Metareasoning about Security Protocols using Distributed Temporal Logic*

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

[1] CLC, Department of Mathematics, IST, Lisbon, Portugal
[2] Department of Computer Science, ETH Zurich, Switzerland

Abstract

We introduce a version of distributed temporal logic that provides a new 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 protocol validation tools. As a first but significant example, we give a rigorous account of three such techniques.

1 Introduction

Many security protocols have been proposed to help build secure distributed systems. Given how difficult it is for humans to predict all the 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, e.g. [1, 2, 5, 11, 14, 15, 16, 17]. In this paper, we report on how a suitable version of temporal logic for communicating agents can be used as a *metalevel tool* for the analysis of security protocol models and properties.

Our starting point is the distributed temporal logic DTL of [8], which focuses on the expressibility of properties from the local point of view of each agent, and which we extend in order to also express global properties. Besides for its clean interpretation structures, which provide a nice, intuitive model of distributed systems, our reasons for using this logic are primarily threefold. First, it is well-suited for specifying and reasoning about communicating agents in distributed systems. Second, its temporal dimension can be effectively used to formalize and reason about interleaved protocol executions. Finally, its distributed dimension, with explicit agent identifiers, supports a neat formalization of the different security goals that protocols are supposed to achieve, such as different forms of authentication and secrecy.

The logic we introduce here provides an *object level tool* where we can specify and reason about specific protocols. In particular, using the logic it is possible to specify

*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 BBW Project 02.0431, "AVISPA: Automated Validation of Internet Security Protocols and Applications"

a protocol-independent distributed communication model, on top of which protocols can be formally defined and analyzed. For instance, in [3, 4] 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) [11]. We have thereby been able both to find the usual man-in-the-middle attack to the 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 the mere ad hoc 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, as we discuss in this paper. We prove here a general lemma about *secret data* that is similar to the secrecy theorems of [7, 13]. 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 [6], 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. [15]. 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 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. [2]). This is work in progress and the first results are promising.

We proceed as follows. In §2 we introduce our distributed temporal logic. Using the logic, in §3, we define a protocol-independent distributed communication model, on top of which protocols and security goals can be formalized and analyzed, as shown in §4. In §5 we present metalevel results, and conclude in §6 with a discussion of future work.

2 Distributed temporal logic

DTL [8] is a logic for reasoning about temporal properties of distributed systems from the local point of view of the system's 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 agents and their interaction. Herein, we introduce a minor extension of DTL tailored to allow also for the smooth formalization and proof of global properties.

The logic is defined over a *distributed signature* $\Sigma = \langle Id, \{Act_i\}_{i \in Id}, \{Prop_i\}_{i \in Id} \rangle$ of

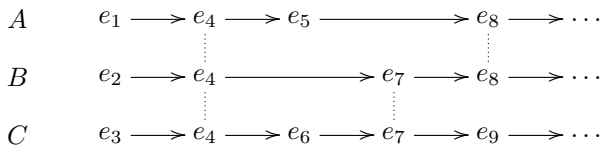


Figure 1: A distributed life-cycle for agents A, B and C.

$$\sigma_A(\emptyset) \xrightarrow{\alpha_A(e_1)} \sigma_A(\{e_1\}) \xrightarrow{\alpha_A(e_4)} \sigma_A(\{e_1, e_4\}) \xrightarrow{\alpha_A(e_5)} \sigma_A(\{e_1, e_4, e_5\}) \xrightarrow{\alpha_A(e_8)} \cdots$$

Figure 2: The progress of agent A.

a system, 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

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

for $i \in Id$, where the *local languages* $\mathcal{L}_i$ 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 about 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.

The interpretation structures of $\mathcal{L}$ are suitably labeled distributed life-cycles, built upon a simplified form of Winskel's *event structures* [18]. For brevity, we just give an outline of their definition here and refer to [3] for details. A *local life-cycle* of an agent $i \in Id$ is a pair $\lambda_i = \langle Ev_i, \rightarrow_i \rangle$, where Ev_i is the set of *local events* and $\rightarrow_i \subseteq Ev_i \times Ev_i$ is the *local successor relation*, such that $\rightarrow_i^*$ defines a well-founded total order of *local causality* on Ev_i. A *distributed life-cycle* is a family $\lambda = \{\lambda_i\}_{i \in Id}$ of local life-cycles such that $\rightarrow^* = (\bigcup_{i \in Id} \rightarrow_i)^*$ defines a partial order of *global causality* on the set $Ev = \bigcup_{i \in Id} Ev_i$ of all events. This latter condition is essential since events can be shared by several agents at communication points. We can check the progress of an agent by collecting all the local events that have occurred up to a certain point. This yields the notion of *local configuration* of an agent i: a finite set $\xi_i \subseteq Ev_i$ closed for local causality, i.e. if $e \rightarrow_i^* e'$ and $e' \in \xi_i$ then also $e \in \xi_i$. The set Ξ_i of all local configurations of an agent i is clearly totally ordered by inclusion and has $\emptyset$ as the minimal element. In general, each non-empty local configuration ξ_i is reached, by the occurrence of an event that we call $last(\xi_i)$, from the local configuration $\xi_i \setminus \{last(\xi_i)\}$. We can also define the notion of a *global configuration*: a finite set $\xi \subseteq Ev$ closed for global causality, i.e. if $e \rightarrow^* e'$ and $e' \in \xi$ then also $e \in \xi$. The set Ξ of all global configurations constitutes a lattice, under inclusion, and has $\emptyset$ as the minimal element. Clearly, every global configuration ξ includes the local configuration $\xi|_i = \xi \cap Ev_i$ of each agent i. Given $e \in Ev$, note that $e{\downarrow} = \{e' \in Ev \mid e' \rightarrow^* e\}$ is always a global configuration.

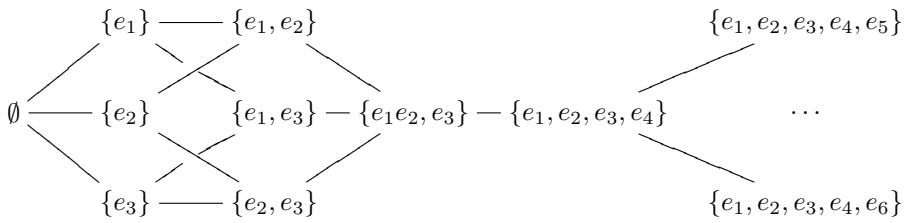


Figure 3: The lattice of global configurations.

An *interpretation structure* $\mu = \langle \lambda, \alpha, \sigma \rangle$ consists of a distributed life-cycle λ plus families $\alpha = \{\alpha_i\}_{i \in Id}$ and $\sigma = \{\sigma_i\}_{i \in Id}$ of local labeling functions. For each $i \in Id$, $\alpha_i : Ev_i \to Act_i$ associates a local action to each local event, and $\sigma_i : \Xi_i \to \wp(Prop_i)$ associates a set of local state propositions to each local configuration. We denote the tuple $\langle \lambda_i, \alpha_i, \sigma_i \rangle$ also by μ_i.

Fig. 1 illustrates the notion of a distributed life-cycle, where each row comprises the local life-cycle of one agent. In particular, $Ev_A = \{e_1, e_4, e_5, e_8, \dots\}$ and $\to_A$ corresponds to the horizontal arrows in A's row. We can think of the occurrence of event e_1 as leading agent A from its initial configuration $\emptyset$ to configuration $\{e_1\}$, and then the occurrence of event e_4 leading to configuration $\{e_1, e_4\}$, and so on; the state-transition sequence of agent A is displayed in Fig. 2. Shared events at communication points are highlighted by the dotted vertical lines. Note that the numbers annotating the events are there only for convenience since no global total order on events is in general imposed. Fig. 3 shows the corresponding lattice of global configurations.

We can then define the *global satisfaction relation* at a global configuration ξ of μ as

- $\mu, \xi \Vdash @_i(\varphi)$ if $\mu, \xi|_i \Vdash_i \varphi$;
- $\mu, \xi \nVdash \bot$;
- $\mu, \xi \Vdash \gamma \Rightarrow \delta$ if $\mu, \xi \nVdash \gamma$ or $\mu, \xi \Vdash \delta$,

where the *local satisfaction relations* at local configurations are 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 $\xi_i'' \subsetneq \xi_i' \subsetneq \xi_i$;
- $\mu, \xi_i \Vdash_i @_j[\varphi]$ if $\xi_i \neq \emptyset$, $last(\xi_i) \in Ev_j$ and $\mu, (last(\xi_i)\downarrow)|_j \Vdash_j \varphi$.

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 $\gamma \in \Gamma$. Other usual operators are defined as abbreviations, e.g. $\neg$, $\top$, $\vee$, $\wedge$, and

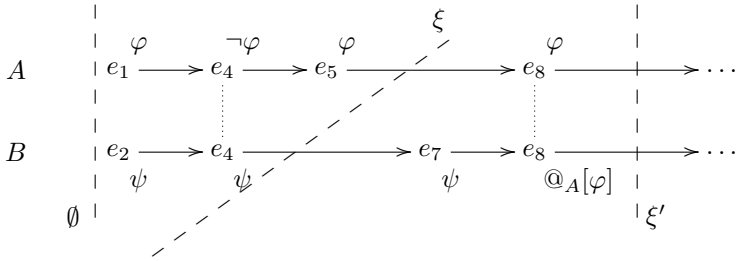


Figure 4: Satisfaction of formulas.

$\mathsf{X}\varphi$	$\equiv \bot\,\mathsf{U}\,\varphi$	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

Fig. 4 illustrates the satisfaction relation with respect to communication formulas of our running example. Clearly $\mu, \emptyset \Vdash @_B[\psi\,\mathsf{U}\,@_A[\varphi]]$, because $\mu, \xi' \Vdash @_B[@_A[\varphi]]$. Note however that $\mu, \xi \nVdash @_B[@_A[\varphi]]$, although $\mu, \xi \Vdash @_A[\varphi]$.

Rules for proving invariants by induction can be established in our logic in the standard way (see [3]).

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, e.g. in §5.2). 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

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

Note that we consider an equational signature with four sorts, namely the sort of messages and its subsorts names, nonces and keys, where we follow the usual *free-algebra assumption* so that syntactically different terms denote different messages.

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.[1]

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 the message M from the channel to principal A; and
- $leak$ — leaking of messages.

The model could, of course, be extended in many ways. For example, we could include other kinds of messages (e.g. for hashing and exponentiation), or further actions and state propositions. We will consider such extensions in future work, where we will also include *servers*, or further channels with distinct accessibility and reliability properties. For now, however, the above should be enough to abstractly present the properties of communication among principals in the analysis of security protocols.

In the network model that we define, 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 the name B' and how to produce the message 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 fresh 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 cannot learn messages in an ad hoc fashion, we specify that the *knows* propositions only hold where strictly necessary. To this end, we follow the idea underlying Paulson's inductive model [15], in accordance with the usual assumption of *perfect cryptography* (that the only way to decrypt an encrypted message is to have the appropriate key). 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 $\xi|_A \neq \emptyset$:

[1] Note that we do not explore the epistemic dimension of this knowledge.

(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., [15]). A number of useful properties follow from **(K)**, e.g. for each principal $A \in Pr$:

(K1) $@_A[knows(M_1; M_2) \Leftrightarrow (knows(M_1) \wedge knows(M_2))]$;
(K2) $@_A[(knows(M) \wedge knows(K)) \Rightarrow knows(\{M\}_K)]$;
(K3) $@_A[(knows(\{M\}_K) \wedge knows(K^{-1})) \Rightarrow knows(M)]$;
(K4) $@_A[knows(M) \Rightarrow \mathsf{G}_\circ\, knows(M)]$;
(K5) $@_A[rec(M) \Rightarrow knows(M)]$;
(K6) $@_A[spy(M) \Rightarrow knows(M)]$; and
(K7) $@_A[nonce(N) \Rightarrow knows(N)]$.

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 (K7), **(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 Modeling security protocols

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 model protocols on top of the network.

We illustrate protocol modeling by using a standard example: the (flawed) simplified Needham-Schroeder Public-Key Protocol NSPK [11], which we present as the following sequence of message exchange steps.

$$\begin{array}{llcll} (\text{step}_1) & a \to b & : & (n_1). & \{n_1; a\}_{K_b} \\ (\text{step}_2) & b \to a & : & (n_2). & \{n_1; n_2\}_{K_a} \\ (\text{step}_3) & a \to b & : & & \{n_2\}_{K_b} \end{array}$$

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 $\text{run}_A^{Init}(A', B', N_1, N_2)$:

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

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

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

In the sequel, we use $w = \langle w_1.w_2.w_3 \ldots \rangle$ to denote a (possibly infinite) sequence w composed of the elements $w_1, w_2, w_3, \ldots$, and we use $|w|$ to denote its length. Of course, $\langle\rangle$ denotes the empty sequence and $|\langle\rangle| = 0$. We assume that $|w| = \infty$ if w is infinite. We also write $w \centerdot w'$ to denote the concatenation of the two sequences, provided that the first sequence is finite.

In general, a protocol description as above may involve j name variables $a_1, \ldots, a_j$, corresponding to j distinct roles, and k nonce variables $n_1, \ldots, n_k$, and consist of a sequence $\langle \text{step}_1 \ldots \text{step}_m \rangle$ of message exchange steps, each of the general form

$$(\text{step}_q) \quad a_s \to a_r \ : \ (n_{q_1}, \ldots, n_{q_t}).\ M\,,$$

where M can involve any of the name and nonce variables. These steps prescribe a sequence of actions to be executed by each of the participants in a run of the protocol: for each role i, we have the corresponding sequence $\text{run}^i = \text{step}^i_1 \centerdot \cdots \centerdot \text{step}^i_m$ where

$$\text{step}^i_q = \begin{cases} \langle \mathit{nonce}(n_{q_1}) \ldots \mathit{nonce}(n_{q_t}).\mathit{send}(M, a_r) \rangle & \text{if } i = s, \\ \langle \mathit{rec}(M) \rangle & \text{if } i = r, \\ \langle\rangle & \text{if } i \neq s \text{ and } i \neq r. \end{cases}$$

We can easily formalize in the logic the complete execution by principal A of the run corresponding to role i of the protocol. If $\text{run}^i = \langle \mathit{act}_1 \ldots \mathit{act}_n \rangle$ then we can consider the local formula role^i_A:

$$\mathit{act}_n \wedge \mathsf{P}(\mathit{act}_{n-1} \wedge \mathsf{P}(\ldots \wedge \mathsf{P}\ \mathit{act}_1) \ldots)\,.$$

A *protocol instantiation* is a variable substitution σ such that each $\sigma(a_i) \in \mathit{Nam}$, each $\sigma(n_i) \in \mathit{Non}$, and σ is injective on name variables, i.e. if $i_1 \neq i_2$ then $\sigma(a_{i_1}) \neq \sigma(a_{i_2})$. We extend σ to messages, actions, sequences, and even formulas in the natural way. Each protocol instantiation provides a concrete sequence of actions to be executed by each participating principal. Clearly, $\sigma(\text{run}^i)$ should be executed by the principal named $\sigma(a_i)$. In general, if we denote the set of all protocol instantiations by Inst, we can define as follows the set Runs^i_A of all possible concrete runs of principal A in role i, and the set Runs_A of all its possible concrete runs in any of the j roles:

$$\mathit{Runs}^i_A = \bigcup_{\sigma \in \mathit{Inst}} \{\sigma(\text{run}^i) \mid \sigma(a_i) \in \mathit{Nam}_A\} \qquad \text{and} \qquad \mathit{Runs}_A = \bigcup_{i=1}^{j} \mathit{Runs}^i_A\,.$$

It should be clear that if $\sigma(a_i) \in \mathit{Nam}_A$ then $\mu, \xi \Vdash @_A[\sigma(\text{role}^i_A)]$ if and only if A has just completed at ξ the required sequence of actions $\sigma(\text{run}^i)$. Often, in examples, we will use $\bar{a} = \langle a_1 \ldots a_j \rangle$ and $\bar{n} = \langle n_1 \ldots n_k \rangle$, and write $\text{run}^i(\sigma(\bar{a}), \sigma(\bar{n}))$ instead of $\sigma(\text{run}^i)$, and $\text{role}^i_A(\sigma(\bar{a}), \sigma(\bar{n}))$ instead of $\sigma(\text{role}^i_A)$.

4.1 Honesty

We take an external view of the system, and consider a *protocol signature* to be a triple $\langle \mathit{Hn}, \mathit{In}, \mathit{Nam} \rangle$ where Hn and In are disjoint sets of *honest* and *intruder* principals, and $\langle \mathit{Hn} \cup \mathit{In}, \mathit{Nam} \rangle$ is a network signature such that every honest principal has exactly one name. Note that this implies that no honest agent will ever play two different roles in the same run of a protocol. Without loss of generality, we assume that $\mathit{Nam}_A = \{A\}$ for every $A \in \mathit{Hn}$. 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 $A \in \mathit{Hn}$:

(Key1) $@_A[* \Rightarrow \mathit{knows}(K_A^{-1})]$; and

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

Models of a protocol will be those network models where, furthermore, all honest principals strictly follow the rules of the protocol. That is, for every $A \in Hn$, if the local life-cycle of A is $e_1 \rightarrow_A e_2 \rightarrow_A e_3 \rightarrow_A \ldots$, then the corresponding (possibly infinite) sequence of actions

$$w(A) = \langle \alpha_A(e_1).\alpha_A(e_2).\alpha_A(e_3)\ldots \rangle$$

must be an interleaving of prefixes of sequences in $Runs_A$, but using distinct fresh nonces in each of them. Formally, we say that two sequences of actions w and w' are *independent* provided that if $w_i = \mathit{nonce}(N)$, for some $i \leq |w|$, then $w'_j \neq \mathit{nonce}(N)$ for every $j \leq |w'|$. The requirement on protocol models can now be rigorously defined. For each $A \in Hn$, there must exist a set $W \subseteq Runs_A$ of pairwise independent sequences such that for every $i \leq |w(A)|$ it is possible to choose $w \in W$, $j \leq |w|$ and $i_1 < \cdots < i_j = i$ satisfying $w(A)_{i_k} = w_k$ for all $k \leq j$.

Note that this is essentially equivalent to approaches such as [15], where the behavior of an honest agent A is defined inductively in such a way that the jth action of a sequence $w \in Runs_A$ can be executed only if the previous $j-1$ actions have already been executed, or to strands spaces [9] where essentially the same sequences of $Runs_A$ are used to model honest agents. In all cases, the intruders (*attackers* or *penetrators*) can act freely, according to the standard Dolev-Yao capabilities.

In the case of NSPK, this means that the life-cycle of each 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 such 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. There are many approaches to specifying secrecy and authentication in the literature, depending in part on the underlying model used. However, the various approaches mostly agree on the general picture. 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:

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

Of course, this property can only be expected to hold in particular situations. Given a protocol instantiation σ, assume that all the participants are honest, i.e. each $\sigma(a_i) = A_i \in Hn$ and so $Nam_{A_i} = \{A_i\}$. 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 $\sigma(\mathrm{secr}_F)$ for the relevant set of fresh nonce variables $F \subseteq \{n_1, \ldots, n_k\}$. As before, we sometimes write $\mathrm{secr}_{\sigma(F)}(\sigma(\bar{a}), \sigma(\bar{n}))$ instead of $\sigma(\mathrm{secr}_F)$.

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., [12]). 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 $\mathrm{auth}^{i,j,q}_{A,B}$, which is

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

$$@_A[\mathrm{role}^i_A] \Rightarrow \bigvee_{C \in In} @_C[\mathsf{P}_\circ\ send(M, A)], \text{ if } B \text{ is dishonest,}$$

assuming that the $\mathrm{protocol\ step}_q$ requires that a_j sends message M to a_i. Note that if we consider only one dishonest principal, as usual, this distinction vanishes, but we argue that it is essential for the sake of generality (see Proposition 5.3 below). Given a protocol instantiation σ with $\sigma(a_i) = A \in Hn$ and $\sigma(a_j) \in Nam_B$, we should therefore require $\sigma(\mathrm{auth}^{i,j,q}_{A,B})$ to hold whenever step q is considered essential. As before, we sometimes write $\mathrm{auth}^{i,j,q}_{A,B}(\sigma(\bar{a}), \sigma(\bar{n}))$ instead of $\sigma(\mathrm{auth}^{i,j,q}_{A,B})$.

In the case of NSPK, assuming for the moment that only one dishonest principal exists, we could require for honest A acting as initiator, the authentication of the responder at step 2 using $\mathrm{auth}^{Init,Resp,2}_{A,B}(A, B', N_1, N_2)$:

$$@_A[\mathrm{role}^{Init}_A(A, B', N_1, N_2)] \Rightarrow @_B[\mathsf{P}_\circ\, send(\{N_1; N_2\}_{K_A}, A)]\,,$$

and for honest B acting as responder, the authentication of the initiator at step 3 using $\mathrm{auth}^{Resp,Init,3}_{B,A}(A', B, N_1, N_2)$:

$$@_B[\mathrm{role}^{Resp}_B(A', B, N_1, N_2)] \Rightarrow @_A[\mathsf{P}_\circ\, send(\{N_2\}_{K_B}, B)]\,.$$

This last property fails in the man-in-the-middle attack to NSPK [11], as we show in [3, 4].

5 Metalevel analysis of the model

Our protocol analysis framework is based on a logic that is not specifically tailored to security protocols, and we are thus not bound to any assumption on the underlying protocol model. Rather, we can use our logic to specify and reason about such assumptions, proving different metalevel properties and simplification techniques of security protocol models within one and the same formalism in a uniform way. We develop our proofs in the context of the general network we have defined above, with explicit asynchronous communication through the channel, and where intruders are modeled as agents within the system.

As significant examples, we now give rigorous accounts of three simplification techniques for protocol models. As we discuss below, these accounts also illustrate how our approach can help clarifying a number of underlying concepts that are often left implicit, or neglected, when considering these simplifications within other approaches.

5.1 Secret data

The following lemma is a significant example of the kind of metalevel properties that any suitable network model should enjoy. Let $S \subseteq Msg$ be a set of *secret* atomic messages (names, nonces, and keys), and denote by Msg_S the set of *S-secure messages*, i.e. all messages where items from S can only appear if under the scope of an encryption with a key whose inverse is also in S. It should be clear that Msg_S contains precisely the messages that can be securely circulated in the network without danger of compromising any of the secrets in S. Indeed, $synth(analyz(Msg_S)) = Msg_S$ and $Msg_S \cap S = \emptyset$.

Lemma 5.1 (Secret Data) *Assume that $G \subseteq Pr$ is a group of principals, μ is a network model such that $\mu \Vdash \bigwedge_{A\in G} @_A[\neg\, send(M, C')]$ for every $M \notin Msg_S$ and every name C', and $\mu \Vdash \bigvee_{A\in G} @_A[* \Rightarrow \mathsf{F}\, nonce(N)]$ for every nonce $N \in S$. If*

$$\mu, \xi \Vdash \bigwedge_{B\in Pr\setminus G} @_B[\neg\, knows(M)] \text{ for every } M \notin Msg_S,$$

then also

$$\mu, \xi \Vdash \bigwedge_{B\in Pr\setminus G} @_B[\mathsf{G}_\circ\, \neg\, knows(M)] \text{ for every } M \notin Msg_S.$$

Proof: By induction on configurations $\xi' \supseteq \xi$. Assuming the base case, as given, it suffices to prove that, given $\xi' \supseteq \xi$, if $\mu, \xi' \Vdash @_B[\neg\, knows(M)]$ for every $M \notin Msg_S$ and every principal $B \notin G$, and $\xi' \cup \{e\} \in \Xi$, then also $\mu, \xi' \cup \{e\} \Vdash @_B[\neg\, knows(M)]$ for every $M \notin Msg_S$ and $B \notin G$.

Suppose, by absurdity, that $\mu, \xi' \cup \{e\} \Vdash @_B[knows(M)]$ for some $M \notin \mathcal{M}_S$ and some $B \notin G$. Then it must be the case that $e \in Ev_B$ and so the local configuration of all other principals does not change. Moreover, $\alpha_B(e)$ cannot be a sending action or the local state of B would also not change. If it is either $rec(M'')$ or $spy(M'')$ then it must be the case that $M'' \notin Msg_S$, but, since it had to have been previously sent to the channel, that is impossible. Indeed, by assumption, principals in G never send such messages, and no other principal could have sent it before. Hence, it must be a $nonce(N)$ action for some $N \in S$. But this contradicts the fresh nonce axioms because,

by assumption, N is generated by some principal in G. Thus, $\mu, \xi' \Vdash @_B[\neg \mathit{knows}(M)]$ for every $M \notin \mathcal{M}_S$, $B \notin G$ and $\xi' \supseteq \xi$, and the result follows. □

Assuming that no principal in G will ever send an S-insecure message and that all the nonces in S are freshly generated among the principals in G, if at some point the S-insecure data is unknown outside of G, then it will forever remain so. Note that the set $\mathit{Msg} \setminus \mathit{Msg}_S$ of S-insecure messages forms precisely what has been called an *ideal* in the context of strand spaces [9], whereas the set Msg_S of S-secure messages is the corresponding *coideal*, in the terminology of [7, 13]. In fact, Lemma 5.1 is a general result about the flow of data in the network, which is independent of protocols. The result can of course be used to reason about secrecy properties in protocol models, providing a result that is very similar to those found in [7, 13]. Indeed, under reasonable conditions, secrecy of generated nonces can be easily seen to hold.

Proposition 5.2 (Secrecy) *A given protocol guarantees $\sigma(\mathit{secr}_F)$ for an instantiation σ with only honest participants $\sigma(a_1) = A_1$, ..., $\sigma(a_j) = A_j$, provided that all the messages ever sent by $A_1, \ldots, A_j$ in any protocol run are $(\{K_{A_1}^{-1}, \ldots, K_{A_j}^{-1}\} \cup \sigma(F))$-secure.*

Proof: The result follows by an application of the Secret Data Lemma 5.1, using $G = \{A_1, \ldots, A_j\}$ and $S = \{K_{A_1}^{-1}, \ldots, K_{A_j}^{-1}\} \cup \sigma(F)$. Let μ be a protocol model, ξ a global configuration, and assume that $\mu, \xi \Vdash \bigwedge_{i=1}^{j} @_{A_i}[\mathsf{P}_\circ\, \sigma(\mathrm{role}_{A_i}^i)]$. The assumption that $A_1, \ldots, A_j$ will only send S-secure messages is the first precondition for the application of the lemma. The second precondition of the lemma follows immediately from the fact that all the corresponding roles of the protocol have been completed and therefore all the nonces in S are generated in μ among the principals in G.

Take the initial configuration $\emptyset$. Clearly, no principal outside G initially knows S-insecure messages. For the nonces it is trivial as they are generated in the model, for the keys it follows from the axioms **(Key1)** and **(Key2)**. By the lemma we conclude that $\mu, \emptyset \Vdash @_B[\mathsf{G}_\circ \neg \mathit{knows}(M)]$ for every $B \in \mathit{Pr} \setminus G$ and $M \notin \mathit{Msg}_S$. In particular, $\mu, \xi \Vdash \bigwedge_{B \in \mathit{Pr} \setminus \{A_1, \ldots, A_j\}} \bigwedge_{N \in \sigma(F)} @_B[\neg \mathit{knows}(N)]$. □

Note that our assumption here that all the messages ever sent by $A_1, \ldots, A_j$ in any protocol run are $(\{K_{A_1}^{-1}, \ldots, K_{A_j}^{-1}\} \cup \sigma(F))$-secure is essentially equivalent to *discreetness* in the terminology of [7, 13].

5.2 One intruder is enough

In the sequel, we distinguish between *one-intruder* and *many-intruder* protocol signatures and models, depending on whether In is a singleton or not. Indeed, most approaches to protocol analysis only consider one-intruder models. Below, we show that this simplification is adequate. In the end, it should all amount to postulating a unique intruder Z who controls the activity of all dishonest principals, by making Z inherit the initial knowledge of all of them and perform, in some compatible sequential order, all the actions each of them performed. Of course, this transformation should be transparent to honest agents.

Proposition 5.3 (One intruder is enough) *The restricted class of one-intruder models is fully representative in the following sense: any attack to a protocol in a many-intruder model can be mapped to a corresponding attack in a one-intruder model.*

Proof: Let $cp\Sigma = \langle Hn, In, Nam\rangle$ be a many-intruder signature and assume that an attack to security goal γ happens at configuration ξ of $\mu = \langle \lambda, \alpha, \sigma\rangle$. Consider the one-intruder signature $cp\Sigma' = \langle Hn, \{Z\}, Nam'\rangle$, with $Nam'_Z = \bigcup_{A\in In} Nam_A$, and build $\mu' = \langle \lambda', \alpha', \sigma'\rangle$ as follows: $\mu'_A = \mu_A$ for every $A \in Hn$; $\mu'_{Ch} = \mu_{Ch}$; and $\lambda'_Z = \langle Ev_Z, \rightarrow_Z\rangle$ where $Ev_Z = \bigcup_{A\in In} Ev_A$ and $\rightarrow_Z$ is the successor relation associated to some discrete linearization $\langle Ev_Z, \rightarrow^*_Z\rangle$ of $\langle Ev_Z, \rightarrow^*\rangle$ that has $\bigcup_{A\in In} \xi|_A$ as a local configuration, $\alpha'_Z(e) = \alpha_A(e)$, where $A \in In$ is the unique principal such that $e \in Ev_A$, and $\sigma'_Z(\emptyset) = \bigcup_{A\in In} \sigma_A(\emptyset)$. It is straightforward to check that μ' is a one-intruder model of the protocol and ξ is still a configuration. We now show that:

(i) $\mu, \xi \Vdash @_A[\varphi]$ iff $\mu', \xi \Vdash @_A[\varphi]$, for every $A \in Hn$ and every $\varphi \in \mathcal{L}_A$ that does not include communication subformulas; and

(ii) $\mu, \xi \Vdash \bigvee_{A\in In} @_A[\mathsf{P}_\circ\ act]$ iff $\mu', \xi \Vdash @_Z[\mathsf{P}_\circ\ act]$, for every action act.

Property (i) is an immediate consequence of the fact that $\mu'_A = \mu_A$ for every $A \in Hn$, if we note that, by definition, the satisfaction of a local formula without communication subformulas only depends on the local life-cycle. Property (ii) follows directly from the fact that $Ev_Z = \bigcup_{A\in In} Ev_A$, and for each $e \in Ev_Z$, $\alpha'_Z(e) = \alpha_A(e)$ where $A \in In$ is the unique such that $e \in Ev_A$. Clearly, these two properties imply that if γ is an authentication property then the attack must also appear at ξ in μ'. Indeed, if $\gamma \equiv \sigma(\mathrm{auth}^{i,j,q}_{A,B})$ with honest A, then it follows from (i) that the antecedent $@_A[\sigma(\mathrm{role}^i_A)]$ of the main implication in γ still holds at μ' and ξ since $\sigma(\mathrm{role}^i_A) \in \mathcal{L}_A$ does not include communication subformulas. As for the consequent, if B is honest then $@_B[\mathsf{P}_\circ\ send(M, A)]$ must also fail at μ' and ξ, again by using (i), given that $\mathsf{P}_\circ\ send(M, A) \in \mathcal{L}_B$ does not have communication subformulas. If B is dishonest then the failure of $\bigvee_{C\in In} @_C[\mathsf{P}_\circ\ send(M, A)]$ at μ and ξ implies the failure of $@_Z[\mathsf{P}_\circ\ send(M, A)]$ at μ' and ξ, according to (ii).

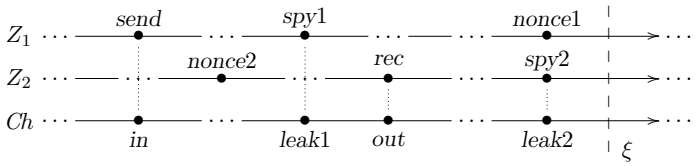
If we also note that

(iii) if $\mu, \xi \Vdash \bigvee_{A\in In} @_A[knows(M)]$ then $\mu', \xi \Vdash @_Z[knows(M)]$,

follows easily from (ii) and condition **(K)**, then it is also clear that an attack to a secrecy property can be shown to appear at ξ in μ'. In fact, if $\gamma \equiv \sigma(\mathrm{secr}_F)$ with all roles played by honest principals, then the antecedent $\bigwedge^j_{i=1} @_{A_i}[\mathsf{P}_\circ\ \sigma(\mathrm{role}^i_{A_i})]$ of the main implication in γ still holds at μ' and ξ, according to (i). To show that the consequent $\bigwedge_{B\in Pr\setminus\{A_1,\ldots,A_j\}} \bigwedge_{N\in F} @_B[\neg\, knows(N)]$ must also fail at μ' and ξ it now suffices to use either (i), for honest B, or (iii) for dishonest B. □

Fig. 5 gives a visual account of this transformation. Note that in this case we chose a linearization where $nonce_1$ happened before $nonce_2$, but any other possibility would be fine, as long as the initial causal restrictions are met, namely, $nonce_1$ must occur after spy_1, and $nonce_2$ must precede the occurrence of *rec*.

The one-intruder reduction is an intuitive and widely used simplification, but its proof can be enlightening. In fact, not only the disjunctive view on the many dishonest



can be reduced to

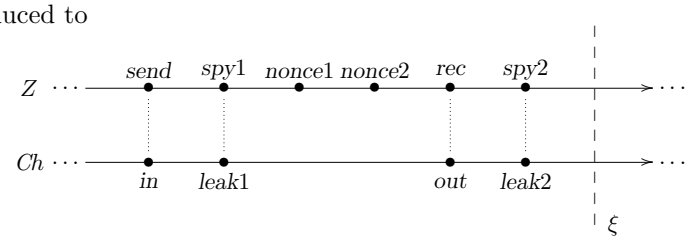


Figure 5: The one-intruder reduction.

principals can be viewed as a kind of "group intruder", but the translation also caters for the one intruder as controlling all of them, for which our characterization of authentication in the presence of many intruders was essential. The result is similar to part of the one obtained in [6]. There, however, the intruder was modeled as an abstract entity, with an obvious counterpart on the way security properties were expressed. Our result takes this same view from inside the system since, for that sake, we model the intruder as a concrete entity, namely one (or several) of the principals.

5.3 The predatory intruder

Among the possible one-intruder models of a given protocol, many will feature a rather passive intruder. Any attack that can happen under these circumstances should certainly also be possible to achieve by a more effective intruder. Herein, we show that we can restrict attention to models where the intruder is relentlessly and effectively committed to his task, namely:

- he spies every message sent by an honest agent immediately after it arrives to the channel, and that is all the spying he does:

$$@_{Ch}[@_Z[spy(M)] \Leftrightarrow \mathsf{Y} \bigvee_{A \in Hn} @_A[\bigvee_{B' \in Nam} send(M, B')]]\,;$$

- he never bothers receiving messages (he has already spied them):

$$@_Z[\neg\, rec(M)]\,;$$

- he only sends messages to honest agents, and he manages to send every message just immediately before the honest agent gets it:

$$@_Z[\neg\, send(M, Z')] \quad \text{and} \quad @_Z[send(M, A) \Rightarrow @_{Ch}[\mathsf{X}\ @_A[rec(M)]]]\,.$$

We call any one-intruder model fulfilling these requirements a *predatory-intruder* protocol model. To show that this restriction is adequate, we need to be able to transform every one-intruder model into an attack-preserving predatory-intruder model. The transformation should amount to purging all the (possibly erratic) *old* interactions of the intruder, and introducing *new* timely interactions according to the predatory-intruder requirements, still without changing anything from the point of view of honest agents.

Proposition 5.4 (The predatory-intruder) *The restricted class of predatory-intruder models is fully representative in the following sense: any attack to a protocol in a one-intruder model can be mapped to a corresponding attack in a predatory-intruder model.*

Proof: Let $cp\Sigma = \langle Hn, \{Z\}, Nam\rangle$ be a one-intruder protocol signature and assume that an attack to security goal γ happens at ξ of $\mu = \langle \lambda, \alpha, \sigma\rangle$. Consider the sets $\mathrm{Old} = Ev_Z \cap Ev_{Ch}$, $\mathrm{Succ} = \bigcup_{A\in Hn}\{e \in Ev_A \mid \alpha_A(e) = send(\_)\}$, $\mathrm{Orig}(M, B') = \{e \in Ev_{Ch} \mid \alpha_{Ch}(e) = in(M, B')\}$ with $M \in Msg$ and $B' \in Nam$, $\mathrm{Pred} = \bigcup_{A\in Hn}\{e \in Ev_A \mid \alpha_A(e) = rec(M),\ \{e' \in \mathrm{Orig}(M, A) \mid e' \rightarrow^+_{Ch} e\} \subseteq \mathrm{Old}\}$, and $\mathrm{New} = \{s(e) \mid e \in \mathrm{Succ}\} \cup \{p(e) \mid e \in \mathrm{Pred}\}$. Define the model $\mu' = \langle \lambda', \alpha', \sigma'\rangle$ as follows:

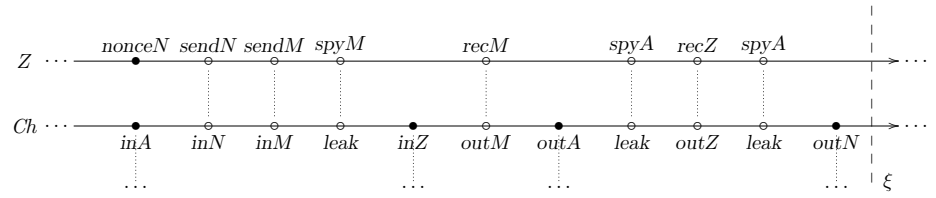
- $\mu'_A = \mu_A$ for every $A \in Hn$;
- $\lambda'_{Ch} = \langle Ev'_{Ch}, \rightarrow'_{Ch}\rangle$ with $Ev'_{Ch} = (Ev_{Ch} \setminus \mathrm{Old}) \cup \mathrm{New}$ and $\rightarrow'_{Ch}$ the successor relation obtained from $\rightarrow^*_{Ch}$ by letting $e \rightarrow'_{Ch} s(e)$ for every $e \in \mathrm{Succ}$, and $p(e) \rightarrow'_{Ch} e$ for every $e \in \mathrm{Pred}$, $\alpha'_{Ch}(e) = \alpha_{Ch}(e)$ for $e \in Ev_{Ch} \setminus \mathrm{Old}$, $\alpha'_{Ch}(s(e)) = leak$ and $\alpha'_{Ch}(p(e)) = in(M, A)$ if $\alpha_{Ch}(e) = out(M, A)$;
- $\lambda'_Z = \langle Ev'_Z, \rightarrow'_Z\rangle$ with $Ev'_Z = (Ev_Z \setminus \mathrm{Old}) \cup \mathrm{New}$ and $\rightarrow'_Z$ any successor relation compatible with $\rightarrow'_{Ch}$ on New that guarantees that every $e \in Ev_Z \setminus \mathrm{Old}$ with $\alpha_Z(e) = nonce(N)$ precedes any $p(e)$ with $\alpha_{Ch}(e) = out(M, A)$ and N occurring in M, $\alpha'_Z(e) = \alpha_Z(e)$ for $e \in Ev_Z \setminus \mathrm{Old}$, $\alpha'_Z(s(e)) = spy(M)$ if $\alpha_{Ch}(e) = in(M, B')$ and $\alpha'_Z(p(e)) = send(M, A)$ if $\alpha_{Ch}(e) = out(M, A)$, and $\sigma'_Z(\emptyset) = \sigma_Z(\emptyset)$.

It is straightforward to check that μ' is a predatory-intruder model of the protocol. Take now the global configuration $\xi' \in \Xi'$ such that $\xi'|_A = \xi|_A$ for every $A \in Hn$, $\xi'|_{Ch} \setminus \mathrm{New} = \xi|_{Ch} \setminus \mathrm{Old}$ and $\xi'|_Z \setminus \mathrm{New} = \xi|_Z \setminus \mathrm{Old}$, plus $\xi' \cap \mathrm{New} = \{p(e) \mid e \in \xi \cap \mathrm{Pred}\} \cup \{s(e) \mid e \in \xi \cap \mathrm{Succ}\}$. We now show that:

(i) $\mu, \xi \Vdash @_A[\varphi]$ iff $\mu', \xi' \Vdash @_A[\varphi]$, for every $A \in Hn$ and every $\varphi \in \mathcal{L}_A$ that does not include communication subformulas; and

(ii) $\mu, \xi \Vdash @_Z[\mathsf{P}_\circ\, send(M, A)]$ if $\mu', \xi' \Vdash @_Z[\mathsf{P}_\circ\, send(M, A)]$, for every $A \in Hn$ and message M.

Property (i) follows from the fact that $\mu'_A = \mu_A$ for every $A \in Hn$. Property (ii) is an outcome of the fact that the only *send* actions of the predatory Z are on $p(e)$ events. Therefore, it must be the case that $e \in \xi \cap \mathrm{Pred}$ is a *out* event preceded by a corresponding *in* event $e' \in \xi$. Clearly, e' is an origination event for the message and so, by definition of Pred, $e' \in \mathrm{Old}$. Therefore $e' \in Ev_Z$ and obviously it must be the case that $\alpha_Z(e')$ is the *send* action we were looking for. Using the two, we can show that if γ is an authentication property then the attack also appears at ξ' in μ'. If we also prove:



can be reduced to

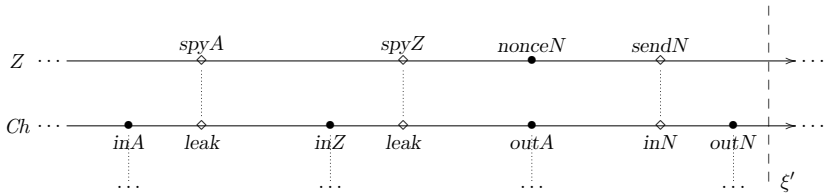


Figure 6: The predatory-intruder reduction.

(iii) if $\mu, \xi \Vdash @_Z[knows(M)]$ then $\mu', \xi' \Vdash @_Z[knows(M)]$,

then an attack can be shown to appear at ξ' in μ' also in the case that γ is a secrecy goal. Property (iii) follows from the fact that the initial knowledge of the predatory-intruder is exactly the same, he does exactly the same *nonce* actions, and spies every message at least as early as the original intruder received or spied it. □

Fig. 6 gives a visual account of the transformation. The *old* events are represented by ∘ on the first model, whereas the *new* events are represented by ⋄ in the second model. Clearly both *inA* and *inZ* are *successor* events, but only *outN* is a *predecessor* event since *outA* is preceded by *inA*, coming from an honest principal. Note also that *nonceN* could be ordered in other ways, but always before *sendN*.

The predatory-intruder reduction points already towards the inductive trace models of protocols (see [15] and also [2], for example), where the communication channel is abstracted away and "replaced" by the intruder. A nice side-effect of this reduction is that the sending actions of the intruder can really be bound by the possible shapes that honest principals can receive in protocol roles, as is commonly assumed.

Corollary 5.5 *The restricted class of one-intruder models where the intruder only ever sends messages according to the protocol description is fully representative.*

6 Conclusion

As illustrated by the above examples, the logic we have introduced here provides a powerful metalevel tool for the analysis of security protocol models and their properties. We have 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. [2]). This is work in progress and the first results are promising. Further ongoing work is

the application of our logic for reasoning about protocol composition, as well as the development of a calculus for the logic.

References

1. A. Armando and L. Compagna. Abstraction-driven SAT-based Analysis of Security Protocols. In *Proc. SAT 2003*, LNCS 2919. Springer-Verlag, 2003.
2. D. Basin, S. Mödersheim, and L. Viganò. An On-The-Fly Model-Checker for Security Protocol Analysis. In *Proc. ESORICS'03*, LNCS 2808. Springer-Verlag, 2003.
3. C. Caleiro, L. Viganò, and D. Basin. Distributed Temporal Logic for Security Protocol Analysis. In preparation, 2004.
4. C. Caleiro, L. Viganò, and D. Basin. Towards a Metalogic for Security Protocol Analysis (extended abstract). In *Proceedings of Comblog'04*, 2004. To appear.
5. Y. Chevalier and L. Vigneron. Automated Unbounded Verification of Security Protocols. In *Proc. CAV'02*, LNCS 2404. Springer-Verlag, 2002.
6. H. Comon-Lundh and V. Cortier. Security properties: two agents are sufficient. In *Proc. ESOP'2003*, LNCS 2618. Springer-Verlag, 2003.
7. V. Cortier, J. Millen, and H. Rueß. Proving secrecy is easy enough. In *Proc. CSFW'01*. IEEE Computer Society, 2001.
8. H.-D. Ehrich and C. Caleiro. Specifying communication in distributed information systems. *Acta Informatica*, 36:591–616, 2000.
9. F. J. T. Fábrega, J. C. Herzog, and J. D. Guttman. Honest ideals on strand spaces. In *Proc. CSFW'98*. IEEE Computer Society, 1998.
10. F. J. T. Fábrega, J. C. Herzog, and J. D. Guttman. Strand spaces: Proving security protocols correct. *Journal of Computer Security*, 7:191–230, 1999.
11. G. Lowe. Breaking and Fixing the Needham-Shroeder Public-Key Protocol Using FDR. In *Proc. TACAS'96*, LNCS 1055. Springer-Verlag, 1996.
12. G. Lowe. A hierarchy of authentication specifications. In *Proc. CSFW'97*. IEEE Computer Society Press, 1997.
13. J. Millen and H. Rueß. Protocol-independent secrecy. In *2000 IEEE Symposium on Security and Privacy*. IEEE Computer Society, 2000.
14. J. Millen and V. Shmatikov. Constraint solving for bounded-process cryptographic protocol analysis. In *Proc. CCS'01*. ACM Press, 2001.
15. L. Paulson. The inductive approach to verifying cryptographic protocols. *Journal of Computer Security*, 6:85–128, 1998.
16. P. Ryan, S. Schneider, M. Goldsmith, G. Lowe, and B. Roscoe. *Modelling and Analysis of Security Protocols*. Addison Wesley, 2000.
17. 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.
18. G. Winskel. Event structures. In *Petri Nets: Applications and Relationships to Other Models of Concurrency*, LNCS 255. Springer-Verlag, 1987.