(M, s) \models p$ *iff* $p \in \mathcal{V}(s)$,
$(M, s) \models \alpha \vee \beta$ *iff* $(M, s) \models \alpha$ *or* $(M, s) \models \beta$,
$(M, s) \models \neg\alpha$ *iff* $(M, s) \not\models \alpha$,
$(M, s) \models \mathrm{E}\bigcirc\alpha$ *iff* $(\exists\pi \in \Pi(s))(M, \pi(1)) \models \alpha$,
$(M, s) \models \mathrm{E}(\alpha\mathcal{U}\beta)$ *iff* $(\exists\pi \in \Pi(s))(\exists m \geq 0)[(M, \pi(m)) \models \beta$ *and* $(\forall j < m)(M, \pi(j)) \models \alpha]$,
$(M, s) \models \mathrm{A}(\alpha\mathcal{U}\beta)$ *iff* $(\forall\pi \in \Pi(s))(\exists m \geq 0)[(M, \pi(m)) \models \beta$ *and* $(\forall j < m)(M, \pi(j)) \models \alpha]$,
$(M, s) \models \mathcal{A}_i\alpha$ *iff* $\alpha \in \mathbb{A}_i(l_i(s))$,
$(M, s) \models \mathcal{K}_i\alpha$ *iff* $(\forall s' \in S)$ $(s \sim_i s'$ *implies* $(M, s') \models \alpha)$,
$(M, s) \models \mathcal{X}_i\alpha$ *iff* $(M, s) \models \mathcal{K}_i\alpha$ *and* $(M, s) \models \mathcal{A}_i(\alpha)$.

Henceforth, we will only consider models with a fixed interpretation for the security-specialised propositional variables $sent_i(m)$ and $received_i(m)$; in particular, we take $\models$ to be defined for these propositions as follows:

$(M, s) \models sent_i(m)$ iff $(\exists m' \in \mathbb{M})(\exists j \in \mathcal{AG})$ such that $m \sqsubseteq m'$ and $sent(j, m') \in l_i(s)$,
$(M, s) \models received_i(m)$ iff $recv(m) \in l_i(s)$.

We leave definitions of the other security-specialised propositions open. Namely, for distinct protocols these propositions will be defined differently.

Note that since $\mathcal{D}_i\alpha$ is a shortcut for $\mathrm{E}(\mathcal{K}_i\alpha\mathcal{U}\mathcal{X}_i\alpha)$, as defined on page 2, we have that $(M, s) \models \mathcal{D}_i\alpha$ iff $(M, s) \models \mathrm{E}(\mathcal{K}_i\alpha\mathcal{U}\mathcal{X}_i\alpha)$. We would like to emphasise that satisfaction for $\mathcal{X}_i$ can be defined using only the $\sim_i$ relation and the $\mathbb{A}_i$ function, but we will find it convenient in the axiomatisation to have a dedicated operator $\mathcal{A}_i$ for awareness of agent i. This is in line with [11]. Note also that the definition of the satisfaction relation is quite general; in particular, it contains nothing that is specific to security protocols, except the interpretation of the security-specialised propositional variables.

The remaining operators can be introduced as abbreviations in the usual way, i.e., $\alpha \wedge \beta \overset{def}{=} \neg(\neg\alpha \vee \neg\beta)$, $\alpha \Rightarrow \beta \overset{def}{=} \neg\alpha \vee \beta$, $\alpha \Leftrightarrow \beta \overset{def}{=} (\alpha \Rightarrow \beta) \wedge (\beta \Rightarrow \alpha)$, $\mathrm{A}\bigcirc\alpha \overset{def}{=} \neg\mathrm{E}\bigcirc\neg\alpha$, $\mathrm{E}\Diamond\alpha \overset{def}{=} \mathrm{E}(\top\mathcal{U}\alpha)$, $\mathrm{A}\Diamond\alpha \overset{def}{=} \mathrm{A}(\top\mathcal{U}\alpha)$, $\mathrm{E}\Box\alpha \overset{def}{=} \neg\mathrm{A}\Diamond\neg\alpha$, $\mathrm{A}\Box\alpha \overset{def}{=} \neg\mathrm{E}\Diamond\neg\alpha$, $\mathrm{A}(\alpha\mathcal{W}\beta) \overset{def}{=} \neg\mathrm{E}(\neg\alpha\mathcal{U}\neg\beta)$, $\mathrm{E}(\alpha\mathcal{W}\beta) \overset{def}{=} \neg\mathrm{A}(\neg\alpha\mathcal{U}\neg\beta)$.

Let M be a model. We say that a TDL formula φ is *valid in M* (written $M \models \varphi$), if $M, s \models \varphi$ for all states $s \in S$, and that a TDL formula φ is *valid* (written $\models \varphi$), if φ is valid in all the models M.

2.4 A Complete Axiomatic System for TDL

An *axiomatic system* consists of a collection of *axioms* and *inference rules*. An axiom is a formula, and an inference rule has the form "from previously derived formulas $\varphi_1, \ldots, \varphi_m$ infer formula φ". We say that φ is *provable* (written $\vdash \varphi$) if there is a sequence of formulas ending with φ, such that each formula is either an instance of an axiom, or follows from other provable formulas by applying an inference rule. An axiom system is said to be *sound* with respect to the models defined as in Definition 1, if $\vdash \varphi$ implies $\models \varphi$. An axiom system is said to be *complete* with respect to the models defined as in Definition 1, if $\models \varphi$ implies $\vdash \varphi$.

Let $i \in \{1, \ldots, n\}$. Consider an axiom system for TDL as defined below:

PC. All substitution instances of classical tautologies.
T1. $\mathrm{E}\bigcirc\top$ T2. $\mathrm{E}\bigcirc(\alpha \vee \beta) \Leftrightarrow \mathrm{E}\bigcirc\alpha \vee \mathrm{E}\bigcirc\beta$
T3. $\mathrm{E}(\alpha\mathcal{U}\beta) \Leftrightarrow \beta \vee (\alpha \wedge \mathrm{E}\bigcirc\mathrm{E}(\alpha\mathcal{U}\beta))$
T4. $\mathrm{A}(\alpha\mathcal{U}\beta) \Leftrightarrow \beta \vee (\alpha \wedge \mathrm{A}\bigcirc\mathrm{A}(\alpha\mathcal{U}\beta))$
K1. $(\mathcal{K}_i\alpha \wedge \mathcal{K}_i(\alpha \Rightarrow \beta)) \Rightarrow \mathcal{K}_i\beta$ K2. $\mathcal{K}_i\alpha \Rightarrow \alpha$
K3. $\neg\mathcal{K}_i\alpha \Rightarrow \mathcal{K}_i\neg\mathcal{K}_i\alpha$ X1. $\mathcal{X}_i\alpha \Leftrightarrow \mathcal{K}_i\alpha \wedge \mathcal{A}_i\alpha$
A1. $\mathcal{A}_i\alpha \Rightarrow \mathcal{K}_i\mathcal{A}_i\alpha$ A2. $\neg\mathcal{A}_i\alpha \Rightarrow \mathcal{K}_i\neg\mathcal{A}_i\alpha$
R1. From α and $\alpha \Rightarrow \beta$ infer β
R2. From α infer $\mathcal{K}_i\alpha$, $i = 1, \ldots, n$
R3. From $\alpha \Rightarrow \beta$ infer $\mathrm{E}\bigcirc\alpha \Rightarrow \mathrm{E}\bigcirc\beta$
R4. From $\gamma \Rightarrow (\neg\beta \wedge \mathrm{E}\bigcirc\gamma)$ infer $\gamma \Rightarrow \neg\mathrm{A}(\alpha\mathcal{U}\beta)$
R5. From $\gamma \Rightarrow (\neg\beta \wedge \mathrm{A}\bigcirc(\gamma \vee \neg\mathrm{E}(\alpha\mathcal{U}\beta)))$ infer $\gamma \Rightarrow \neg\mathrm{E}(\alpha\mathcal{U}\beta)$

Theorem 1 ([16]). *The system TDL is sound and complete, i.e. $\models \varphi$ iff $\vdash \varphi$, for any formula $\varphi \in \mathcal{WF}(TDL)$.*

3. THE DOLEV-YAO ADVERSARY

The Dolev-Yao adversary [9] can compose messages, replay them, or decipher them if he has the right keys. In the paper we use a formalisation of the Dolev-Yao adversary given in [14]. Namely, we define a derivation relation $H \vdash_{DY} m$ between a set of messages H and a message m. Intuitively, $\vdash_{DY}$ means that an adversary can "extract" message m from a set H of received messages and keys, using the admissible operations.

The derivation relation $\vdash_{DY}$ is defined as follows:

- if $m \in H$, then $H \vdash_{DY} m$
- if $H \vdash_{DY} \{m\}_k$ and $H \vdash_{DY} k^{-1}$, then $H \vdash_{DY} m$
- if $H \vdash_{DY} m \cdot m'$, then $H \vdash_{DY} m$
- if $H \vdash_{DY} m \cdot m'$, then $H \vdash_{DY} m'$

In our framework, to capture the capabilities of the Dolev-Yao adversary, we specify how the adversary can extract a message, by defining an awareness function $\mathbb{A}_i^{DY}$ for the intruder i, and by introducing the following fixed interpretation for the proposition $has_i(m)$:

$$M, s \models has_i(m) \text{ iff there exists } m' \text{ such that } m \sqsubseteq m' \text{ and } recv(m') \in l_i(s).$$

The function $\mathbb{A}_i^{DY} : L_i \to 2^{\mathcal{WF}(TDL)}$ is defined by: $has_i(m) \in \mathbb{A}_i^{DY}(l)$ iff one of the following conditions holds:

- $recv(m) \in l$.
- $m \in e_0(l)$, where the function e_0 returns the set of messages contained in the initial event of the local state l.
- there exists m' such that $has_i(m \cdot m') \in \mathbb{A}_i^{DY}(l)$ or $has_i(m' \cdot m) \in \mathbb{A}_i^{DY}(l)$.
- there exists an asymmetric key k such that $has_i(\{m\}_k) \in \mathbb{A}_i^{DY}(l)$ and $has_i(k^{-1}) \in \mathbb{A}_i^{DY}(l)$.
- there exists a symmetric key k such that $has_i(\{m\}_k) \in \mathbb{A}_i^{DY}(l)$ and $has_i(k) \in \mathbb{A}_i^{DY}(l)$.
- $m = m_1 \cdot m_2$ and $has_i(m_1) \in \mathbb{A}_i^{DY}(l)$ and $has_i(m_2) \in \mathbb{A}_i^{DY}(l)$.
- $m = \{m_1\}_k$ and $has_i(m_1) \in \mathbb{A}_i^{DY}(l)$ and $has_i(k) \in \mathbb{A}_i^{DY}(l)$.

We can now show that using $\mathbb{A}_i^{DY}$ an adversary can check that he explicitly knows $has_i(m)$ in a state s iff m can be derived from the messages received in the states s together with the messages initially known via the relation $\vdash_{DY}$.

LEMMA 1. *Let $M = (S, T, \sim_1, \ldots, \sim_n, \mathcal{V}, \mathbb{A}_1, \ldots, \mathbb{A}_n)$ be a model such that $\mathbb{A}_i = \mathbb{A}_i^{DY}$. Then, $M, s \models \mathcal{X}_i(has_i(m))$ iff $\{m' \mid recv(m') \in l_i(s)\} \cup e_0(l_i(s)) \vdash_{DY} m$.*

Proof. (Left to right). Assume $M, s \models \mathcal{X}_i(has_i(m))$ hold. By the definition of $\models$, we have that $M, s \models \mathcal{K}_i(has_i(m))$ and $has_i(m) \in \mathbb{A}_i(l_i(s))$. Since $has_i(m) \in \mathbb{A}_i(l_i(s))$, by the definition of $\mathbb{A}_i^{DY}$ we have that either $m \in \{m' \mid recv(m') \in l_i(s)\} \cup e_0(l_i(s))$, or there exists m'' such that $m \sqsubseteq m''$ and $m'' \in \{m' \mid recv(m') \in l_i(s)\} \cup e_0(l_i(s))$. This implies that $\{m' \mid recv(m') \in l_i(s)\} \cup e_0(l_i(s)) \vdash_{DY} m$.
(Right to left). Assume $\{m' \mid recv(m') \in l_i(s)\} \cup e_0(l_i(s)) \vdash_{DY} m$ hold. Then, we have that either $recv(m) \in l_i(s)$ or there exists m' such that $m \sqsubseteq m'$ and $recv(m') \in l_i(s)$. So, by the definition of $\mathbb{A}_i^{DY}$ we have that $has_i(m) \in \mathbb{A}_i^{DY}(l_i(s))$. Moreover, by the definition of $\models$ we have that $M, s \models has_i(m)$. Since $\mathbb{A}_i^{DY}$ is defined over the local states of agent i, we have that for all $t \in S$ such that $s \sim_i t$, $M, t \models has_i(m)$ holds. Hence, by the definition of $\models$ we have that $M, s \models \mathcal{K}_i(has_i(m))$. Hence, we conclude that $M, s \models \mathcal{X}_i(has_i(m))$.
□

A model of the Dolev-Yao adversary was already considered in the interpreted system framework in [14, 20]. However, in those papers, the notion of *algorithmic knowledge* was used to capture the capabilities of the Dolev-Yao adversary instead.

4. TESLA

The *timed efficient stream loss-tolerant authentication* (TESLA) protocol was developed by A. Perrig et al. [19]. It was designed to enable authentication of a continuous stream of packets over an unreliable channel. The developers of TESLA proposed five variants of the protocol; only the first of these is discussed here.

Every variant of the TESLA protocol assumes a single sender broadcasting a continuous stream of packets. Since receivers act independently of one another, only one receiver will be considered below. TESLA uses cryptographic primitives like *commitments* to keys and MAC values. Recall that the commitment to a key k is computed by applying a pseudo-random function f to k, and the MAC value is computed by applying a *message authentication code* function MAC to a part of the packet content and a key. It is assumed that both the sender and the receiver know the pseudo-random function f, and the message authentication code function MAC. With the exception of the two initial packets, each TESLA packet (in the variant we analyse here) contains: (1) the message to be delivered; (2) a *commitment* to the key to be used to encode the MAC of the next packets; (3) the key that was used to encode the MAC of the previous sent packet; (4) the MAC of the current packet.

We now introduce a TESLA variant assuming that the protocol uses one pseudo-random function only. We have introduced such a simplification, because in our opinion the extra pseudo-random function does not provide any additional security; the same assumption is made in [3, 5].

Let **S** stand for the sender, **R** stand for the receiver, and $[x, y]$ denote the concatenation of x and y. Moreover, assume that **S** has a digital signature key pair, with private key $k_{\mathbf{S}}^{-1}$ and public key $k_{\mathbf{S}}$ known to **R**. Further, assume that **R** chooses a random and unpredictable nonce. Then, the initial n steps, for $n > 1$, of the protocol for one sender and one receiver are the following:

(-1) $\mathbf{R} \to \mathbf{S} : n_{\mathbf{R}}$
(0) $\mathbf{S} \to \mathbf{R} : \{f(k_1), n_{\mathbf{R}}\}_{k_{\mathbf{S}}^{-1}}$
(1) $\mathbf{S} \to \mathbf{R} : [P_1, \mathtt{MAC}(k_1, P_1)]$, for $P_1 = [t_1, f(k_2)]$
(2) $\mathbf{S} \to \mathbf{R} : [P_2, \mathtt{MAC}(k_2, P_2)]$, for $P_2 = [t_2, f(k_3), k_1]$
...
(n) $\mathbf{S} \to \mathbf{R} : [P_n, \mathtt{MAC}(k_n, P_n)]$, for $P_n = [t_n, f(k_{n+1}), k_{n-1}]$

In the first step (i.e. step (-1.)) **R** sends a nonce $n_{\mathbf{R}}$ to **S** in order to ensure freshness of the session. In step (0), **S** encodes with $k_{\mathbf{S}}^{-1}$ the pair composed by the nonce just received and the commitment to key k_1 (i.e. $f(k_1)$), produced by the agreed pseudo-random function f. This is sent to **R**. **R** knows $k_{\mathbf{S}}$, so he can decode the message $\{f(k_1), n_{\mathbf{R}}\}_{k_{\mathbf{S}}^{-1}}$. Given this, **R** knows that commitment $f(k_1)$ is authentic; however, because he does not know the reverse of f, he cannot compute the value of the key k_1. In step (1), **S** transmits in clear the first packet P_1 together with its MAC, i.e. the value $\mathtt{MAC}(k_1, P_1)$. P_1 contains message t_1 and commitment to key k_2 only. k_2 is used to compute the MAC of the next packet. After this step **R** knows the content of t_1, but he cannot be sure of its authenticity. This is because he does not know the value of the key k_1, and so he cannot check the MAC value. The authentication of packet P_1 is done by the sender's disclosing of k_1 in step (2). In steps (2),...,(n) **S** sends packet P_i, for $i \in \{1, \ldots, n\}$, authenticated by the MAC value computed from key k_i and P_i (i.e. the value $\mathtt{MAC}(k_i, P_i)$). P_i contains message t_i, commitment to key k_{i+1} that is computed by applying pseudo-random function f to k_{i+1} , and key k_{i-1} thereby enabling $\mathbf{R}$ to verify the commitment $f(k_{i-1})$ and the MAC of packet P_i.

The TESLA scheme is susceptible to packet loss. In particular, once a packet is dropped no further packets can be authenticated; in the next section we will formally show that TESLA satisfies this property. Moreover, the scheme can be subverted if an attacker gets packet P_{i+1} before the receiver gets P_i. This is because the attacker would then know the secret key k_i that is used to compute the MAC value of P_i, and use this to forge all subsequent traffic by impersonating $\mathbf{S}$. To prevent this attack, the TESLA participants are initially synchronised, they know the precise schedule of packets, and $\mathbf{S}$ has to send packets at regular intervals that were agreed with $\mathbf{R}$ during the synchronisation process. More details are in [19]. This is not essential for our analysis.

TESLA guarantees the following security property: "*the receiver does not accept as authentic any message unless it was actually sent by the sender*". A streaming scheme that provides this guarantee is usually called a *secure stream authentication scheme* [19]. The main aim of the rest of the paper is to analyse this statement formally.

5. MODELLING OF TESLA IN TDL

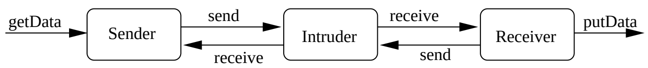

Figure 1: The Dolev-Yao assumption

So far, the TESLA protocol has been modelled and checked via process algebras [5] and Lynch-Vaandrager automata [3] only. Here, we are interested in applying the interpreted systems framework and the TDL language, which has a computationally-grounded semantics, to perform a *semantic* analysis of this protocol. So, let us begin by building a model $M = (S, T, \sim_{\mathbf{S}}, \sim_{\mathbf{R}}, \sim_{\mathbf{I}}, \mathcal{V}, \mathbb{A}_{\mathbf{S}}, \mathbb{A}_{\mathbf{R}}, \mathbb{A}_{\mathbf{I}})$ for TESLA. In order to perform this task, we have to provide definitions of the set of reachable global states S, the transition relation T, the valuation function $\mathcal{V}$, and the awareness functions $\mathbb{A}_{\mathbf{S}}$, $\mathbb{A}_{\mathbf{R}}$ and $\mathbb{A}_{\mathbf{I}}$.

There are three active components in the TESLA protocol: a sender, a receiver, and an intruder. In the formalism of interpreted systems, it is convenient to see the sender and the receiver as agents, and the intruder as the environment. We assume that the communication channel is under complete control of the intruder, hence all messages from the sender to the receiver and vice versa go through the intruder (see Figure 1). The intruder can overhear, capture, drop, resend, delay and fake messages. We do not allow the intruder to encrypt and decrypt messages unless he has the appropriate key, and we force the intruder to send well-formed packets only. Namely, a packet sent (re-sent, faked) by the intruder has to contain a message body, a key commitment, a key, and a MAC value computed from the packet and some key. Moreover, we assume that the sender, the receiver, and the intruder use a shared pseudo-random function f and a shared message authentication code function MAC. Namely, they can all compute $f(m)$ and $\mathtt{MAC}(m,n)$, if they have m and n. Further, we assume that the receiver and the intruder know the public key of the sender, and that the sender and the intruder start off with disjoint sets of keys.

Each of the principals as well as the intruder can be modelled by considering their local states. Before we define them, we first introduce the following notion. For $t_i \in \mathbb{M}$, $f(k_{i+1}) \in \mathbb{F}$, $k_i \in \mathbb{K}$, $n_{\mathbf{R}} \in \mathbb{N}$ we define a message P_i, called a *packet*, as follows:

$$P_i = \begin{cases} (t_i \cdot f(k_{i+1}) \cdot k_{i-1}) \cdot \mathtt{MAC}(k_i, (t_i \cdot f(k_{i+1}) \cdot k_{i-1})), \\ \text{if } i > 1; \\ (t_i \cdot f(k_{i+1})) \cdot \mathtt{MAC}(k_i, (t_i \cdot f(k_{i+1}))), \quad \text{if } i = 1; \\ \{f(k_{i+1}) \cdot n_{\mathbf{R}}\}_{k_{\mathbf{S}}^{-1}}, \quad \text{if } i = 0. \end{cases}$$

We assume that the receiver knows the precise sending schedule of packets, and that this information is incorporated into packet P_0. Notice, that from this assumption it follows that P_0 always has to arrive to the receiver safely for the protocol to function. Further, we assume the sender and the receiver to keep the three most recent packets, and the intruder to keep five.

For the sender $\mathbf{S}$ we take the following set of local states:

$$\begin{gathered} L_{\mathbf{S}} = \{[\cdot], [recv(n_{\mathbf{R}})], [sent(\mathbf{R}, P_0)]\} \cup, \\ \{[sent(\mathbf{R}, P_{i-1}), sent(\mathbf{R}, P_i)] \mid i > 0\} \cup \\ \{[sent(\mathbf{R}, P_{i-1}), sent(\mathbf{R}, P_i), sent(\mathbf{R}, P_{i+1})] \mid i > 0\} \end{gathered}$$

$[\cdot]$ represents $\mathbf{S}$'s initial state in the protocol. In any state, we assume that $\mathbf{S}$ has all the information he needs to prepare a packet, i.e. he has a complete set of messages $M_{\mathbf{S}} \subseteq \mathbb{M}$. Further, we assume that $M_{\mathbf{S}}$ constitutes $\mathbf{S}$'s initial database that remains accessible to him throughout the run. For convenience we do not add this explicitly to the local states. The local state $[recv(n_{\mathbf{R}})]$ represents the message from $\mathbf{R}$ to establish communication, whereas $[sent(\mathbf{R}, P_0)]$ represents the state in which $\mathbf{S}$ has just sent P_0 to $\mathbf{R}$. $[sent(\mathbf{R}, P_{i-1})$, $sent(\mathbf{R}, P_i)]$ and $[sent(\mathbf{R}, P_{i-1})$, $sent(\mathbf{R}, P_i)$, $sent(\mathbf{R}, P_{i+1})]$ represent states, in which $\mathbf{S}$ has sent packets P_j, where $j \leq i+1$ and $i > 0$.

We take the following set of local states for the receiver $\mathbf{R}$:

$$\begin{gathered} L_{\mathbf{R}} = \{[\cdot], [sent(\mathbf{S}, n_{\mathbf{R}})], [stop], [recv(P_0)]\} \cup \\ \{[recv(P_0), recv(P_2)]\} \cup \{[recv(P_i), recv(P_{i+1})] \mid i \geq 0\} \cup \\ \{[recv(P_{i-1}), recv(P_i), recv(P_{i+1})] \mid i > 0\} \cup \\ \{[recv(P_{i-1}), recv(P_i), recv(P_{i+2})] \mid i > 0\} \cup \\ \{[recv(P_i), recv(P_{i+1}), recv(P'_{i+2})] \mid i \geq 0\} \cup \\ \{[recv(P_0), recv(P'_1)]\} \cup \{[recv(P_0), recv(P'_1), recv(P'_2)]\} \\ \cup \{[recv(P_0), recv(P'_1), recv(P_2)]\} \end{gathered}$$

$[\cdot]$ represents $\mathbf{R}$'s initial state in the protocol, in which $\mathbf{R}$ has his set of nonces only. $[sent(\mathbf{S}, n_{\mathbf{R}})]$ represents the local state in which $\mathbf{R}$ has just sent the nonce $n_{\mathbf{R}}$ to $\mathbf{S}$, and he is waiting for packets. $[stop]$ represents the local state, in which $\mathbf{R}$ has just stopped collecting packets. The local states $[recv(P_0)]$, $[recv(P_0), recv(P_2)]$, $[recv(P_i)$, $recv(P_{i+1})]$ for $i \geq 0$, $[recv(P_{i-1}), recv(P_i)$, $recv(P_{i+2})]$ and $[recv(P_{i-1})$, $recv(P_i), recv(P_{i+1})]$ for $i > 0$ represent the packets $\mathbf{R}$ has received from $\mathbf{S}$, whereas $[recv(P_0)$, $recv(P'_1)]$, $[recv(P_0)$, $recv(P'_1), recv(P'_2)]$, $[recv(P_0), recv(P'_1)$, $recv(P_2)]$, and $[recv(P_i), recv(P_{i+1}), recv(P'_{i+2})]$ for $i \geq 0$ represent the faked packets $\mathbf{R}$ has received. Specifically, these states represent situations in which $\mathbf{I}$ has produced a new packet, say P'_i, and has sent it to $\mathbf{R}$ instead of forwarding the intercepted packet P_i.

It remains to model the intruder's set of local states. For $\mathbf{I}$ we take the following set:

$$\begin{gathered} L_{\mathbf{I}} = \{[\cdot], [recv(n_{\mathbf{R}})], [recv(P_0)]\} \cup \\ \{[recv(P_i), recv(P_{i+1})] \mid i \geq 0\} \cup \\ \{[recv(P_{i-1}), recv(P_i), recv(P_{i+1})] \mid i > 0\} \cup \end{gathered}$$

$$\{[recv(P_0), recv(P_1), send(\mathbf{R}, P_1')]\} \cup$$
$$\{[recv(P_0), recv(P_1), send(\mathbf{R}, P_1'), recv(P_2)]\} \cup$$
$$\{[recv(P_0), recv(P_1), send(\mathbf{R}, P_1'), recv(P_2), send(\mathbf{R}, P_2')]\} \cup$$
$$\{[recv(P_{i-1}), recv(P_i), recv(P_{i+1}), send(\mathbf{R}, P_{i+1}')] \mid i > 0\}$$

$[\cdot]$ represents $\mathbf{I}$'s initial state in the protocol. In any state, we assume that $\mathbf{I}$ has all the information needed to prepare well-formed packets, with $M_{\mathbf{I}} \subseteq \mathbb{M}$ such that $M_{\mathbf{I}} \cap M_{\mathbf{S}} = \emptyset$. Further, we assume that $M_{\mathbf{I}}$ can grow during the run. $[recv(n_{\mathbf{R}})]$ represents $\mathbf{I}$'s state following the interception of $\mathbf{R}$'s initial message to $\mathbf{S}$. $[recv(P_0)]$, $[recv(P_i), recv(P_{i+1})]$ and $[recv(P_{i-1}), recv(P_i), recv(P_{i+1})]$ represent the packets intercepted by $\mathbf{I}$ sent by $\mathbf{S}$ to $\mathbf{R}$. $[recv(P_0), recv(P_1), send(\mathbf{R}, P_1')]$, $[recv(P_0), recv(P_1), send(\mathbf{R}, P_1'), recv(P_2)]$, $[recv(P_0), recv(P_1), send(\mathbf{R}, P_1'), recv(P_2), send(\mathbf{R}, P_2')]$, and $[recv(P_{i-1}), recv(P_i), recv(P_{i+1}), send(\mathbf{R}, P_{i+1}')]$ for $i > 0$, represent the packets intercepted by $\mathbf{I}$ sent by $\mathbf{S}$ to $\mathbf{R}$ and their faked versions.

Given the sets of local states for the TESLA agents, i.e. the sets $L_{\mathbf{S}}$, $L_{\mathbf{R}}$ and $L_{\mathbf{I}}$, the following sets of actions are available to the agents:

- $Act_{\mathbf{S}} = \{sendP_i \mid i \geq 0\} \cup \{acceptP_i \mid i \geq 0\} \cup \{\lambda\}$,
- $Act_{\mathbf{R}} = \{\lambda, nonce, stop\} \cup \{acceptP_i \mid i > 0\}$,
- $Act_{\mathbf{I}} = \{\lambda\} \cup \{dropP_i \mid i > 0\} \cup \{fakeP_i \mid i > 0\} \cup \{acceptP_i \mid i > 0\}$,

where λ stands for no action.

The protocols executed by agents are defined as follows:

Sender:

- $P_{\mathbf{S}}([\cdot]) = \{\lambda\}$
- $P_{\mathbf{S}}([recv(n_{\mathbf{R}})]) = \{sendP_0\}$
- $P_{\mathbf{S}}([sent(\mathbf{R}, P_0)]) = \{sendP_1\}$
- $P_{\mathbf{S}}([sent(\mathbf{R}, P_{i-1}), sent(\mathbf{R}, P_i)]) = \{sendP_{i+1}\}$,
- $P_{\mathbf{S}}([sent(\mathbf{R}, P_{i-1}), sent(\mathbf{R}, P_i), sent(\mathbf{R}, P_{i+1})]) = \{acceptP_{i-1}, \lambda\}$, for $i \geq 0$.

Intruder:

- $P_{\mathbf{I}}([\cdot]) = P_{\mathbf{I}}([recv(n_{\mathbf{R}})]) = \{\lambda\}$
- $P_{\mathbf{I}}([recv(P_0)]) = \{\lambda, dropP_1, fakeP_1\}$
- $P_{\mathbf{I}}([recv(P_i), recv(P_{i+1})]) = \{\lambda, dropP_{i+2}, fakeP_{i+2}\}$, for $i \geq 0$
- $P_{\mathbf{I}}([recv(P_{i-1}), recv(P_i), recv(P_{i+1})]) = \{acceptP_{i-1}, \lambda\}$, for $i > 0$
- $P_{\mathbf{I}}([recv(P_0), recv(P_1), send(\mathbf{R}, P_1')]) = \{fakeP_2\}$,
- $P_{\mathbf{I}}([recv(P_0), recv(P_1), send(\mathbf{R}, P_1'), recv(P_2)]) = \{\lambda\}$,
- $P_{\mathbf{I}}([recv(P_0), recv(P_1), send(\mathbf{R}, P_1'), recv(P_2), send(\mathbf{R}, P_2')]) = \{\lambda\}$,
- $P_{\mathbf{I}}([recv(P_{i-1}), recv(P_i), recv(P_{i+1}), send(\mathbf{R}, P_{i+1}')]) = \{\lambda\}$, for $i > 0$

Receiver:

- $P_{\mathbf{R}}([\cdot]) = \{nonce\}$
- $P_{\mathbf{R}}([sent(\mathbf{S}, n_{\mathbf{R}})]) = P_{\mathbf{R}}([recv(P_0)]) = \{\lambda\}$
- $P_{\mathbf{R}}([recv(P_i), recv(P_{i+1})]) = \{\lambda\}$, for $i \geq 0$
- $P_{\mathbf{R}}([recv(P_{i-1}), recv(P_i), recv(P_{i+1})]) = \{acceptP_i\}$, for $i > 0$
- $P_{\mathbf{R}}([stop]) = P_{\mathbf{R}}([recv(P_0), recv(P_2)]) = \{stop\}$,
- $P_{\mathbf{R}}([recv(P_{i-1}), recv(P_i), recv(P_{i+2})]) = \{stop\}$, for $i > 0$
- $P_{\mathbf{R}}([recv(P_0), recv(P_1')]) = \{\lambda\}$,
- $P_{\mathbf{R}}([recv(P_0), recv(P_1'), recv(P_2')]) = \{stop\}$,
- $P_{\mathbf{R}}([recv(P_0), recv(P_1'), recv(P_2)]) = \{stop\}$,
- $P_{\mathbf{R}}([recv(P_{i-1}), recv(P_i), recv(P_{i+1}')]) = \{stop\}$, for $i > 0$

The evolution of the system during the execution of TESLA is defined by means of an evolution function $t : (L_{\mathbf{S}} \times L_{\mathbf{R}} \times L_{\mathbf{I}}) \times Act \rightarrow 2^{L_{\mathbf{S}} \times L_{\mathbf{R}} \times L_{\mathbf{I}}}$, where Act is a subset of $Act_{\mathbf{S}} \times Act_{\mathbf{R}} \times Act_{\mathbf{I}}$. More precisely, let us assume that the system starts from the initial state $([\cdot], [\cdot], [\cdot])$, and let rv denote the even $recv$, and st the even $sent$. Then, the evolution function for TESLA is defined as follows:

Intruder forwards all packets

1. $t(([\cdot], [\cdot], [\cdot]]), (\lambda, nonce, \lambda)) = ([rv(n_{\mathbf{R}})], [st(\mathbf{S}, n_{\mathbf{R}})], [rv(n_{\mathbf{R}})]);$
2. $t(([rv(n_{\mathbf{R}})], [st(\mathbf{S}, n_{\mathbf{R}})], [rv(n_{\mathbf{R}})]), (sendP_0, \lambda, \lambda))$
   $= ([st(\mathbf{R}, P_0)], [rv(P_0)], [rv(P_0)]);$
3. $t(([st(\mathbf{R}, P_0)], [rv(P_0)], [rv(P_0)]), (sendP_1, \lambda, \lambda)) =$
   $([st(\mathbf{R}, P_0), st(\mathbf{R}, P_1)], [rv(P_0), rv(P_1)], [rv(P_0), rv(P_1)]);$
4. $t(([st(\mathbf{R}, P_0), st(\mathbf{R}, P_1)], [rv(P_0), rv(P_1)], [rv(P_0), rv(P_1)]),$
   $(sendP_2, \lambda, \lambda)) = ([st(\mathbf{R}, P_0), st(\mathbf{R}, P_1), st(\mathbf{R}, P_2)],$
   $[rv(P_0), rv(P_1), rv(P_2)], [rv(P_0), rv(P_1), rv(P_2)]);$
5. $t(([st(\mathbf{R}, P_0), st(\mathbf{R}, P_1), st(\mathbf{R}, P_2)], [rv(P_0), rv(P_1), rv(P_2)],$
   $[rv(P_0), rv(P_1), rv(P_2)]), (acceptP_0, acceptP_1, acceptP_0)) =$
   $([st(\mathbf{R}, P_1), st(\mathbf{R}, P_2)], [rv(P_1), rv(P_2)], [rv(P_1), rv(P_2)]);$
6. $t(([st(\mathbf{R}, P_1), st(\mathbf{R}, P_2)], [rv(P_1), rv(P_2)], [rv(P_1), rv(P_2)]),$
   $(sendP_3, \lambda, \lambda)) = ([st(\mathbf{R}, P_1), st(\mathbf{R}, P_2), st(\mathbf{R}, P_3)],$
   $[rv(P_1), rv(P_2), rv(P_3)], [rv(P_1), rv(P_2), rv(P_3)]);$
7. $t(([st(\mathbf{R}, P_1), st(\mathbf{R}, P_2), st(\mathbf{R}, P_3)], [rv(P_1), rv(P_2), rv(P_3)],$
   $[rv(P_1), rv(P_2), rv(P_3)]), (acceptP_1, acceptP_2, acceptP_1)) =$
   $([st(\mathbf{R}, P_2), st(\mathbf{R}, P_3)], [rv(P_2), rv(P_3)], [rv(P_2), rv(P_3)]]);$

$\dots$

- $t(([st(\mathbf{R}, P_n), st(\mathbf{R}, P_{n+1})], [rv(P_n), rv(P_{n+1})],$
  $[rv(P_n), rv(P_{n+1})]), (sendP_{n+2}, \lambda, \lambda)) =$
  $([st(\mathbf{R}, P_n), st(\mathbf{R}, P_{n+1}), st(\mathbf{R}, P_{n+2})], [rv(P_n), rv(P_{n+1}),$
  $rv(P_{n+2})], [rv(P_n), rv(P_{n+1}), rv(P_{n+2})])$, for $n > 0$;
- $t(([st(\mathbf{R}, P_n), st(\mathbf{R}, P_{n+1}), st(\mathbf{R}, P_{n+2})],$
  $[rv(P_n), rv(P_{n+1}), rv(P_{n+2})], [rv(P_n), rv(P_{n+1}), rv(P_{n+2})]),$
  $(acceptP_n, acceptP_{n+1}, dropP_n)) = ([st(\mathbf{R}, P_{n+1}), st(\mathbf{R}, P_{n+2})],$
  $[rv(P_{n+1}), rv(P_{n+2})], [rv(P_{n+1}), rv(P_{n+2})])$, for $n > 0$;

Intruder blocks packet P_1

1. $t(([st(\mathbf{R}, P_0)], [rv(P_0)], [rv(P_0)]), (sendP_1, \lambda, dropP_1)) =$
   $([st(\mathbf{R}, P_0), st(\mathbf{R}, P_1)], [rv(P_0)], [rv(P_0), rv(P_1)]);$
2. $t([st(\mathbf{R}, P_0), st(\mathbf{R}, P_1)], [rv(P_0)], [rv(P_0), rv(P_1)]),$
   $(sendP_2, \lambda, \lambda)) = ([st(\mathbf{R}, P_0), st(\mathbf{R}, P_1), st(\mathbf{R}, P_2)],$
   $[rv(P_0), rv(P_2)], [rv(P_0), rv(P_1), rv(P_2)]);$
3. $t([st(\mathbf{R}, P_0), st(\mathbf{R}, P_1), st(\mathbf{R}, P_2)], [rv(P_0), rv(P_2)],$
   $[rv(P_0), rv(P_1), rv(P_2)]), (\lambda, stop, \lambda)) = ([st(\mathbf{R}, P_0),$
   $st(\mathbf{R}, P_1), st(\mathbf{R}, P_2)], [stop], [rv(P_0), rv(P_1), rv(P_2)]);$
4. $t([st(\mathbf{R}, P_0), st(\mathbf{R}, P_1), st(\mathbf{R}, P_2)], [stop], [rv(P_0), rv(P_1),$
   $rv(P_2)]), (\lambda, stop, \lambda)) = ([st(\mathbf{R}, P_0), st(\mathbf{R}, P_1), st(\mathbf{R}, P_2)],$
   $[stop], [rv(P_0), rv(P_1), rv(P_2)]);$

Intruder blocks packet P_n, $n > 1$

1. $t(([st(\mathbf{R}, P_{n-2}), st(\mathbf{R}, P_{n-1})], [rv(P_{n-2}), rv(P_{n-1})],$
   $[rv(P_{n-2}), rv(P_{n-1})]), (sendP_n, \lambda, dropP_n)) =$
   $([st(\mathbf{R}, P_{n-2}), st(\mathbf{R}, P_{n-1}), st(\mathbf{R}, P_n)],$
   $[rv(P_{n-2}), rv(P_{n-1})], [rv(P_{n-2}), rv(P_{n-1}), rv(P_n)]);$
2. $t(([st(\mathbf{R}, P_{n-2}), st(\mathbf{R}, P_{n-1}), st(\mathbf{R}, P_n)],$
   $[rv(P_{n-2}), rv(P_{n-1})], [rv(P_{n-2}), rv(P_{n-1}), rv(P_n)]),$
   $(accept_{n-2}, \lambda, accept_{n-2})) = ([st(\mathbf{R}, P_{n-1}), st(\mathbf{R}, P_n)],$
   $[rv(P_{n-2}), rv(P_{n-1})], [rv(P_{n-1}), rv(P_n)]);$
3. $t(([st(\mathbf{R}, P_{n-1}), st(\mathbf{R}, P_n)], [rv(P_{n-2}), rv(P_{n-1})],$
   $[rv(P_{n-1}), rv(P_n)]), (sendP_{n+1}, \lambda, \lambda)) = ([st(\mathbf{R}, P_{n-1}),$
   $st(\mathbf{R}, P_n), st(\mathbf{R}, P_{n+1})], [rv(P_{n-2}), rv(P_{n-1}),$
   $rv(P_{n+1})], [rv(P_{n-1}), rv(P_n), rv(P_{n+1})]);$
4. $t(([st(\mathbf{R}, P_{n-1}), st(\mathbf{R}, P_n), st(\mathbf{R}, P_{n+1})], [rv(P_{n-2}),$
   $rv(P_{n-1}), rv(P_{n+1})], [rv(P_{n-1}), rv(P_n), rv(P_{n+1})]),$
   $(\lambda, stop, \lambda)) = ([st(\mathbf{R}, P_{n-1}), st(\mathbf{R}, P_n), st(\mathbf{R}, P_{n+1})],$
   $[stop], [rv(P_{n-1}), rv(P_n), rv(P_{n+1})]);$
5. $t([st(\mathbf{R}, P_{n-1}), st(\mathbf{R}, P_n), st(\mathbf{R}, P_{n+1})], [stop],$
   $[rv(P_{n-1}), rv(P_n), rv(P_{n+1})]), (\lambda, stop, \lambda)) =$
   $([st(\mathbf{R}, P_{n-1}), st(\mathbf{R}, P_n), st(\mathbf{R}, P_{n+1})], [stop],$
   $[rv(P_{n-1}), rv(P_n), rv(P_{n+1})]);$

Intruder fakes packet P_1

1. $t(([st(\mathbf{R}, P_0)], [rv(P_0)], [rv(P_0)]), (sendP_1, \lambda, fakeP_1)) =$
   $([st(\mathbf{R}, P_0), st(\mathbf{R}, P_1)], [rv(P_0), rv(P_1')],$
   $[rv(P_0), rv(P_1), st(\mathbf{R}, P_1')]);$

2.a $t([st(\mathbf{R}, P_0), st(\mathbf{R}, P_1)], [rv(P_0), rv(P_1')], [rv(P_0), rv(P_1),$
$st(\mathbf{R}, P_1')]), (sendP_2, \lambda, fakeP_2)) = ([st(\mathbf{R}, P_0),$
$st(\mathbf{R}, P_1), st(\mathbf{R}, P_2)], [rv(P_0), rv(P_1'), rv(P_2')],$
$[rv(P_0), rv(P_1), st(\mathbf{R}, P_1'), rv(P_2)], st(\mathbf{R}, P_2'));$

3.a $t([st(\mathbf{R}, P_0), st(\mathbf{R}, P_1), st(\mathbf{R}, P_2)], [rv(P_0), rv(P_1'), rv(P_2')],$
$[rv(P_0), rv(P_1), st(\mathbf{R}, P_1'), rv(P_2), st(\mathbf{R}, P_2')]), (\lambda, stop, \lambda)) =$
$([st(\mathbf{R}, P_0), st(\mathbf{R}, P_1), st(\mathbf{R}, P_2)], [stop],$
$[rv(P_0), rv(P_1), st(\mathbf{R}, P_1'), rv(P_2), st(\mathbf{R}, P_2')]);$

4.a $t([st(\mathbf{R}, P_0), st(\mathbf{R}, P_1), st(\mathbf{R}, P_2)], [stop],$
$[rv(P_0), rv(P_1), st(\mathbf{R}, P_1'), rv(P_2), st(\mathbf{R}, P_2')]),$
$(\lambda, stop, \lambda)) = ([st(\mathbf{R}, P_0), st(\mathbf{R}, P_1), st(\mathbf{R}, P_2)],$
$[stop], [rv(P_0), rv(P_1), st(\mathbf{R}, P_1'), rv(P_2), st(\mathbf{R}, P_1')]);$

2.b $t([st(\mathbf{R},P_0), st(\mathbf{R},P_1)], [rv(P_0), rv(P_1')], [rv(P_0), rv(P_1), st(\mathbf{R},P_1')]), (sendP_2, \lambda, \lambda)) = ([st(\mathbf{R},P_0), st(\mathbf{R},P_1), st(\mathbf{R},P_2)], [rv(P_0), rv(P_1'), rv(P_2)], [rv(P_0), rv(P_1), st(\mathbf{R},P_1'), rv(P_2)]);$

3.b $t([st(\mathbf{R},P_0), st(\mathbf{R},P_1), st(\mathbf{R},P_2)], [rv(P_0), rv(P_1'), st(\mathbf{R},P_1'), rv(P_2)], [rv(P_0), rv(P_1), rv(P_2)]), (\lambda, stop, \lambda)) = ([st(\mathbf{R},P_0), st(\mathbf{R},P_1), st(\mathbf{R},P_2)], [stop], [rv(P_0), rv(P_1), st(\mathbf{R},P_1'), rv(P_2)]);$

4.b $t([st(\mathbf{R},P_0), st(\mathbf{R},P_1), st(\mathbf{R},P_2)], [stop], [rv(P_0), rv(P_1), st(\mathbf{R},P_1'), rv(P_2)]), (\lambda, stop, \lambda)) = ([st(\mathbf{R},P_0), st(\mathbf{R},P_1), st(\mathbf{R},P_2)], [stop], [rv(P_0), rv(P_1), st(\mathbf{R},P_1'), rv(P_2)]);$

Intruder fakes packet P_n, $n > 1$

1. $t(([st(\mathbf{R},P_{n-2}), st(\mathbf{R},P_{n-1})], [rv(P_{n-2}), rv(P_{n-1})], [rv(P_{n-2}), rv(P_{n-1})]), (sendP_n, \lambda, fakeP_n)) = ([st(\mathbf{R},P_{n-2}), st(\mathbf{R},P_{n-1}), st(\mathbf{R},P_n)], [rv(P_{n-2}), rv(P_{n-1}), rv(P_n')], [rv(P_{n-2}), rv(P_{n-1}), rv(P_n)]);$
2. $t(([st(\mathbf{R},P_{n-2}), st(\mathbf{R},P_{n-1}), st(\mathbf{R},P_n)], [rv(P_{n-2}), rv(P_{n-1}), rv(P_n')], [rv(P_{n-2}), rv(P_{n-1}), rv(P_n)]), (\lambda, stop, \lambda)) = ([st(\mathbf{R},P_{n-2}), st(\mathbf{R},P_{n-1}), st(\mathbf{R},P_n)], [stop], [rv(P_{n-2}), rv(P_{n-1}), rv(P_n)]);$
3. $t([st(\mathbf{R},P_{n-2}), st(\mathbf{R},P_{n-1}), st(\mathbf{R},P_n)], [stop], [rv(P_{n-2}), rv(P_{n-1}), rv(P_n)]), (\lambda, stop, \lambda)) = ([st(\mathbf{R},P_{n-2}), st(\mathbf{R},P_{n-1}), st(\mathbf{R},P_n)], [stop], [rv(P_{n-2}), rv(P_{n-1}), rv(P_n)]);$

The above evolution function determines not only the set of reachable global states $S \subseteq L_\mathbf{S} \times L_\mathbf{R} \times L_\mathbf{I}$, but also gives the transition relation T; namely, for all the s, $s' \in S$, $(s, s') \in T$ iff there exists $act \in Act$ such that $t(s, act) = s'$.

We have now defined states, actions, protocols, and transitions for the model M for TESLA. To conclude, we define a valuation function $\mathcal{V} : S \to 2^{\mathcal{PV}}$ and the awareness functions $\mathbb{A}_X : L_X \to 2^{\mathcal{WF}(TDL)}$, for $X \in \{\mathbf{S}, \mathbf{R}, \mathbf{I}\}$. We first introduce the following set $\mathcal{PV}$ of propositional variables, which we find useful in analysis of the TESLA scenario:

$$\mathcal{PV} = \{has_\mathbf{R}(m), sent_\mathbf{S}(m), received_\mathbf{R}(m), dropped_\mathbf{I}(m), faked_\mathbf{I}(m) \mid m \in M_\mathbf{S} \cup M_\mathbf{I}\}$$

We define $\mathcal{V} : S \to 2^{\mathcal{PV}}$ by assuming that:

- for $i > 0$, $has_\mathbf{R}(t_i) \in \mathcal{V}(s)$ if there exist packets P_{i-1}, P_i and P_{i+1} such that $f(k_i) \sqsubseteq P_{i-1}$, $t_i \sqsubseteq P_i$, $k_i \sqsubseteq P_{i+1}$, $recv(P_{i-1}) \in l_\mathbf{R}(s)$, $recv(P_i) \in l_\mathbf{R}(s)$ and $recv(P_{i+1}) \in l_\mathbf{R}(s)$,
- $sent_\mathbf{S}(m) \in \mathcal{V}(s)$ if there exists packet P_i such that $m \sqsubseteq P_i$ and $sent(\mathbf{R}, P_i) \in l_\mathbf{S}(s)$, for any $m \in M_\mathbf{S}$,
- $received_\mathbf{R}(m) \in \mathcal{V}(s)$ if $recv(m) \in l_\mathbf{R}(s)$, for any $m \in M_\mathbf{S} \cup M_\mathbf{I}$,
- $dropped_\mathbf{I}(m) \in \mathcal{V}(s)$ if $recv(m) \notin l_\mathbf{R}(s)$ and $recv(m) \in l_\mathbf{I}(s)$, for any $m \in M_\mathbf{S}$,
- $faked_\mathbf{I}(m) \in \mathcal{V}(s)$ if there exist packets P_j such that $m \sqsubseteq P_j$ and $send(\mathbf{R}, P_j) \in l_\mathbf{I}(s)$, for any $m \in M_\mathbf{S} \cup M_\mathbf{I}$.

For $X \in \{\mathbf{S}, \mathbf{I}\}$, we take the following awareness function $\mathbb{A}_X : L_X \to 2^{\mathcal{WF}(TDL)}$: $\mathbb{A}_X(l) = \emptyset$, for any $l \in L_X$. In turn, $\mathbf{R}$'s awareness function $\mathbb{A}_\mathbf{R} : L_\mathbf{R} \to 2^{\mathcal{WF}(TDL)}$ is defined as follows. For any $l \in L_\mathbf{R}$, $\alpha \in \mathbb{A}_\mathbf{R}(l)$ if one of the following holds:

- $\alpha = received_\mathbf{R}(m)$ and $recv(m) \in l$ and $m \in M_\mathbf{S} \cup M_\mathbf{I}$,
- $\alpha = faked_\mathbf{I}(m)$ and $l = [stop]$ and $m \in M_\mathbf{S} \cup M_\mathbf{I}$,
- $\alpha = dropped_\mathbf{I}(m)$ and $l = [stop]$ and $m \in M_\mathbf{S}$,
- $\alpha = has_\mathbf{R}(m)$ and ($recv(m) \in l$ or $\exists m'$ such that $m \sqsubseteq m'$ and $recv(m') \in l$) and $m \in M_\mathbf{S} \cup M_\mathbf{I}$.

We have now completed defining the model M for TESLA.

Note that when we put a bound on the number of packets to be sent, M becomes finite.

Given the model M as defined above, we proceed to the semantical analysis of the TESLA protocol. First, we would like to establish whether or not TESLA satisfies the desired security property: *"the receiver does not accept as authentic any message unless it was actually sent by the sender"*. It can be expressed by the following TDL formula: for any $i > 0$,

$$has_\mathbf{R}(t_i) \Rightarrow (sent_\mathbf{S}(P_{i-1}) \wedge sent_\mathbf{S}(P_i) \wedge sent_\mathbf{S}(P_{i+1})) \quad (1)$$

It is easy (albeit tedious) to check manually that formula 1 is valid in M.

As one can notice, formula 1 is a pure propositional formula, so in order to verify the TESLA security property there is no need to use strong formalisms like epistemic or temporal logics. Indeed, this property can be checked via process algebras [5] or via Lynch-Vaandrager automata [3]. However, unlike the process algebras and Lynch-Vaandrager automata, the TDL language together with its semantic model is a powerful formalism that allows us to model and verify other properties of TESLA. In particular, using the model M and TDL we can show that TESLA satisfies a much stronger property: *"the receiver does not accept as authentic any message unless he knows that it was actually sent by the sender"*. Indeed it can be checked that, for any $i > 0$,

$$M \models \quad has_\mathbf{R}(t_i) \Rightarrow \mathcal{K}_\mathbf{R}(sent_\mathbf{S}(P_{i-1}) \wedge sent_\mathbf{S}(P_i) \wedge sent_\mathbf{S}(P_{i+1})) \quad (2)$$

Note that our model M of TESLA does not fulfil the contrappositive of the above, i.e. the following holds:

$$M \not\models \quad \mathcal{K}_\mathbf{R}(sent_\mathbf{S}(P_{i-1}) \wedge sent_\mathbf{S}(P_i) \wedge sent_\mathbf{S}(P_{i+1})) \Rightarrow has_\mathbf{R}(t_i) \quad (3)$$

The above seems reasonable, and it does not violate the security property given the presence of the intruder.

Further, we can check that TESLA actually meets the following property: *"it is always the case that the receiver does not accept as authentic any message unless he knows that it was actually sent by the sender"*.

$$M \models \quad \mathrm{A}\Box(has_\mathbf{R}(t_i) \Rightarrow \mathcal{K}_\mathbf{R}(sent_\mathbf{S}(P_{i-1}) \wedge sent_\mathbf{S}(P_i) \wedge sent_\mathbf{S}(P_{i+1}))) \quad (4)$$

The principals know about the presence of the intruder, so, the following should holds: for some $i > 0$:

$$M \models K_\mathbf{S}\mathrm{E}\Diamond(sent_\mathbf{S}(P_i) \wedge \neg received_\mathbf{R}(P_i)) \quad (5)$$

Indeed, a manual checking of M that we have done confirmed the above observation.

Also, by analysing TESLA one can expect that the protocol should fulfil the following property: *if a packet is faked, then the receiver should discover this.* It expresses the fact that the receiver is able to check the source of messages. This can be formalised as follows:

$$M \models faked_\mathbf{I}(P_i) \Rightarrow \mathcal{D}_\mathbf{R}(faked_\mathbf{I}(P_i)) \quad (6)$$

Again, a manual checking of M confirmed the above.

Investigating the TESLA model M, we could observe that if the receiver obtains some packets P_{i-1}, P_i, and P_{i+1} with a message $t_i \sqsubseteq P_i$, and he does not accept t_i as authentic, then he knows that at least one of the packets was not sent by the intended sender. Moreover, if a packet was indeed faked, the receiver is able to deduce this fact. So, property 7 should hold, and a manual check of M that we have done confirmed this. For any $i > 0$,

$$M \models (received_{\mathbf{R}}(P_{i-1}) \wedge received_{\mathbf{R}}(P_i) \wedge received_{\mathbf{R}}(P_{i+1}) \wedge \neg has_{\mathbf{R}}(t_i)) \Rightarrow \big(\mathcal{K}_{\mathbf{R}}(\neg sent_{\mathbf{S}}(P_{i-1}) \vee \neg sent_{\mathbf{S}}(P_i) \vee \neg sent_{\mathbf{S}}(P_{i+1})) \wedge (\mathcal{D}_{\mathbf{R}}(fake_{\mathbf{I}}(P_{i-1})) \vee \mathcal{D}_{\mathbf{R}}(fake_{\mathbf{I}}(P_i)) \vee \mathcal{D}_{\mathbf{R}}(fake_{\mathbf{I}}(P_{i+1})))\big) \quad (7)$$

Messages are sent to the receiver according to the schedule. This schedule is known to the intruder as well. So, when the intruder tries to fake the current transmission, he has to follow the sending schedule, otherwise, the receiver will stop the reception immediately. This means that the intruder has to send a packet at each interval, which was agreed by the sender and the receiver at the beginning of the transmission under consideration. This observation can be formalised as follows. For any $i > 0$,

$$M \models \mathrm{A}\Box(dropped_{\mathbf{I}}(P_i) \Rightarrow \mathcal{D}_{\mathbf{R}}(dropped_{\mathbf{I}}(P_i)) \quad (8)$$

It can be checked that property 8 also holds.

6. CONCLUSION

In the paper we have proposed a new security-specialised logic TDL defined on a computationally-grounded semantics, and with a sound and complete axiomatisation. TDL employs different concepts of knowledge (based on the notion of awareness), as well as time in order to represent the knowledge of principals that operate in an environment that is under control of an intruder. Further, we have applied the logic to reason about TESLA, which we have formalised in the interpreted system framework. A temporal epistemic analysis has allowed us not only to reconfirm the TESLA authentication property but also to check more sophisticated specifications.

Obviously we do not suggest one should check security protocols manually as done here. Work is in progress to extend a BDD-based model checker [17] to encode the epistemic notions described here. Since the semantics of TDL is defined on interpreted systems we are hopeful that the exercise conduced here manually can be automated. Since TDL has a computationally-grounded semantics, a model checking technique for it can be defined, which is the scope of a further investigation.

7. REFERENCES

[1] M. Abadi and M. R. Tuttle. A semantics for a logic of authentication. In *Proc. of PODC '91*, pp. 201–216. ACM Press, 1991.

[2] R. Accorsi, D. Basin, and L. Viganò. Towards an awareness-based semantics for security protocol analysis. In *Post Proc. of LACPV'01*, volume 55(1) of ENTCS, pp 5–24. Elsevier, 2003.

[3] M. Archer. Proving correctness of the basic TESLA multicast stream authentication protocol with TAME. In *Proc. of WITS'02*, 2002.

[4] C. Boyd and W. Mao. On a limitation of BAN logic. In *Proc. of EUROCRYPT '93*, pp. 240–247. Springer-Verlag, 1994.

[5] P. J. Broadfoot and G. Lowe. Analysing a stream authentication protocol using model checking. In *Proc. of ESORICS'02*, pp. 146–161. Springer-Verlag, 2002.

[6] M. Burrows, M. Abadi, and R. Needham. A logic of authentication. In *Proc. of the Royal Society of London A*, volume 426, pp. 233–271 , 1989.

[7] E. Clarke and E. Emerson. Design and synthesis of synchronization skeletons for branching-time temporal logic. In *Proc. of WLP*, volume 131 of *LNCS*, pp. 52–71. Springer-Verlag, 1981.

[8] C. Dixon, M. C. Fernandez Gago, M. Fisher, and W. van der Hoek. Using temporal logics of knowledge in the formal verification of security protocols. In *Proc. of TIME'04*, pp. 148–151. IEEE, 2004.

[9] D. Dolev and A. Yao. On the security of public key protocols. *IEEE Trans. Inf. Theory*, 29(2):198–208, 1983.

[10] R. Fagin and J. Y. Halpern. Belief, awareness, and limited reasoning. *Artificial Intelligence*, 34(1):39–76, 1988.

[11] R. Fagin, J. Y. Halpern, Y. Moses, and M. Y. Vardi. *Reasoning about Knowledge*. MIT Press, 1995.

[12] Li Gong, R. Needham, and R. Yahalom. Reasoning About Belief in Cryptographic Protocols. In *Proceedings of IEEE Symposium on Research in Security and Privacy*, pp. 234–248. IEEE Press, 1990.

[13] J. Y. Halpern, Y. Moses, and M. Y. Vardi. Algorithmic knowledge. In *Proc. of TARK 1994*, pages 255–66. Morgan Kaufmann Publishers, 1994.

[14] J.Y. Halpern and R. Pucella. Modeling Adversaries in a Logic for Security Protocol Analysis. In *Proc. of FASec'02*, volume 2629 of *LNCS*, pp. 115–132. Springer-Verlag, 2002.

[15] R. Kailar and V. D. Gligor. On belief evolution in authentication protocols. In *Proc. of Computer Security Foundations Workshop IV*, pp. 103–116. IEEE Press, 1991.

[16] A. Lomuscio and B. Woźna. A combination of explicit and deductive knowledge with branching time: completeness and decidability results. *In Post-proc. of DALT'05*, volume 3904 of LNAI. 2006. To appear.

[17] A. Lomuscio and F. Raimondi. *MCMAS: A model checker for multi-agent systems.* In *Proc. of TACAS'06*. Springer Verlag, March, 2006. To appear.

[18] D. M. Nessett. A critique of the Burrows Abadi Needham logic. *Operating Systems Review*, 24(3):35–38, 1990.

[19] A. Perrig, R. Canetti, J.D. Tygar, and Dawn X. Song. Efficient authentication and signing of multicast streams over lossy channels. In *IEEE Symposium on Security and Privacy*, pp. 56–73, 2000.

[20] R. Pucella. Deductive Algorithmic Knowledge. In *Proc. of SAIM'04*, AI&M 22-2004, 2004.

[21] R. Ramanujam and S. P. Suresh. Deciding knowledge properties of security protocols. In *Proc. of TARK'05*, pp. 218–235, 2005.

[22] M. Wooldridge. *Reasoning about Rational Agents.* MIT Press, 2000.