Towards a Metalogic for Security Protocol Analysis*

(Extended Abstract)

Carlos Caleiro[1] Luca Viganò[2] David Basin[2]

[1] CLC, Department of Mathematics, IST, Lisbon, Portugal
cs.math.ist.utl.pt/ccal.html

[2] Department of Computer Science, ETH Zurich, Switzerland
www.infsec.ethz.ch/~vigano www.infsec.ethz.ch/~basin

1 Introduction

Many security protocols have been proposed to help build secure distributed systems. Given how difficult it is for humans to predict all possible ways for distributed computation to proceed, it is not so surprising that attacks have been found on many protocols that were originally believed to be secure. Due to the subtlety of the problem, the use of formal methods for analyzing security protocols has been gaining popularity. These include approaches based on process algebras, e.g. [5, 14, 20, 25], which support elegant models but lack a suitable logical language to express protocol properties; model-checking and related techniques, such as [2, 3, 11, 22, 26], which are suitable for automation but rely on simplifying assumptions (to yield finite models) and hence are difficult to use reliably on different applications; special-purpose epistemic logics like BAN, e.g. [10], which provide for high-level knowledge-based formalizations of protocols and their properties, but whose semantics is complex, restricted, or simply lacking; and inductive theorem proving, like [24], which are general, but require time consuming interactive theorem proving by experienced researchers.

In this paper, we report on our work-in-progress on the formalization of a suitable version of temporal logic for communicating agents which provides both an *object level tool*, where we can specify and reason about specific protocols, and a *metalevel tool* for the compared analysis of security protocol models and properties. Our starting point is the work of [15, 17], which focus on the expressibility of properties from the local point of view of each agent, and that we extend in order to express also global properties. Besides its very clean interpretation structures, which provide a nice and intuitive model of distributed systems, our reasons for using this logic are primarily threefold. First, its temporal dimension can be effectively used to formalize and reason about *interleaved* protocol executions; this is in contrast to other approaches based on epistemic or doxastic logics, which are not well-suited for reasoning about such interleavings but consider only single protocol executions. Second, its distributed dimension, with explicit agent identification, supports formalizing the different security properties that the protocols have been designed to achieve, such as secrecy of information and different notions of authentication between agents. Finally, it is well-suited for specifying communicating agents in distributed systems.

Using the logic we are able to specify a protocol-independent distributed communication model, on top of which protocols can be formally defined and analyzed. For instance, we have used the logic to analyze a number of protocols and properties the protocols are supposed to establish. In particular, we have analyzed the well-known Needham-Schroeder Public-Key Protocol (NSPK) [18]. We have thereby been able both to find the usual man-in-the-middle attack to the

*This work was partially supported by FCT and EU FEDER via the Project FibLog POCTI/MAT/37239/2001 of CLC, and by the FET Open Project IST-2001-39252 and the BBW Project 02.0431, "AVISPA: Automated Validation of Internet Security Protocols and Applications".

NSPK and to show that authentication properties hold for the corrected version NSL (given by Lowe to prevent the man-in-the-middle attack).

The principal aim of our work, however, is not merely the concrete analysis of specific protocols. Rather, our long-term objective is to use our logic as a *metalevel tool* for the compared analysis of security protocol models and properties. Our logic provides a basis to rigorously investigate general metalevel properties of different protocol models, by establishing modeling and analysis simplification techniques that may contribute to the sound design of effective protocol validation tools. In this regard, we believe that our logic can contribute to clarifying the concepts involved through a natural representation of the underlying computational models. We anticipate several applications. The most direct consists of a rigorous account of widely used simplification techniques, namely by reasoning about (formally proving) the correctness of widely used simplification principles, like bounds on the number of principals involved, the adequacy of the intruder trace as an abstraction of the hostile communication channel, step-compression and other reduction, abstraction and approximation techniques (see, for example, [1, 3, 4, 6, 9, 12, 16, 23]). A number of promising preliminary results are described in [7, 8].

2 Distributed temporal logic

DTL [15] is a logic for reasoning about temporal properties of distributed systems from the local point of view of its agents, which are assumed to execute sequentially and to interact by means of synchronous event sharing. Distribution is implicit, making it easier to state the properties of an entire system through the local properties of its component agents and their interaction. Herein, we introduce a version of DTL tailored to allow also for the smooth spelling and proof of global properties.

The logic is defined in the context of a given *signature* of a distributed system, $\langle Id, \{Act_i\}_{i\in Id}, x\{Prop_i\}_{i\in Id}\rangle$ where Id is a finite set (of agent identifiers) and, for each $i \in Id$, Act_i is a set (of local action symbols) and $Prop_i$ is a set (of local state propositions). The *global language* $\mathcal{L}$ is defined by the grammar

$$\mathcal{L} ::= @_i[\mathcal{L}_i] \mid \bot \mid \mathcal{L} \Rightarrow \mathcal{L}\,,$$

where the *local languages* $\mathcal{L}_i$ for each $i \in Id$ are defined by

$$\mathcal{L}_i ::= Act_i \mid Prop_i \mid \bot \mid \mathcal{L}_i \Rightarrow \mathcal{L}_i \mid \mathcal{L}_i \,\mathsf{U}\, \mathcal{L}_i \mid \mathcal{L}_i \,\mathsf{S}\, \mathcal{L}_i \mid @_j[\mathcal{L}_j]\,,$$

with $j \in Id$. Locally for an agent, U and S are respectively the *until* and *since* temporal operators. Furthermore, actions correspond to true statements of an agent when they have just occurred, whereas state propositions characterize the current local states of the agents. Note that $@_j[\varphi]$ means different things depending on the context. If it is a global formula, it means that φ holds at the current local state of agent j. If it is a local formula appearing inside an $@_i$-formula, it means that agent i has just communicated with agent j for whom φ held.

We can then define a number of other operators as abbreviations as usual, e.g. $\neg$, $\top$, $\vee$, $\wedge$, $\Leftrightarrow$, as well as:

$\mathsf{X}\varphi$	$\equiv \bot\,\mathsf{U}\,\varphi$	next	$\dagger$	$\equiv \neg\mathsf{X}\top$	in the end
$\mathsf{Y}\varphi$	$\equiv \bot\,\mathsf{S}\,\varphi$	previous	$*$	$\equiv \neg\mathsf{Y}\top$	in the beginning
$\mathsf{F}\varphi$	$\equiv \top\,\mathsf{U}\,\varphi$	sometime in the future	$\mathsf{F}_\circ\varphi$	$\equiv \varphi \vee \mathsf{F}\varphi$	now or sometime in the future
$\mathsf{P}\varphi$	$\equiv \top\,\mathsf{S}\,\varphi$	sometime in the past	$\mathsf{P}_\circ\varphi$	$\equiv \varphi \vee \mathsf{P}\varphi$	now or sometime in the past
$\mathsf{G}\varphi$	$\equiv \neg\mathsf{F}\neg\varphi$	always in the future	$\mathsf{G}_\circ\varphi$	$\equiv \varphi \wedge \mathsf{G}\varphi$	now and always in the future
$\mathsf{H}\varphi$	$\equiv \neg\mathsf{P}\neg\varphi$	always in the past	$\mathsf{H}_\circ\varphi$	$\equiv \varphi \wedge \mathsf{H}\varphi$	now and always in the past

The interpretation structures of $\mathcal{L}$ are built upon adequate forms of Winskel's *event structures* [27]. A *local life-cycle* (of agent $i \in Id$) is a pair $\lambda_i = (Ev_i, \rightarrow_i)$ where Ev_i is a set (of local events) and $\rightarrow_i \subseteq Ev_i \times Ev_i$ is a (local successor) relation such that $\rightarrow_i^*$ defines a well-founded total order of causality on Ev_i. Of course, Ev_i can be finite or infinite, but it is always denumerable.

A *local configuration* of λ_i is any finite set $\xi_i \subseteq Ev_i$ such that if $e' \in \xi_i$ and $e \rightarrow_i e'$ then $e \in \xi_i$. We denote the set of all local configurations of λ_i by Ξ_i. Clearly, every local configuration $\xi_i \neq \emptyset$ has a maximum event that we denote by $last(\xi_i)$. Moreover, for every local configuration $\xi_i \neq Ev_i$ there exists a unique next event $next(\xi_i)$, corresponding to the minimum event in $Ev_i \setminus \xi_i$, such that $\xi_i \cup \{next(\xi_i)\}$ is a local configuration.

A *distributed life-cycle* is a family $\lambda = \{\lambda_i\}_{i \in Id}$, where $\lambda_i = (Ev_i, \rightarrow_i)$ is the local life-cycle of each agent i, such that $\rightarrow^* = (\bigcup_{i \in Id} \rightarrow_i)^*$ defines a partial order of causality on the set $Ev = \bigcup_{i \in Id} Ev_i$ of all events. Note that each event may be shared by several agents at communication points. Therefore, the condition that $\rightarrow^*$ defines a global ordering of the set of events amounts to requiring that communication does not introduce cycles among the different local causality orderings. A *global configuration* of λ is any set $\xi \subseteq Ev$ such that if $e' \in \xi$ and $e \rightarrow e'$ then $e \in \xi$. We denote the set of all global configurations of λ by Ξ. Clearly, every global configuration ξ contains a local configuration $\xi|_i = \xi \cap Ev_i$ for each $i \in Id$. Moreover, given $E \subseteq Ev$ finite, $(E \downarrow) = \{e' \in Ev \mid e' \rightarrow^* e \text{ for some } e \in E\}$ is a global configuration. Given a global configuration ξ and $E \not\subseteq \xi$, $(\xi + E)$ stands for $\xi \cup (E \downarrow)$. If $E = \{e\}$ we just write $(e \downarrow)$ and $(\xi + e)$. The following lemma shows that the configurations of any distributed life-cycle λ can be built by consecutively adding events.

Lemma 2.1 *If $\xi, \xi' \in \Xi$ and $\xi \subsetneq \xi'$ then there exists $e \in \xi' \setminus \xi$ such that $\xi \cup \{e\} \in \Xi$.*

An *interpretation structure* for $\mathcal{L}$ is a suitably labeled distributed life-cycle, that is, a triple $\mu = \langle \lambda, \sigma, \alpha \rangle$ where λ is a distributed life-cycle, $\sigma = \{\sigma_i \mid \Xi_i \rightarrow 2^{Prop_i}\}_{i \in Id}$ is an agent-indexed family of maps that associate a local state to each local configuration, and $\alpha = \{\alpha_i \mid Ev_i \rightarrow 2^{Act_i}\}_{i \in Id}$, with $\alpha_i(e) \neq \emptyset$ and finite for every $e \in Ev_i$, is an agent-indexed family of maps that associate a non-empty finite set of local actions to each local event. We can now define the satisfaction relation at a given global configuration ξ of μ

$$\mu, \xi \Vdash @_i[\varphi] \text{ if } \mu, \xi|_i \Vdash_i \varphi; \quad \mu, \xi \nVdash \bot; \quad \text{and} \quad \mu, \xi \Vdash \gamma \Rightarrow \delta \text{ if } \mu, \xi \nVdash \gamma \text{ or } \mu, \xi \Vdash \delta,$$

where local satisfaction is defined by

- $\mu, \xi_i \Vdash_i act$ if $\xi_i \neq \emptyset$ and $\alpha_i(last(\xi_i)) = act$;
- $\mu, \xi_i \Vdash_i p$ if $p \in \sigma_i(\xi_i)$;
- $\mu, \xi_i \nVdash_i \bot$;
- $\mu, \xi_i \Vdash_i \varphi \Rightarrow \psi$ if $\mu, \xi_i \nVdash_i \varphi$ or $\mu, \xi_i \Vdash_i \psi$;
- $\mu, \xi_i \Vdash_i \varphi \,\mathsf{U}\, \psi$ if there exists $\xi_i'' \in \Xi_i$ with $\xi_i \subsetneq \xi_i''$ such that $\mu, \xi_i'' \Vdash_i \psi$, and $\mu, \xi_i' \Vdash_i \varphi$ for every $\xi_i' \in \Xi_i$ with $\xi_i \subsetneq \xi_i' \subsetneq \xi_i''$;
- $\mu, \xi_i \Vdash_i \varphi \,\mathsf{S}\, \psi$ if there exists $\xi_i'' \in \Xi_i$ with $\xi_i'' \subsetneq \xi_i$ such that $\mu, \xi_i'' \Vdash_i \psi$, and $\mu, \xi_i' \Vdash_i \varphi$ for every $\xi_i' \in \Xi_i$ with $\xi_i'' \subsetneq \xi_i' \subsetneq \xi_i$; and
- $\mu, \xi_i \Vdash_i @_j[\varphi]$ if $\xi_i \neq \emptyset$, $last(\xi_i) \in Ev_j$ and $\mu, (last(\xi_i) \downarrow)|_j \Vdash_j \varphi$.

As usual, we say that μ is a model of $\Gamma \subseteq \mathcal{L}$ if $\mu, \xi \Vdash \gamma$ for every global configuration ξ of μ and every $\gamma \in \Gamma$.

We now present some useful lemmas about the logic.

Lemma 2.2 (Local properties) *Let $\varphi \in \mathcal{L}_i$ be a local formula and μ an interpretation structure. If $\xi, \xi' \in \Xi$ are such that $\xi|_i = \xi'|_i$ then $\mu, \xi \Vdash @_i[\varphi]$ if and only if $\mu, \xi' \Vdash @_i[\varphi]$.*

In the sequel, if $\varphi \in \mathcal{L}_i$ does not include @ subformulas then it is called a *private formula*. Furthermore, if φ is also free of the temporal operators U and S then it is called a *state formula*. The following is a useful lemma concerning private formulas.

Lemma 2.3 (Private properties) *Let $\varphi \in \mathcal{L}_i$ be a private formula and μ, μ' interpretation structures with $\mu_i = \mu'_i$. If $\xi \in \Xi$ and $\xi' \in \Xi'$ are such that $\xi|_i = \xi'|_i$ then $\mu, \xi_i \Vdash @_i[\varphi]$ if and only if $\mu', \xi'_i \Vdash @_i[\varphi]$.*

The logic allows us to state the following invariance rule for global properties.

Proposition 2.4 (Global invariance rule) *Let $\gamma \in \mathcal{L}$ be a global formula, μ an interpretation structure and $\xi \in \Xi$ a global configuration. If both $\mu, \xi \Vdash \gamma$, and $\mu, \xi' \Vdash \gamma$ implies $\mu, \xi' \cup \{e\} \Vdash \gamma$ for every $\xi' \in \Xi$ and $e \in Ev \setminus \xi'$ such that $\xi \subseteq \xi'$ and $\xi' \cup \{e\} \in \Xi$, then $\mu, \xi' \Vdash \gamma$ for every $\xi' \in \Xi$ such that $\xi \subseteq \xi'$.*

For local state properties, the invariance rule can be stated in the following more familiar way.

Proposition 2.5 (Local invariance rule) *Let $\varphi \in \mathcal{L}_i$ be a local state formula, μ an interpretation structure and $\xi_i \in \Xi_i$ a local configuration. If both $\mu, \xi_i \Vdash_i \varphi$, and $\mu, \xi'_i \Vdash_i \varphi$ implies $\mu, \xi'_i \cup \{next(\xi'_i)\} \Vdash \varphi$ for every $\xi'_i \in \Xi_i$ such that $\xi_i \subseteq \xi'_i \subsetneq Ev_i$, then $\mu, \xi'_i \Vdash_i \varphi$ for every $\xi'_i \in \Xi_i$ such that $\xi_i \subseteq \xi'_i$, or equivalently, $\mu, \xi_i \Vdash_i \mathsf{G}_\circ \varphi$.*

Hence, μ is a model of $@_i[\varphi]$ if and only if μ is a model of both $@_i[*\Rightarrow\varphi]$ and $@_i[(\varphi \wedge \mathsf{X}\top) \Rightarrow \mathsf{X}\varphi]$, or equivalently, $@_i[(\varphi \wedge \mathsf{X}\, act) \Rightarrow \mathsf{X}\varphi]$ for every $act \in Act_i$.

3 The network model

We provide the specification of a generic open network where agents interact by exchanging messages through an insecure public channel. A *network signature* is a pair $\langle Pr, Nam\rangle$, where Pr is a finite set of principal identifiers $A, B, C, \ldots$, and Nam is a family $\{Nam_A\}_{A \in Pr}$ of pairwise disjoint finite sets of *names*, corresponding to the possible aliases used by each principal (the importance of aliases will become clearer below. We write A' to denote a name used by principal A. By abuse of notation, we also use $Nam = \bigcup_{A \in Pr} Nam_A$. Furthermore, we assume fixed two sets Non and Key of "numbers" that can be used as *nonces* and *keys*, respectively, and whose members we denote by N and K, possibly with annotations. In general, we assume that several kinds of keys can coexist and that each key K has its own inverse key K^{-1}. *Messages*, which we denote by M possibly with annotations, are built inductively from *atomic messages* (names and "numbers"), by concatenation $(\_;\_)$, which we assume to be associative, and encryption under a key K ($\{\_\}_K$). The set Msg of messages is thus defined by

$$Msg ::= Nam \mid Non \mid Key \mid Msg; Msg \mid \{Msg\}_{Key}\,.$$

Given a network signature $\langle Pr, Nam\rangle$, we obtain a distributed signature by taking $Id = Pr \uplus \{Ch\}$, where Ch is the communication channel (used to model asynchronous communication), and letting the local alphabet of each agent (the principals and the channel) be defined as follows. The signature of a principal A requires actions Act_A and state propositions $Prop_A$, where Act_A includes

- $send(M, B')$ — sending of the message M to B';
- $rec(M)$ — reception of the message M;
- $spy(M)$ — eavesdropping of the message M; and
- $nonce(N)$ — generation of the fresh nonce N,

and $Prop_A$ includes

- $knows(M)$ — knowledge of the message M.

For the channel Ch we do not require any state propositions, i.e. $Prop_{Ch} = \emptyset$, whereas the actions Act_{Ch} include

- $in(M, A')$ — arrival at the channel of the message M addressed to A';

- $out(M, A')$ — delivery of M from the channel to principal A; and
- *leak* — leaking of messages.

In the network model, principals can send and receive messages, at will, always through the channel. If the principal A sends a message to B', then the message synchronously arrives at the channel, where it is stored for future delivery to B. If delivery ever happens, it must be synchronized with the corresponding receive action of B. However, principal A can only send M to B' if A knows both B' and M. As usual, the knowledge of principals is not static. In addition to their initial knowledge, principals gain knowledge from the messages they receive and the nonces they generate. Principals may also spy on messages being leaked by the channel and learn their content. We do not allow principals to explicitly divert messages, but we also do not guarantee that messages delivered to the channel are ever received.

To ensure that principals do not learn messages in an ad hoc fashion, we specify that the *knows* propositions only hold where strictly necessary. We follow the idea underlying Paulson's inductive model [24], in accordance with the usual assumption of *perfect cryptography*. We restrict attention to those interpretation structures μ such that, for every principal A, the following condition holds for all messages M and global configurations $\xi \in \Xi$ such that $\xi|_A \neq \emptyset$:

(K) $\mu, \xi \Vdash_A knows(M)$ iff

$$M \in synth(analyz(\{M' \mid \mu, \xi \Vdash_A (\mathsf{Y}\, knows(M')) \vee rec(M') \vee spy(M') \vee nonce(M')\}))\,,$$

where *analyz* and *synth* are the functions representing how principals analyze or synthesize messages from a given set of messages (see, e.g., [24]).

To guarantee the freshness and uniqueness of the nonces generated by each principal, we further require the axioms

(N1) $@_A[nonce(N) \Rightarrow \mathsf{Y} \neg\, knows(M_N)]$,

(N2) $@_A[nonce(N)] \Rightarrow \bigwedge_{B \in Pr \setminus \{A\}} @_B[\neg\, knows(M_N)]$,

where M_N ranges over all the messages containing the nonce N. Together with **(K)**, **(N1)** and **(N2)** guarantee that every nonce is generated at most once, if at all, in each model, and always freshly (taking also into account the initial knowledge of all agents). The specification of the network model also comprises a number of axioms that characterize the behavior of the channel and of each principal $A \in Pr$:

(C1) $@_{Ch}[in(M, A') \Rightarrow \bigvee_{B \in Pr} @_B[send(M, A')]]$;

(C2) $@_{Ch}[out(M, A') \Rightarrow \mathsf{P}\ in(M, A')]]$; and

(C3) $@_{Ch}[out(M, A') \Rightarrow @_A[rec(M)]]$,

(P1) $@_A[send(M, B') \Rightarrow \mathsf{Y}(knows(M) \wedge knows(B'))]$;

(P2) $@_A[send(M, B') \Rightarrow @_{Ch}[in(M, B')]]$;

(P3) $@_A[rec(M) \Rightarrow @_{Ch}[\bigvee_{A' \in Nam_A} out(M, A')]]$;

(P4) $@_A[spy(M) \Rightarrow @_{Ch}[leak \wedge \mathsf{P} \bigvee_{B' \in Nam} in(M, B')]]$;

(P5) $@_A[\bigwedge_{B \in Pr \setminus \{A\}} \neg @_B[\top]]$; and

(P6) $@_A[nonce(N) \Rightarrow \neg\, @_{Ch}[\top]]$.

The channel axioms **(C1–C3)** are straightforward. They state that a message addressed to A' only arrives at the channel if it is sent to A' by some principal B; that the channel only delivers a message to A' if such a message for A' has previously arrived; and that if the channel delivers a message to A' then A receives it. The principal axioms are also simple. **(P1)** is a precondition for sending a message, stating that the sender must know both the message and the recipient's name beforehand. The next three formulas are interaction axioms. **(P2)** and **(P3)** state that the sending and receiving of messages, respectively, must be shared with the corresponding arrival and delivery actions of the channel. **(P4)** guarantees that a spied message must have arrived at the channel, addressed to some recipient. The two final axioms limit the possible amount of interaction: **(P5)** guarantees that principals never communicate directly (only through the channel), and **(P6)** states that nonce generating actions are not communication actions.

4 Protocol modeling and analysis

Protocols are usually informally described by short sequences of messages that are exchanged by principals in order to achieve particular security goals in open, hostile environments. We illustrate protocol modeling on top of our network by using a standard example: the (flawed) simplified Needham-Schroeder Public-Key Protocol NSPK [18], which we present as the following sequence of message exchange steps.

$$\begin{array}{lcll} a \to b & : & (n_1). & \{n_1; a\}_{K_b} \\ b \to a & : & (n_2). & \{n_1; n_2\}_{K_a} \\ a \to b & : & & \{n_2\}_{K_b} \end{array}$$

In this notation a and b are variables of sort name that denote each of the roles played in one execution of the protocol, and n_1 and n_2 are variables of sort nonce. The arrows represent communication, from sender to receiver. The parenthesized nonces prefixing the first and second messages exchanges signify that these nonces must be freshly generated before the subsequent message is sent. Moreover, it is assumed that an underlying "infrastructure" of *public* and *private* keys exists: K_a represents the public key of a, whose inverse key should be private, i.e. known by no one but the principal using that name. Although other possibilities could be easily added to the model, we refrain from doing so here, for simplicity, and assume that these are the only existing keys.

Formalizing a protocol like the above involves defining the sequences of actions (*send*, *rec*, and *nonce*) taken by honest agents executing the protocol. Namely, for each role, we formalize the actions taken and the order in which they must be taken. In the case of NSPK there are two roles: an initiator role *Init*, represented by a, and a responder role *Resp*, represented by b. Given distinct names A' and B', of principals A and B respectively, and nonces N_1 and N_2, the role instantiations should correspond to the execution, by principal A, of the sequence of actions $\mathrm{run}_A^{Init}(A', B', N_1, N_2)$:

$$\langle nonce(N_1).send(\{N_1; A'\}_{K_{B'}}, B').rec(\{N_1; N_2\}_{K_{A'}}).send(\{N_2\}_{K_{B'}}, B')\rangle\,,$$

and to the execution, by principal B, of the sequence $\mathrm{run}_B^{Resp}(A', B', N_1, N_2)$:

$$\langle rec(\{N_1; A'\}_{K_{B'}}).nonce(N_2).send(\{N_1; N_2\}_{K_{A'}}, A').rec(\{N_2\}_{K_{B'}})\rangle\,.$$

If $\mathrm{run}_A^i(\overline{M}) = \langle act_1 \ldots act_n\rangle$ then we can consider the local formula $\mathrm{role}_A^i(\overline{M})$:

$$act_n \wedge \mathsf{P}(act_{n-1} \wedge \mathsf{P}(\ldots \wedge \mathsf{P}\, act_1)\ldots)\,.$$

For example, it should be clear that $\mu, \xi \Vdash @_A[\mathrm{role}_A^{Init}(A', B', N_1, N_2)]$ if and only if A has just completed at ξ the required sequence of actions.

4.1 Honesty

We take an external view of the system, supported by the *one-intruder* reduction reported in [7, 8], and consider a *protocol signature* to be a triple $\langle Hn, \{Z\}, Nam\rangle$ where Hn is the set of *honest* of principals and $Z \notin Hn$ is the *intruder*, and $\langle Hn \cup \{Z\}, Nam\rangle$ is a network signature such that every honest principal has exactly one name and never plays two distinct roles in the same protocol run. Without loss of generality, we assume that $Nam_A = \{A\}$ for every $A \in Hn$. We assume also that the private key of each honest principal is initially only known by that principal. This can be achieved by the axioms **(Key1)** and **(Key2)** below, where $A \in Hn$:

(Key1) $@_A[* \Rightarrow knows(K_A^{-1})]$; and
(Key2) $@_B[* \Rightarrow \neg\, knows(M)]$, for every $B \in Pr \setminus \{A\}$ and M containing K_A^{-1}.

Models of a protocol will be those network models where, furthermore, all honest principals follow the rules of the protocol. That is, for every $A \in Hn$, if the local life-cycle of A is $e_1 \to_A e_2 \to_A$

$e_3 \rightarrow_A \ldots$, then the corresponding (possibly infinite) sequence of actions $\langle \alpha_A(e_1).\alpha_A(e_2).\alpha_A(e_3)\ldots\rangle$ must be an interleaving of prefixes of possible protocol runs, but using distinct fresh nonces in each of them. In the case of NSPK, this means that the life-cycle of an honest agent must be built by interleaving prefixes of sequences of the form $\mathrm{run}_A^{Init}(A, B', N_1, N_2)$ or $\mathrm{run}_A^{Resp}(B', A, N_1, N_2)$, such that no two initiator runs can have the same N_1, no two responder runs can have the same N_2, and the N_1 of any initiator run must be different from the N_2 of any responder run.

4.2 Security goals

The aim of protocol analysis is to prove (or disprove) the correctness of a protocol with respect to the security goals that it is supposed to achieve. For instance, *secrecy* of the critical data exchanged during an execution of the protocol among its participants is certainly a goal to be achieved. In addition, an honest principal running the protocol should be able to *authenticate* the identities of its protocol partners through the examination of the messages he receives. Below, we show how to formulate the required secrecy and authentication goals of protocols in the general case, illustrating them by means of the NSPK protocol.

As usual, we call an *attack* to the protocol, and specifically to a given security goal, any protocol model μ and configuration ξ for which the formula expressing the goal does not hold. Let us start with secrecy.

Secrecy We can formalize that the messages in a finite set S will remain a shared secret between the participants after a complete execution of the protocol with participants $A_1, \ldots, A_j$ by the formula secr_S:

$$\bigwedge_{i=1}^{j} @_{A_i}[\mathsf{P}_\circ\ \mathrm{role}^i_{A_i}(\overline{M})] \Rightarrow \bigwedge_{B \in Pr \setminus \{A_1,\ldots,A_j\}} \bigwedge_{M \in S} @_B[\neg\, knows(M)].$$

Of course, this property can only be expected to hold in particular situations. Assume that all the participants in a complete run of the protocol are honest. One should then expect that the "critical" nonces generated during that run will remain a secret shared only by the participating principals. Indeed, being honest, they will not reuse those nonces in further protocol runs. Using the logic, we can check the property $\mathrm{secr}_F(\overline{M})$ for the relevant set F of fresh nonces. In the case of NSPK, this would amount to requiring $\mathrm{secr}_{\{N_1,N_2\}}(A, B, N_1, N_2)$, with A and B both honest.

Authentication There are many possible shades of authentication (see, e.g., [19]). However, most authors agree that authentication should be expressed as some kind of correspondence property between the messages an agent receives in a protocol run and the messages that other participants of the same run are supposed to send. The typical authentication goal states that if an honest principal A completes his part of a run of a protocol in role i, with certain partners and data, then it must be the case that these partners have also been actively involved by sending to A the messages that he received. The property that A authenticates a partner B in role j at step q of the protocol can be defined in our logic by the formula $\mathrm{auth}^{i,j,q}_{A,B}(\overline{M})$, which is

$$@_A[\mathrm{role}^i_A(\overline{M})] \Rightarrow @_B[\mathsf{P}_\circ\ send(M, A)],$$

assuming that the protocol step q requires that B, in role j, sends message M to A, in role i. We should therefore require $\mathrm{auth}^{i,j,q}_{A,B}(\overline{M})$ to hold whenever step q is considered essential. In the case of NSPK, we could require for honest A acting as initiator, the authentication of the responder at step 2 using $\mathrm{auth}^{Init,Resp,2}_{A,B}(A, B', N_1, N_2)$, and for honest B acting as responder, the authentication of the initiator at step 3 using $\mathrm{auth}^{Resp,Init,3}_{B,A}(A', B, N_1, N_2)$. The latter fails in the man-in-the-middle attack to NSPK [18], as we explain below.

4.3 Analysis

To evaluate the cogency of our approach, we have analyzed the well-known Needham-Schroeder Public-Key Protocol (NSPK) and the protocol NSL, the corrected version given by Lowe to prevent the man-in-the-middle attack on NSPK [18]. We have applied our logic to these and other similar examples, and have thereby been able both to:

- find the usual man-in-the-middle attack to NSPK (which results from the failure of the proof of the mentioned authentication formula), and
- show that the authentication properties hold for NSL.

A detailed account of our analysis of these and other protocols and properties can be found in [7, 8].

5 Conclusion

We have been applying our logic also to investigate general metatheoretic properties of the underlying protocol models and model simplification techniques that may contribute to the sound design of effective protocol analysis tools. Such results also help simplify the underlying protocol model and thereby simplify the analysis of properties such as the ones considered in the previous section. Namely, in [7, 8], we prove a general lemma about *secret data* that is similar to the secrecy theorems of [13, 21]. We also obtain soundness and completeness results, with respect to typical security goals, for two model-simplification techniques: *one intruder is enough*, in the lines of [12], and the *predatory-intruder*, a bound on the behavior of the intruder that goes in the direction of the trace models used in practice, e.g. [24]. While these results, mutatis mutandis, have already been shown for other particular formalisms, our logic provides a means for proving them in a general and uniform way, within the same formalism, which opens the way for further general investigations. Our formalization has also allowed us to clarify aspects of these simplification properties that are often neglected or cannot be specified in the first place (e.g. concerning principals' identities and the way security properties are established). We have also begun applying our logic to other metatheoretical investigations, such as the development of appropriate partial-order techniques that may reduce the (potentially infinite) state-space exploration involved in model-checking protocol properties (cf. [3]). This is work in progress and the first results are promising.

References

1. A. Armando and L. Compagna. Abstraction-driven SAT-based Analysis of Security Protocols. In *Proc. SAT 2003*, LNCS 2919. Springer-Verlag, 2003. Available at `http://www.avispa-project.org`.
2. A. Armando, L. Compagna, and P. Ganty. SAT-based Model-Checking of Security Protocols using Planning Graph Analysis. In *Proc. FME'2003*, LNCS 2805. Springer-Verlag, 2003.
3. D. Basin, S. Mödersheim, and L. Viganò. An On-The-Fly Model-Checker for Security Protocol Analysis. In E. Snekkenes and D. Gollmann, editors, *Proc. ESORICS'03*, LNCS 2808, pages 253–270. Springer-Verlag, 2003. Available at `http://www.avispa-project.org`.
4. D. Basin, S. Mödersheim, and L. Viganò. Constraint Differentiation: A New Reduction Technique for Constraint-Based Analysis of Security Protocols. In V. Atluri and P. Liu, editors, *Proc. CCS'03*, pages 335–344. ACM Press, 2003. Available at `http://www.avispa-project.org`.
5. C. Bodei, P. Degano, R. Focardi, and C. Priami. Primitives for authentication in process algebras. *Theoretical Computer Science*, 283(2), 2002.
6. L. Bozga, Y. Lakhnech, and M. Perin. Pattern-based abstraction for verifying secrecy in protocols. In *Proc. TACAS 2003*, LNCS 2619. Springer-Verlag, 2003.
7. C. Caleiro, L. Viganò, and D. Basin. Distributed Temporal Logic for Security Protocol Analysis. In preparation, 2004.

8. C. Caleiro, L. Viganò, and D. Basin. Metareasoning about Security Protocols using Distributed Temporal Logic. To appear, 2004.
9. I. Cervesato, C. Meadows, and P. F. Syverson. Dolev-Yao is no better than Machiavelli. In P. Degano, editor, *Proc. of WITS'00*, 8–9 July 2000.
10. I. Cervesato and P. F. Syverson. The logic of authentication protocols. In *Foundations of Security Analysis and Design*, LNCS 2171, pages 63–136. Springer-Verlag, 2001.
11. Y. Chevalier and L. Vigneron. Automated Unbounded Verification of Security Protocols. In E. Brinksma and K. G. Larsen, editors, *Proc. CAV'02*, LNCS 2404, pages 324–337. Springer-Verlag, 2002.
12. H. Comon-Lundh and V. Cortier. Security properties: two agents are sufficient. In *Proc. ESOP'2003*, LNCS 2618, pages 99–113. Springer-Verlag, 2003.
13. V. Cortier, J. Millen, and H. Rueß. Proving secrecy is easy enough. In *Proc. of CSFW'01*. IEEE Computer Society, 2001.
14. B. Donovan, P. Norris, and G. Lowe. Analyzing a library of security protocols using Casper and FDR. In *Proc. FLOC'99 Workshop on Formal Methods and Security Protocols (FMSP'99)*, 1999.
15. H.-D. Ehrich and C. Caleiro. Specifying communication in distributed information systems. *Acta Informatica*, 36:591–616, 2000.
16. T. Genet and F. Klay. Rewriting for cryptographic protocol verification. In *Proc. CADE'00*, LNCS 1831, pages 271–290. Springer-Verlag, 2000.
17. K. Lodaya, R. Parikh, R. Ramanujam, and P. Thiagarajan. A logical study of distributed transition systems. *Information and Computation*, 119(1):91–118, 1995.
18. G. Lowe. Breaking and Fixing the Needham-Shroeder Public-Key Protocol Using FDR. In T. Margaria and B. Steffen, editors, *Proc. TACAS'96*, LNCS 1055, pages 147–166. Springer-Verlag, 1996.
19. G. Lowe. A hierarchy of authentication specifications. In *Proc. CSFW'97*. IEEE Computer Society Press, 1997.
20. G. Lowe. Casper: a Compiler for the Analysis of Security Protocols. *Journal of Computer Security*, 6(1):53–84, 1998.
21. J. Millen and H. Rueß. Protocol-independent secrecy. In *2000 IEEE Symposium on Security and Privacy*. IEEE Computer Society, May 2000.
22. J. Millen and V. Shmatikov. Constraint solving for bounded-process cryptographic protocol analysis. In *Proc. of CCS'01*, pages 166–175. ACM Press, 2001.
23. F. Oehl, G. Cécé, O. Kouchnarenko, and D. Sinclair. Automatic approximation for the verification of cryptographic protocols. In *Proc. Conference on Formal Aspects of Security*, LNCS 2629. Springer-Verlag, 2003.
24. L. Paulson. The inductive approach to verifying cryptographic protocols. *Journal of Computer Security*, 6:85–128, 1998.
25. P. Ryan, S. Schneider, M. Goldsmith, G. Lowe, and B. Roscoe. *Modelling and Analysis of Security Protocols*. Addison Wesley, 2000.
26. D. Song, S. Berezin, and A. Perrig. Athena: a novel approach to efficient automatic security protocol analysis. *Journal of Computer Security*, 9:47–74, 2001.
27. G. Winskel. Event structures. In W. Brauer, W. Reisig, and G. Rozenberg, editors, *Petri Nets: Applications and Relationships to Other Models of Concurrency*, LNCS 255, pages 325–392. Springer-Verlag, 1987.