Information based reasoning about security protocols

R. Ramanujam and S. P. Suresh [1]

The Institute of Mathematical Sciences
C.I.T. Campus, Chennai 600 113, India.

Abstract

We propose a notion of *information based abstraction* for the logical study of security protocols and study how protocol actions update agents' information. We show that interesting security properties of Needham-Schroeder like protocols can be verified automatically.

1 Background

The design of *cryptographic protocols* involves complicated exchanges of messages between agents to try and achieve secrecy of information and authenticity of communication. Despite considerable ingenuity on the part of the designers of these protocols, many possible attacks are often discovered later. Interestingly, most of these attacks are independent of the encryption schemes used, but rely on logical design flaws, and use intruders' abilities to replay old messages, forge addresses etc. Therefore, in recent years, a number of researchers have proposed formal verification of such protocols.

The BAN logic [BAN89] is widely held to be the initiator of logical studies of security protocols. It proposed a transformation of security protocols to a special form and then used special rules to analyze them. The logic proposed was a modal logic of belief, based on notions of trust between principals. While it was criticized extensively for its inability to express certain events ([BM93], [Nes90]), it provided a basis on which refinements like ([Bie90], [Mo99b]) could be built.

Approaches based on theorem proving ([Bol97], [Pau98]), as well as model checking ([Mea92], [Low96], [KW96], [MCJ97]), have been used for security protocol verification. Invariably, while the advantage of using formal logics for this purpose is that it provides a good framework for detailed analysis of the

[1] E-mail: {jam,spsuresh}@imsc.ernet.in

structure of such protocols, the difficulty is that logics are useful only when the semantics is precise, and most of the problems seem to relate to precise and detailed modelling of notions like authentication, trust etc.

A host of approaches based on process calculi ([AG97]), term rewriting systems ([GK99], [JRV00]), tree automata ([Mo99a], [Gou00]), multiset rewriting ([DLMS99], [CDLMS99], [DM99]) etc address the question of modelling protocols. These differ on what set of messages an intruder can construct at any step of the protocol, assumptions about the environment, the capabilities available to intruders, etc. The approach of [DMTY97] studies the roles played by principals in leaking information to intruders and uses them to generate inference rules then used in automatic verification.

In this paper, we attempt to combine the simple and precise features used in these recent models of security protocols with the older style of BAN-like logics [GNY90], [SvO94]. We do not use epistemic notions like belief, trust etc, but present an *information based abstraction* for modelling the reasoning that is typical of security protocols. Though we do present a verification result, our emphasis is not so much on finding flaws in specific protocols, but rather on setting up a logical framework with very simple primitives which is easy to reason in. We believe that the approach contains the rudiments for reasoning about both authentication protocols as well as access control models [Mc94] in one framework; however, in this account, we focus only on message exchanges meant to ensure secrecy and authentication.

To ilustrate the essential features of this abstraction, consider the following (very) simple protocol between two principals A and B.

$$A \to B : \{x\}_B$$

$$B \to A : \{x\}_A$$

This is read as follows: A sends to B her secret x encrypted with B's public key. It is assumed that A has 'generated' a 'fresh' instance of x and hence can assume that no agent in the system has access to x. After the first message, A cannot be sure that B receives the message; a hacker H may well *block* the message. However, there is something that A can be sure of: under the perfect encryption assumption, only someone who has access to the inverse key of B's public key can decipher the ciphertext and get x. Thus if B's inverse has not been leaked, A can be sure that only B can first decode the message.

Coming to the second message, it stands for B sending a message to A with content x encrypted with A's public key. However, A has no way of determining that the message does come from B, since a hacker could be pretending to be B. Nevertheless, from the fact that the message content is x, A can be sure that B has indeed got the information x.

Thus if A gets the second message at all (since a hacker could block it), then A gets the information that B has seen x. On the other hand, B does not have the information that A has x, since a hacker could be posing as A.

Consider the following *attack*:

$$\begin{aligned} A \to B(C) &: \{x\}_B \\ C \to B &: \{x\}_B \\ B \to C &: \{x\}_C \\ (C)B \to A &: \{x\}_A \end{aligned}$$

The hacker C first blocks A's message. Though C cannot decrypt the message, he can pass it on to B as his own. Given such a possibility, B cannot tell whether the message came from A or C and hence does not have the information that A has x. Now B follows the protocol and responds with the secret to C. At this point, C has learnt the secret and can further mislead A by pretending to be B and 'confirming' to A as required by the protocol.

Note the role of the secret x; when sent by A encrypted with B's key, it serves to inform A that any later receipt of x from anyone adds the information that B has seen x. Indeed, roughly speaking, in cryptographic protocols, this is the typical way one agent gains access to information about another agent's state.

The model we propose in this paper concentrates on this aspect of communications. We define possible information states of systems over a vocabulary of secrets and define rules for how such information can be updated. Any security protocol can then be seen as generating runs of such an information transition system. However, along with the intended run, there can be many unintended variants due to hacks, leaks etc. We would like to ensure that all these variants do satisfy a given security specification. We propose a propositional modal logic with information modalities in which such specifications can be expressed. The semantics of the logic mirrors the structure of information transition systems closely enough so that verifying that a protocol satisfies its logical specification is easy.

The paper is structured as follows. In the next section, we present the main semantic structure of information transition systems and explain the design choices that underlie the definitions. We then show how security protocols can be 'compiled' into the low-level transition systems. Later, we present our logic with information modalities, show examples of reasoning in the logic and present the main result that the verification problem is decidable. We conclude the paper with a discussion of issues left unstudied here.

2 Information systems

We fix a finite set Ag of *agents* in the system. Ag is intended to include the principals of protocols as well as the intruders. (As we will see later, much of the reasoning can be carried out with the assumption of one intruder, but we will keep the generality of this framework for convenience.) We use $A, B, \ldots$ etc to denote agents in Ag. (Without prejudice, we will refer to an agent's information as "its" information from now on.)

We further fix P, an at most countable set of *basic information terms*. These can be thought of as basic secrets. We will treat them as atomic propositions that can be true or false. However, in the context of most protocols, these will stand for **nonces**; we may think of the proposition asserting that such a nonce has been generated 'afresh'. A **locality map** $\lambda : P \to Ag$ will be used to specify which secret is local to which agent. Thus, systems will be parametrized by the tuple (Ag, P, λ). Let P_A denote the set $\{p \in P \mid \lambda(p) = A\}$.

From P, the set of basic information terms, we can build a set of complex terms by the following syntax. $\mathcal{I}$, the set of all information terms (items), is given by:

$$\theta \in \mathcal{I} ::= p \in P \mid (\theta)^A \mid \{\theta_1, \ldots \theta_k\}_A \ (k > 0) \mid A : \theta$$

where $A \in Ag$. $(\theta)^A$ stands for the information θ received from an agent who is possibly A. The term $A : \theta$ attests to the information that A has θ (in its database). $\{\theta_1, \ldots \theta_k\}_A$ refers to a set of information terms together encrypted with A's public key. This serves two purposes: not only will this enable only A to 'see' the contents of this item, but will also provide the guarantee that once A sees one of them, it will see them all. We refer only to public keys here, but the framework can be easily extended to include shared keys as well.

For an information item θ and $A \in Ag$, we define $ST_A(\theta)$, the set of *subterms* visible to A, in an inductive manner as follows:

- $ST_A(p) = \begin{cases} \{p, A : p\} & \text{if } \lambda(p) = A \\ \emptyset & \text{otherwise} \end{cases}$
- $ST_A((\theta)^B) = \{(\theta)^B\} \cup \{(\theta')^B \mid \theta' \in ST_A(\theta)\}$
- $ST_A(\theta) = \begin{cases} \{\theta\} \cup ST_A(\theta_1) \cup \ldots, \cup ST_A(\theta_k) & \text{if } A = B \\ \{\theta\} & \text{otherwise} \end{cases}$
  where $\theta = \{\theta_1, \ldots, \theta_k\}_B$.
- $ST_A(B : \theta) = \begin{cases} \{B : \theta\} \cup \{\theta', B : \theta' \mid \theta' \in ST_B(\theta)\} & \text{if } A = B \\ \{B : \theta\} \cup \{B : \theta' \mid \theta' \in ST_B(\theta)\} & \text{otherwise} \end{cases}$

For $\sigma \subseteq \mathcal{I}$, define $ST_A(\sigma) \stackrel{\text{def}}{=} \{\theta' \mid \theta' \in ST_A(\theta) \text{ for some } \theta \in \sigma\}$.

Definition 2.1 *$\sigma \subseteq \mathcal{I}$ is defined to be an* **information state** *if $ST_A(\sigma) \subseteq \sigma$, for every $A \in Ag$.*

Information states can be seen as a complete description of all the information available to agents in the system. Note that if $A : \theta \in \sigma$ then $\theta \in \sigma$. This reflects our intention that agents have only definite information in any state. Let Σ denote the set of all information states. We will often refer to information states simply as states.

Definition 2.2 *Let $\sigma \in \Sigma$. The* database *of A in σ, denoted $A : \sigma$ is defined*

by: $A:\sigma \stackrel{\text{def}}{=} \{\theta, A:\theta \mid A:\theta \in \sigma\}$. *The information* purportedly from A *in* σ, *denoted* σ^A *is defined by:* $\sigma^A \stackrel{\text{def}}{=} \{\theta \mid (\theta)^A \in \sigma\}$. *The information* meant for A *in* σ, *denoted* $\{\sigma\}_A$, *is defined to be the least state containing* $\{\theta \mid \{\theta\}_A \in \sigma\}$.

Proposition 2.1 *For all states* σ, $A:\sigma$ *and* σ^A *are also states.*

The set of *legal message items* for A in state σ is the least subset X of $\mathcal{I}$ such that:

- $A:\sigma \subseteq X$ and
- if $\theta_1 \in X, \ldots, \theta_k \in X$ for $k > 0$, then for all $B \neq A$, $\{\theta_1, \ldots, \theta_k\}_B \in X$.

These are the messages that A could 'legitimately' send in that state.

On the other hand, if A were an intruder, it could send any message constructed from the information it has in its database. The set of *general message items* for A in state σ is the least subset X of $\mathcal{I}$ such that:

- $A:\sigma \subseteq X$,
- if $\theta \in X$, then for any B, $B:\theta$ and $(\theta)^B$ are also in X, and
- if $\theta_1, \ldots \theta_k \in X$ for $k > 0$, then for any B, $\{\theta_1, \ldots, \theta_k\}_B \in X$.

Note that an intruder cannot construct the basic secrets of other principals, unless it has already acquired them by explicit communication.

We now define **information updates**. For this, we first consider a set of **extended names**. Let $Ho, Ha \subseteq Ag$ be given such that $Ho \neq \emptyset, Ha \neq \emptyset$ and $Ho \cap Ha = \emptyset$. Ho stands for the 'honest' principals and Ha for 'hackers'. The set of extended names Ex is given by: $Ex = Ho \cup Ha \cup \{(C)A \mid A \in Ho, C \in Ha\} \cup \{A(C) \mid A \in Ho, C \in Ha\}$. Here, $(C)A$ stands for the intruder C forging A's address and sending a message in A's name. $A(C)$ stands for the intruder C intercepting a message meant for A and blocking A from getting it.

The *update alphabet* (also referred to as action alphabet) is the set $\mathcal{U} = P \cup Snd \cup Rec$, where Snd, the 'send' alphabet, and Rec, the 'receive' alphabet are defined as follows.
$Snd = \{(A!B, \theta), ((C)A!B, \theta) \mid A \in Ho, B \in (Ho \cup Ha), C \in Ha\}$.
$Rec = \{(A?B, \theta), (A?B(C), \theta) \mid B \in Ho, A \in (Ho \cup Ha), C \in Ha\}$.

We read $(X!B, \theta)$ as a message sent by X, intended for B, with content θ. Similarly, $(A?X, \theta)$ is a message received by X, with A in the "from" address and content θ. $p \in P$ can be thought of as notation for $new(p)$, meaning that the agent $\lambda(p)$ has 'generated' a fresh instance of secret p.

We now have some definitions that relate to the process by which an agent A sends information out to another agent B and obtains confirmation that B has indeed got it. Crucially, when A sends $\{p, q\}_B$ and later sees $(p)^C$, it confirms not only that B has got p but also that B has got q. This gets tricky when we consider terms like $\{\{p, q\}_B, \{p, r\}_C\}_D$.

We first impose a graph structure on information terms. For this, define

$\theta_1 \overset{A}{\Rightarrow} \theta_2$ iff $\theta_2 \in ST_A(\theta_1)$. Let $\Rightarrow \overset{\text{def}}{=} (\bigcup_{A \in Ag} \overset{A}{\Rightarrow})^*$. The $\Rightarrow$-maximal elements of $\mathcal{I}$ are of the form p or $(\cdots((p)^{A_1})\cdots)^{A_k}$ for some $p \in P$, $k > 0$, and $A_1, \ldots, A_k \in Ag$.

While the chains given by maximal elements as above indicate possible sequences of communications, they are not of much use in the process of confirmation. For instance, if A sends $\{p\}_B$ and later receives $(((p)^C)^D)^E$ from someone, this confirms the fact that B has got p, but not that B has $(p)^A$, or even that B has p in its database. This is because B cannot be sure that p was indeed sent by A, and must consider the possibility of $(p)^D$ for any agent D. Hence A can only infer that B got p from 'someone' and would assert it in its database. Suppose $B : p^*$ stands for the information that B received p from some agent (possibly by a sequence of communications); then $A : B : p^*$ could be asserted on confirmation. In effect, we get $A : B : (p)^D$ for any D, since none of this is definite. Rather than clutter up states with such items. we prefer to add a new syntactic construct $(\theta)^*$, with the remark that it can be eliminated.

Let $(\theta)^*$ be an item whenever θ is an item; extend the definition of ST_A by: $ST_A((\theta)^*) = \{(\theta)^*\} \cup \{(\theta')^* \mid \theta' \in ST_A(\theta)\}$. In addition, $\theta^* \in ST_A((\theta)^B)$, for any A, B. Note that Proposition 2.1 is still true under this extension.

For $\theta \in \mathcal{I}$ and $X \subset \mathcal{I}$ define $\uparrow(\theta, X) \overset{\text{def}}{=} \{\theta'' = \{\theta_1, \ldots, \theta_k\}_A \mid \theta \Rightarrow \theta'' \Rightarrow \theta'$, where $\theta' \in X$, $A \in Ag, \theta_1, \ldots, \theta_k \in \mathcal{I}\}$.

Definition 2.3 • *We say that p is* fresh *for A in an information state σ iff: $p \in P_A$, $A : p \in \sigma$ and for all $\theta \in \sigma$ such that $\theta \Rightarrow p$, the only agent mentioned in θ is A.*

- *p marks θ for A in σ iff:*
  - *$A \in Ho$ and $p \in P_A$,*
  - *$\theta \Rightarrow p$ and p occurs only encrypted in θ,*
  - *for all $A : \theta' \in \sigma$, $\theta' \not\Rightarrow (p)^*$,*
  - *$A : \theta \in \sigma$, and for all $A : \theta'' \in \sigma$ such that $\theta'' \Rightarrow p$, $\theta \Rightarrow \theta''$.*
- *A confirms X in σ iff for all $\theta = \{\theta_1, \ldots, \theta_k\}_C \in X$, $A : C : (\theta)^* \in \sigma$.*

Definition 2.4 *An* **update relation** *R_u is a subset of $(\Sigma \times \mathcal{U} \times \Sigma)$ which satisfies the following conditions:*

- *If $(\sigma, (X!Y, \theta), \sigma') \in R_u$ or $(\sigma, (X?Y, \theta), \sigma') \in R_u$, then for all $A \in Ag$, $(A : \sigma') \subseteq (A : \sigma) \cup ST_A(\theta)$.*
- *If $(\sigma, p, \sigma') \in R_u$ (where $\lambda(p) = A$), then p is fresh for A in σ' and for all θ such that $\theta \not\Rightarrow p$, $\theta \in \sigma$ iff $\theta \in \sigma'$.*
- *If $(\sigma, (A!B, \theta), \sigma') \in R_u$, then*
  - *for all $D \in Ho$ such that $D \neq A$, $D : \sigma' = D : \sigma$,*
  - *θ is a legal message term for A in σ,*
  - *$A : \sigma' = A : \sigma \cup ST_A(\theta)$, and*
  - *for every $C \in Ha$, $\theta \in C : \sigma'$.*

- *If* $(\sigma, (A?B, \theta), \sigma') \in R_u$, *then*
  - *for all* $D \in Ho$ *such that* $D \neq B$, $D : \sigma' = D : \sigma$,
  - $B : \sigma' = B : \sigma \cup ST_B((\theta)^A)$, *and*
  - *for all* p *such that* $((\theta)^A \stackrel{B}{\Rightarrow} (p)^*$ *and* p *marks* θ' *in* $\sigma)$, B *confirms* $\uparrow (\theta', X)$ *in* σ', *for some* $X \subseteq Y$, *where* Y *is the set of minimal encrypted subterms of* θ' *containing* p.
- *If* $(\sigma, ((C)A!B, \theta), \sigma') \in R_u$, *then* θ *is a general message term for* C *in* σ *and for every* $D \in Ho$, $D : \sigma = D : \sigma'$.
- *If* $(\sigma, (A?B(C), \theta), \sigma') \in R_u$, *then for every* $D \in Ho$, $D : \sigma = D : \sigma'$.

Definition 2.5 *An* **information system** $\mathcal{S}$ *over* $\mathcal{U}$ *is given by a pair* (Γ, R) *where* $\Gamma \subseteq \Sigma$ *is the set of states of* $\mathcal{S}$ *and* R *is any update relation on* Γ.

A sequence $\rho \in \mathcal{U}^*$ is *admissible for* $\mathcal{S}$ if it defines a run on $\mathcal{S}$. Let $Adm(\mathcal{S})$ be the set of all sequences admissible for $\mathcal{S}$.

3 Protocol descriptions

Security protocols are typically specified as sequences of communications of the form $A \longrightarrow B : M$, where M is a message which is typically a nonce or a set of nonces, possibly encrypted with B's public key. This is an abstract description that hides details like whether messages are delivered instantaneously or suffer delays, whether they are delivered in order or out of order etc. More crucially, exception handling details like what an agent should do when it expects a message of a kind and gets another one, are left implicit. Of course, this has a bearing on the verification of such protocols. Nevertheless, crucial design problems can be captured even at this level of abstraction.

While retaining this model of a protocol, we focus our attention on how agents' information gets updated during the course of protocol execution, including hacked variants of admissible sessions. For this purpose, we generalize the message alphabet to contain all information terms. Therefore it is possible for the protocol to include communications like: $A \longrightarrow B : \{(C : p)^D\}_B$. These correspond to terms like D *said* that C has p [BAN89]. While most extant protocols do not demand such a sophisticated message mechanism, many do incorporate *names* inside messages, which constitute information terms. As we will see, the full generality of information terms gives rise to more mutual information properties being realised by protocols.

We will make a crucial assumption that *the designated principals always follow the protocol*. Thus system behaviours will be constructed in such a way that 'honest' principals take actions according to protocol and update their information according to set rules, whereas intruders are (obviously) unconstrained thus.

Formally, a protocol alphabet Π resembles the set of extended names. It is parametrized by a pair (Ho, Ha) (for convenience, assume: $Ho \cup Ha = Ag$),

and is given by the set $\Pi \stackrel{\text{def}}{=} \{X \longrightarrow Y : \theta \mid \theta \in \mathcal{I}, X \in \{A, (C)A\}, Y \in \{B, B(C)\}, A, B \in (Ho \cup Ha), C \in Ha\}$. A communication $\chi = X \longrightarrow Y : \theta$ is said to be principal if $X, Y \in Ho$.

Since we wish to consider hacked variants of messages, we need to specify how an intruder can modify the content of messages. Given an information element θ, which is a legal message term for some principal and $C \in Ha$,we define $V_C(\theta)$ to be a set of 'C-variant' terms defined inductively as follows: $V_C(p) = \{p\} \cup P_C$; $V_C((\theta')^B) = \{(\theta'')^D \mid \theta'' \in V_C(\theta'), D \in Ag\}$; $V_C(\{X\}_B) = \{\{X\}_B\}$; $V_C(B : \theta') = \{D : \theta'' \mid \theta'' \in V_C(\theta'), D \in Ag\}$. Now, for $X \subseteq \mathcal{I}$, $g_C : X \to \mathcal{I}$ is said to be a message variant map for C if for all $\theta \in X, g_C(\theta)$ is in $V_C(\theta)$.

A protocol is specified by a **principal session** which is a finite non-null sequence of principal communications $\delta = \chi_1 \ldots \chi_k$, $k > 0$. Let X be the set of all information elements used as messages in the communications in δ. Let H be the set of all agents who either send or receive messages in δ and let $T_1, T_2 \subset H$ such that $T_1 \cup T_2 \subset H$ and $T_1 \cap T_2 = \emptyset$. For $C \in Ha$, fix a message variant map $g : X \to \mathcal{I}$; a (T_1, T_2, C, g)-variant of δ is defined as follows: If $\chi_j = A \longrightarrow B : \theta$, a communication $\chi' = X \longrightarrow Y : g(\theta)$ in the protocol alphabet Π is said to be a (T_1, T_2, C, g)-variant of χ_j if ($A \in T_1$ implies $X = (C)A$), ($B \in T_1$ implies $Y = B(C)$), ($A \in T_2$ implies $X = C$), and ($B \in T_2$ implies $Y = C$). This is easily extended to sequences. A (T_1, T_2, C, g) variant of δ is a session where C plays the role of agents in T_2 in its own name, and plays the role of each agent $A \in T_1$ assuming A's name.

Let $\mathcal{M}_\delta$ denote the set of all (T_1, T_2, C, g)-variants of session δ, for all (T_1, T_2, C, g). The variants in $\mathcal{M}_\delta$ are referred to as **mono-sessions** defined by δ. A **general session** of δ is an interleaving of a finite set of mono-sessions $\mathcal{M}$ of δ. Let $\mathcal{P}_\delta$ denote the set of all general sessions of δ.

Thus, given a security protocol $\delta \in \Pi^*$, we have, associated with it, a language $\mathcal{P}_\delta \subseteq \Pi^*$. However, since we have defined updates separately for sends and receives, we need to split communications: $X \longrightarrow Y : \theta$ is translated to $(X!Y, \theta)(X?Y, \theta)$. In addition, we prefix each general session with a sequence of $new(p)$ updates for all secrets p used in that session. This is extended pointwise so that $\mathcal{P}_\delta$ defines a subset of $\mathcal{P}'_\delta$ of $\mathcal{U}^*$. The language generated by δ with respect to an information system $\mathcal{S} = (\Sigma, R)$, denoted $Lang(\mathcal{S}, \delta)$, is defined to be $\mathcal{P}'_\delta \cap Adm(\mathcal{S})$.

The following lemma is easily shown:

Lemma 3.1 *Given a protocol $\delta \in \Pi^*$ and an information system $\mathcal{S}$, there exists a nondeterministic finite state automaton N over alphabet $\mathcal{U}$ such that $\mathcal{L}(N) = Lang(\mathcal{S}, \delta)$.*

The construction of such a *protocol automaton* associated with δ is simple. We first synthesize the set of information states that the automaton needs. For this we first need to add information elements specified by δ; (for instance, if δ has a communication $A \longrightarrow B : \{\theta\}_B$, then we must add $A : \{\theta\}_B$, $A : B : (\theta)^*$

etc.) Then we close the space under subterms, and under all C-variants, for $C \in Ha$. But this makes the space infinite. We observe that the structure of δ allows us to equate behaviours in such a way that it suffices to consider, for any $C \in Ha$, C-variants from some finite subset of P_C. This allows one to work with quotiented information states, the size of which is bounded by a function of the size of δ. This gives us a finite set of automaton states, which also additionally possess control information to code up which part of the protocol is completed and what is left. Further, the structure of δ (alongwith its hacked variants) specifies the set of transitions for the automaton. Final states are identified by the completion of communications specified by δ. The details are somewhat tedious but straightforward.

As it happens, when we consider the verification problem, the security specification will also give us a finite subset of P_C for any hacker C, such that we need to consider only C-variants built from this finite set, thereby making the state space of the constructed automaton finite. Hence, we will not need the quotient construction outlined above, but a simpler one will suffice.

The automaton mentioned in the lemma is only intended to give a machine representation of general sessions, and is distinct from the kind of tree automata used by others ([Mo99a], [Gou00], for example). There, the tree automata are used to provide a succinct, automatically updatable representation of intruders' knowledge at each state. The automata studied here can be enriched so that every state contains a tree automaton to obtain such a detailed representation.

4 Logic

We now introduce the logical language in which security properties can be specified. We have a propositional modal logic, the modalities of which reflect the information structures that we have seen so far.

The logic is parameterized by (P, Ag) where P is the set of basic information terms (secrets) and Ag is the set of agents. The logic is presented in two layers: we have two syntactic categories of formulas in the logic, that of **state formulas** and that of **session formulas**.

The syntax of state formulas is given by:

$$\alpha \in \Psi ::= p \in P \mid \neg\alpha \mid \alpha \vee \beta \mid from_A\ \alpha \mid for_A\ \alpha \mid A : \alpha$$

where $A \in Ag$. Note that these formulas correspond to information elements: $from_A\ \alpha$ asserts the existence of information α that has come from an agent who is possibly A; $for_A\ \alpha$ refers to an item whose content is available only for A; the formula $A : \alpha$ attests to the definite information α being available in A's database.

The semantics formalizes the intention described above. We define the notion $\sigma \models_i \alpha$ inductively below, where σ is a state and α is a state formula:

- $\sigma \models_i p$ iff $p \in \sigma$.

- $\sigma \models_i \neg\alpha$ iff $\sigma \not\models_i \alpha$.
- $\sigma \models_i \alpha \vee \beta$ iff $\sigma \models_i \alpha$ or $\sigma \models_i \beta$.
- $\sigma \models_i from_A\ \alpha$ iff $(\sigma)^A \models_i \alpha$.
- $\sigma \models_i for_A\ \alpha$ iff $\{\sigma\}_A \models_i \alpha$.
- $\sigma \models_i A : \alpha$ iff $A : \sigma \models_i \alpha$.

The syntax of session formulas is given by:

$$\phi \in \Phi ::= \alpha \in \Psi \mid \neg\phi \mid \phi_1 \vee \phi_2 \mid \bigcirc\phi \mid \Diamond\phi$$

Let $\tau \in \Sigma^*$ and let $m = |\tau|$, $m > 0$. For $1 \leq k \leq m$, let $\tau(k)$ denote the k^{th} element of the sequence τ. The notion $\tau, k \models_s \phi$ is defined inductively as follows:

- $\tau, k \models_s \alpha$ iff $\tau(k) \models_i \alpha$.
- $\tau, k \models_s \neg\phi$ iff $\tau, k \not\models_s \phi$.
- $\tau, k \models_s \phi_1 \vee \phi_2$ iff $\tau, k \models_s \phi_1$ or $\tau, k \models_s \phi_2$.
- $\tau, k \models_s \bigcirc\phi$ iff $k < m$ and $\tau, k+1 \models_s \phi$.
- $\tau, k \models_s \Diamond\phi$ iff there exists $\ell \geq k$ such that $\tau, \ell \models_s \phi$.

The dual modalities are defined as usual: $\bigodot\phi \stackrel{\text{def}}{=} \neg\bigcirc\neg\phi$, is weak, and asserts that if there is a successor state, then that state satisfies ϕ. $\Box\phi \stackrel{\text{def}}{=} \neg\Diamond\neg\phi$ asserts that ϕ is an invariant for the rest of the session.

For $\tau \in \Sigma^*$, we say that $\tau \models_s \phi$ if $\tau, 1 \models_s \phi$. Let $Mod(\phi) \stackrel{\text{def}}{=} \{\tau \in \Sigma^* \mid \tau \models_s \phi\}$.

Given an information system $\mathcal{S} = (\Sigma, R)$, define $Lang(\mathcal{S}, \phi) \stackrel{\text{def}}{=} \{\rho = u_1 \cdots u_k \in \mathcal{U}^* \mid$ there exists $\tau = \sigma_0 \cdots \sigma_k \in Mod(\phi)$ such that for all $j : 1 \leq j \leq k, (\sigma_{j-1}, u_j, \sigma_j) \in R\}$.

The main theorem of the paper follows.

Theorem 4.1 *Given an information system $\mathcal{S}$, a protocol Pr over alphabet Π and a formula ϕ, with $|Pr| = k$, $|\Pi| = m$ and $|\phi| = n$, checking whether it is the case that $Pr \models \phi$ can be done in time $2^{O(k+m+n)}$.*

The proof proceeds by associating with each formula ϕ an automaton that accepts not $Lang(\phi)$, but $Lang(\phi)\lceil Y$, for some appropriate finite set $Y \subseteq \mathcal{I}$ such that $Lang(\delta) \subseteq Lang(\phi)\lceil Y$ iff $Lang(\delta) \subseteq Lang(\phi)$.

The subformula closure CL of the given formula ϕ is defined in the usual manner, and the set X of all information terms which play a role in determining the truth of ϕ is inferred from CL. X determines the set $\Sigma\lceil X$ of information states that we need to consider when evaluating ϕ. An *atom* is then defined to be a pair (A, σ) such that $\sigma \in \Sigma\lceil X$, A is a 'locally' consistent subset of CL and for all state formulas α in A, $\sigma \models_i \alpha$. These atoms are the states of the constructed automaton. Transitions between atoms are defined in a straightforward manner. Having information states as part of the atoms simplifies the construction considerably.

Note that this theorem is not at variance with results on undecidability of cryptographic protocol verification in [DLMS99] and [CDLMS99]. This is because this logic does not have enough expressive power to talk about an unbounded number of new nonces, which is basically what contributes to undecidability results in the papers mentioned above.

5 Examples of reasoning

To illustrate the kind of reasoning that goes on in the logic, we return to the example mentoned in Section 1. This situation is similar to that which obtains in the famous Needham Schroeder protocol, and the attack by Lowe [Low96] on it.

Rephrasing the protocol in the syntax presented in Section 3, we have:

$$new_A(x)$$
$$A \to B : \{x\}_B$$
$$B \to A : \{x^A\}_A$$

where $Ho = \{A, B\}$ and $Ha = \{C\}$. Note that the message B sends in line 3 is different from the one in the earlier version. This is because B being an honest principal, the messages it sends are legal, and hence originate from its database. After the first message, B could only possibly have $(\{x\}_B)^A$ in its database, which after decrypting, yields x^A.

What is the specification of this protocol ? The intention is that on termination, A and B 'mutually' have the information that x holds. This may be specified as $\alpha_1 \stackrel{\text{def}}{=} A : B : x \wedge B : A : x$. But, as we will see, the protocol above does not satisfy α_1. It does achieve the much weaker formula: $\alpha_2 \stackrel{\text{def}}{=} B : \ from_A x$.

If this requirement is seen as 'liveness', we also would like an additional 'safety' requirement, that no honest principal unintentionally leaks x to an intruder. We thus have the specification $\phi = \Box(\bigwedge_{H \in Ho, D \in Ha} \neg leak(H, D, x)) \wedge \Diamond(B : \ from_A x))$.

The notion of $H \in Ho$ unintentionally leaking x to $D \in Ha$ may be specified by:

$$leak(H, D, x) \stackrel{\text{def}}{=} (\neg D : x \wedge \bigcirc D : \ from_H x) \wedge (H : \ from_D(x \vee \bigvee_A x^A))$$

The first conjunct specifies that at some point of time during the session, H sends x to D when D does not have x. The second says that H is replying to an earlier message from D with content x or with information about having heard x from A.

Now consider any completed session of the protocol, say $\delta = \chi_1 \ldots \chi_m$. Then it contains communications say χ_i, χ_j when some variant of line 2 and line 3 are completed, respectively. These variants are of the form $A' \to B' :$

$\{x\}_B$ and $B'' \to A'' : \{x^A\}_A$, where A' is A or $(C)A$, B' is B or $B(C)$, A'' is A or $A(C)$ and B'' is B or $(C)B$.

After $new_A(x)$, we have $A : (x \wedge \neg B : x \wedge \neg C : x)$. Moreover, $\Box((\neg B : x \wedge \neg C : x) \supset \odot((B : x \supset B : \ from_A x) \wedge (C : x \supset C : \ from_A x))$. Therefore, we can argue that A' must be A.

After communication i, the formula $(C : \ from_A \ for_B x) \vee (B : \ from_A x)$ holds, and hence $(C : \ for_B x) \vee (B : \ from_A x)$ holds.

Now, B'' is either B or $(C)B$. If it is the former, then B being an honest principal, $B : \ from_A x$ holds, and the goal is satisfied. Otherwise, B'' is $(C)B$. But at i, we also have: $\Box(C : \ from_A x \supset C : \ from_B \ from_A x)$. But C can use the fact that B sends x only if B has x in its database. Therefore, at i, $\Box(C : \ from_B \ from_A x \supset B : \ from_A x)$. But at j, either $B : \ from_A x$ or $C : \ from_A x$ holds, and thus in either case $B : \ from_A x$ holds, satisfying the goal.

On the other hand, it is easy to see that the protocol does not satisfy the safety condition. The attack mentioned in Section 1 can be formalized to show this.

We can change the protocol in the following manner and show that the invariant holds.

$$new_A(x)$$
$$A \to B : \{A : x\}_B$$
$$B \to A : \{(A : x)^A\}_A$$

The other goal, namely $A : B : x$ or even a weaker form $A : B : \ from_A x$, cannot be satisfied, since the last message meant for A can always be blocked by C. To achieve such a goal, we need another acknowledgement.

$$new_A(x)$$
$$new_B(x)$$
$$A \to B : \{A : x\}_B$$
$$B \to A : \{(A : x)^A, B : y\}_A$$
$$A \to B : \{(B : y)^B\}_B$$

Since the last message is not possible unless A has received the previous one, in any completed session, A has the information that B has seen x.

These examples show the difficulty of obtaining even two 'levels' of information, of the kind: $A : B : \alpha$. However, weaker 'levels' like $A : \ from_B \alpha$ etc are easier to achieve. This is because the protocols we study are typically those where the content of messages are typically nonces and names. When messages contain complex information elements, higher levels become feasible; for instance, $A : (C : p^D)^B$ will hold after B sends a message about what information it has about C.

This remark raises another issue of interest: given a protocol, under what assumptions is a specific 'level' of information achieved ? Such a logical anal-

ysis may help in deciding patterns of communication to be followed in the design of security protocols.

When messages are sets of several terms encrypted together, this adds to an agent's inference capability. We saw in Section 2 how this complicates the update rules. For instance suppose that p is local to A in σ and A sends a message $\{\{x,p\}_C,\{y,p\}_D\}_B$. When A later sees p^E, for some E, A gets the confirmation that B has seen $\{x,p\}_C$ and $\{y,p\}_D$, but is unsure whether C has seen x or D has seen y. In such a state, it can be checked that the formula $A:(\ from_E\ p \supset (C:\ from_A\ x \vee D:\ from_A\ y))$ holds.

6 Discussion

We have presented a model of agent's information states and a logic in which updates on them can be reasoned about. The operators $from_A$, for_A and $A:$ refer to definite information that an agent has; this seems to us an essential requirement in the context of information security. While we have atempted to illustrate this in the context of authentication of messages, the reasoning primitives are similar in access control models as well. In future work, we hope to present a unified framework for both types of security properties.

We have not presented any axiomatization of validities. Of particular interest are inference rules that let us derive protocol validities: from $Pr \models \phi$ infer $Pr \models \psi$. Obtaining a complete set of such rules seems to be quite non-trivial. For instance, see that $Pr \models \alpha$ may hold without $Pr \models A:\alpha$ being true for some A.

The modality $A:\alpha$ is closely related to, but not the same as, knowledge modalities of the kind discussed in [FHMV95]. These epistemic modalities refer to implicit knowledge attributed to the agent by the protocol designer, whereas $A:\alpha$ represents knowledge of a more explicit kind [Ram99]. For instance, as mentioned above, when $Pr \models \alpha$ holds, so would $Pr \models K_A\alpha$, whereas it may be the case that $Pr \not\models A:\alpha$. Formally relating these information based modalities to epistemic ones like knowledge or belief seems an interesting issue.

We would also like to note that the reasoning in the logic is *global*. An assertion like $B:\ from_A\alpha \supset A:\alpha$ may well hold at a state, without A being 'aware' of it, in the sense that $A:B:\ from_A\alpha \wedge \neg A:B:A:\alpha$ may hold. An interesting question is how much of this reasoning can be carried out locally, and this is of importance in compositional design of security protocols.

On the modelling side, it is important to generalize the framework from mono-sessions and interleaved sessions with hacked variants to multi-sessions, where many agents are running many instances of the protocol simultaneously. The ideas of [DMTY97], [Gou00] seem very relevant here. Moreover, in the context of repeated sessions, an intruder can use past information as well, and an extension of the logic using past modalities may be indicated for this. Another important limitation of this approach is that principals are individual,

whereas in the context of information security, it is important to model group principals. A related question is to logically characterize situations where it suffices to study single hacker attacks and where collusions between hackers is crucial.

References

[AG97] Abadi, M. and Gordon, A.D., "A calculus for cryptographic protocols: The *spi* calculus", *4th ACM Conf. on Comp and Comm Security*, ACM Press, 1997.

[Bie90] Bieber, P., "A logic of communication in a hostile environment", *3rd Computer security foundations workshop*, IEEE Press, 1990, 14-22.

[Bol97] Bolignano, D., "Towards a mechanization of cryptographic protocol verification", *CAV'97*, LNCS 1254, 1997, 131-142.

[BM93] Boyd, C., and Mao, W., "On a limitation of BAN logic", *Eurocrypt'93*, LNCS, 1993, 240-247.

[BAN89] Burrows, M., Abadi, M. and Needham, R., "A logic of authentication", *Proceedings of the Royal Society*, 426 (1871), 1989, 233-271.

[CDLMS99] Cervesato, I., Durgin, N.A., Lincoln, P.D., Mitchell, J.C. and Scedrov, A., "A Meta-notation for Protocol Analysis", *12-th IEEE Computer Security Foundations Workshop*, Syverson, P., editor, IEEE Computer Society Press, 1999.

[DLMS99] Durgin, N.A., Lincoln, P.D., Mitchell, J.C. and Scedrov, A., "The undecidability of bounded security protocols", *Workshop on Formal Methods and Security Protocols (FMSP'99)*. Electronic proceedings available at: `http://www.cs.bell-labs.com/who/nch/fmsp99/program.html`, 1999.

[DM99] Durgin, N.A. and Mitchell, J.C., "Analysis of security protocols", *Calculational System Design, Series F: Computer and System Sciences, Vol. 173*, IOS Press, 1999.

[DMTY97] Debbabi, M., Mejri, M., Tawbi, N. and Yahmadi, I., "Formal automatic verification of authentication protocols", *ICFEM'97*, IEEE Press, 1997.

[FHMV95] Fagin, R., Halpern, J., Moses, Y. and Vardi, M., *Reasoning about knowledge*, M.I.T. Press, 1995.

[GK99] Genet, T. and Klay, F., "Rewriting for cryptographic protocol verification" Technical report, CNET-France Telecom, 1999.

[GNY90] Gong, L., Needham, R., and Yahalom, R., "Reasoning about belief in cryptographic protocols", *IEEE Computer Security symposium*, IEEE Press, 1990, 234-248.

[Gou00] Goubault-Larrecq, J., "A method for automatic cryptographic protocol verification", *IPDPS*, LNCS 1800, 2000, 977-984.

[JRV00] Jacquemard, F., Rusinowitch, M. and Vigneron, L., "Compiling and verifying security protocols", *LPAR*, LNCS 1955, 2000.

[KW96] Kindred, D. and Wing, J.M., "Fast automatic checking of security protocols", *2nd USENIX workshop on e-commerce*, 1996, 41-52.

[Low96] Lowe, G., "Breaking and fixing the Needham - Schroeder public key protocol using FD", *TACAS 96*, LNCS 1055, 1996, 147-166.

[MCJ97] Marrero, W., Clarke, E. M. and Jha, S., "Model checking for security protocols", *DIMACS Workshop on design and verificaton of security protocols*, 1997.

[Mc94] McLean, J., "Security models", in *Encyclopedia of Soft Engg*, (Ed) J. Marciniak, Wiley, 1994.

[Mea92] Meadows, C.A., "Applying formal methods to the analysis of a key management protocol", *Journal of computer security*, 1(1), 1992, 5-36.

[Mo99a] Monniaux, D., "Abstracting cryptographic protocols with tree automata", *Static analysis symposium*, LNCS 1694, 1999.

[Mo99b] Monniaux, D., "Decision procedures for the analysis of cryptographic protocols by logics of belief", *12th computer security foundations workshop*, IEEE, 1999.

[Nes90] Nessett, D.M., "A critique of the Burrows, Abadi and Needham logic", *ACM Operating systems review*, 24(2), 1990, 35-38.

[Pau98] Paulson, L., "The inductive approach to verifying cryptographic protocols", *Journal of computer security*, vol 6, 1998, 85-128.

[Ram99] Ramanujam, R., "View based explicit knowledge", *Annals of Pure and Applied Logic*, vol 96, 1999, 343-368.

[SvO94] Syverson, P. F., and van Oorschot, P.C., "On unifying some cryptographic protocol logics", *13th IEEE Symp. on security and privacy*, IEEE press, 1994, 14-28.