Deciding knowledge properties of security protocols *

R. Ramanujam † S. P. Suresh ‡

Abstract

Logics for specifying properties of security protocols and reasoning about them have received increasing attention over the past few years. In this paper, we propose a propositional logic of knowledge, augmented with tense modalities, in which many important properties of security protocols can be naturally expressed. We also describe in some detail the protocol model, which helps provide a precise and general semantics for the logic. The main technical result is the decidability of the verification problem for the logic.

1 Summary

The problem

Security protocols are specifications of communication patterns which are intended to let agents share secrets over a public network. They are required to perform correctly even in the presence of malicious intruders who listen to the message exchanges that happen over the network and also manipulate the system (by blocking or forging messages, for instance). Obvious correctness requirements include secrecy: an intruder cannot read the contents of a message intended for others, and authenticity: if B receives a message that appears to be from agent A and intended for B, then A indeed sent the same message intended for B in the recent past.

Mechanisms for ensuring security typically use encrypted communication. However, even the use of the most perfect cryptographic tools does not always ensure the desired security goals. (See [4] for an illuminating account.) This situation arises primarily because of logical flaws in the design of protocols. It is widely acknowledged that security protocols are hard to analyze, bugs difficult to detect, and hence that it is desirable to look for automatic means by which attacks on protocols can be discovered.

Formally, this is a verification problem of the following kind: given a security protocol Pr and a security property α, we ask whether $\mathscr{M}(\mathsf{Pr})_{\mathscr{I}} \models \alpha$: that is, whether all *runs* of the model $\mathscr{M}(\mathsf{Pr})$ associated with the protocol Pr, under the *intruder theory* $\mathscr{I}$, satisfy α. There are two important issues here, that of how the model $\mathscr{M}(\mathsf{Pr})$ is defined, and the logic in which α is specified. Ideally, we would like the logic to be abstract, in the sense that it should not refer to model elements (like encryption, which are mechanisms for implementing security), and the model to be general enough so that the verification certificate is worthwhile. [1] In general, the verification problem is hard, since $\mathscr{M}(\mathsf{Pr})$ is an infinite state system and even simple properties like reachability tend to be undecidable.

*We thank the anonymous referees for detailed comments, which have helped to improve the presentation greatly.

†The Institute of Mathematical Sciences, Chennai, India. E-mail:jam@imsc.res.in

‡Chennai Mathematical Institute, Chennai, India. E-mail: spsuresh@cmi.ac.in

[1] This requirement of abstraction may be seen as similar to that in temporal logic in the context of verifying reactive systems.

Logic of knowledge

We suggest that the propositional logic of knowledge, augmented with tense modalities, is a good candidate for studying the verification problem. As usual, the knowledge operator is defined using an indistinguishability relation for an agent A. For any protocol Pr, any run ξ of Pr, and any $A \in Ag$, define $\xi\lceil A$ to be the subsequence of all A-events in ξ. For any two runs ξ and ξ', we say that $\xi \sim_A \xi'$ iff $\xi\lceil A = \xi'\lceil A$.

This seems standard, and yet we get a new and interesting notion, because of the nature of how runs in security protocols are constructed. Firstly, while protocols refer to abstract names of agents and secrets, they only refer to *roles*, and the system has many actual agents performing possibly a multiplicity of roles concurrently in several parallel sessions.

For example, a protocol may consist of the following three communications (this is the famous Needham Schroeder protocol [22]):

$$\begin{array}{lllll} \text{Msg 1.} & A & \rightarrow & B & : \{x, A\}_{pubk_B} \\ \text{Msg 2.} & B & \rightarrow & A & : \{x, y\}_{pubk_A} \\ \text{Msg 3.} & A & \rightarrow & B & : \{y\}_{pubk_B} \end{array}$$

Here, A and B are abstract role names, and x, y abstract names for secrets. The notation $\{x\}_k$ stands for x encrypted with key k. $pubk_A$ and $pubk_B$ are the public keys of A and B respectively. Runs are obtained by interleaving several instantiations of roles. Every time secrets like x are instantiated, it has to be to a fresh value (called nonce). Agents may generate unboundedly many fresh nonces and keys in the course of a run. The intruder might force the principals to generate unboundedly long messages. There is no bound on the number of parallel sessions which can be part of a run. All this ensures that the behaviour of the protocol corresponds to an infinite state system.

There is more: runs are not merely sequences of protocol actions, but every send action of a role should be *enabled* by the previous receive actions. All this is happening in the presence of an intruder who may be listening to messages (though he cannot break encryption), forging messages or blocking them. When A sends a message to B, the latter is only the intended recipient, and when A receives a message from B, the latter is merely the purported sender.

Given all this, the notion of knowledge captured by the indistinguishability relation is indeed strong. Note that honest principals follow the protocol and hence the variability in runs is mainly due to intruder actions. *Thus, the knowledge modality refers to invariance under a variety of intruder capabilities, specific to the communication patterns in the protocol.* Hence knowledge assertions carry important security implications and verifying them offers an interesting challenge.

We propose the logic in three definitional layers. The basic logic, denoted $\mathscr{L}_0$ is standard. Assume a countable set of *basic propositions* P.

$$\mathscr{L}_0 \ ::= \ p\ (\in P) \mid \neg\alpha \mid \alpha \vee \beta \mid \mathsf{G}\alpha \mid \mathsf{H}\alpha \mid \mathsf{K}_A\alpha$$

$\mathsf{G}\alpha$ asserts that α holds *always* in the future. Its dual $\mathsf{F}\alpha$ says that α holds *sometime* in the future. Similarly $\mathsf{H}\alpha$ and $\mathsf{P}\alpha$ refer to all the time points and some time point, respectively, in the past. $\mathsf{L}_A\alpha$ is the dual of $\mathsf{K}_A\alpha$, as usual.

While this is a simple logic, there are a number of problems in ascribing knowledge to agents in this manner. For one thing, this notion of knowledge mixes up several things: knowledge of the protocol itself (in the way communications follow a specific pattern), knowledge regarding occurrence of certain events, and knowledge built up using keys and nonces. The standard logics of knowledge are principally concerned with the first two of these forms of knowledge, whereas our principal concern relates to the third. A more serious problem relates to limiting agents' knowledge to what cryptographic primitives

allow: for example, if A receives a message m encrypted with key k and in every run of the protocol the only message encrypted with k is m, then this semantics entitles A to know m, despite A having no access to the inverse of k. This is clearly problematic.

We therefore define what we call a *specification logic*. This is an attempt to limit knowledge of agents to only that data which they can explicitly construct from their initial knowledge (of terms and keys) using communications. In this logic, if an agent knows $\{m\}_k$ and does not know the inverse of k, we cannot conclude that the agent also knows m.

Formally, the logic $\mathscr{L}_1$ is defined by the following syntax. Below the subscripts r, r' etc. are assumed to come from a countable set $\mathscr{R}$, about which we will discuss later.

$$\begin{aligned}\mathscr{L}_1 \;::=\; & A_r \;\mathsf{has}\; x_{r'} \mid \mathsf{sent}(A_r, B_{r'}, x_{r''}) \mid \mathsf{received}(A_r, B_{r'}, x_{r''}) \mid \mathsf{honest}(A_r) \\ & \mid \neg\alpha \mid \alpha \vee \beta \mid \mathsf{G}\alpha \mid \mathsf{H}\alpha \mid \mathsf{K}_{A_r}\alpha\end{aligned}$$

Note that by constraining the atomic propositions, we considerably restrict the knowledge assertions. However we iterate, ultimately we refer only to knowledge of atomic data present in an agent's database, or constructable using information present in the database.

With an ability to refer to secrets and communications, we can specify many interesting constraints, where the form of knowledge used comes considerably close to informal notions of explicit knowledge. For instance, consider the following formula which specifies a version of secrecy.

$$\mathsf{sent}(A,B,x) \supset \mathsf{K}_A\mathsf{G}\,(\mathsf{honest}(B) \supset \bigvee_{C \neq A,B} \neg C \;\mathsf{has}\; x)$$

Another basic property is *authentication*: If A receives x purportedly from B, then B actually sent x intended for A. In $\mathscr{L}_1$, this is specified by: $\mathsf{received}(A,B,x) \supset \mathsf{K}_A\mathsf{sent}(B,A,x)$. Consider the property specified by: $\mathsf{sent}(A,B,x) \wedge \mathsf{K}_A(B \;\mathsf{has}\; x \wedge \mathsf{K}_B\mathsf{sent}(A,B,x))$. It states that A knows that B knows that she communicated with him.

Calling $\mathscr{L}_1$ a specification logic pertains to the fact that the only terms available in the syntax of formulas are atomic. While the protocol may use a rich term structure to achieve security, the specifications only refer to who has access to which secret and who should not have.

When the logic also refers to terms constructed using encryption, tupling etc, we get a *protocol logic*, where we can reason about how a protocol achieves (or fails to achieve) security requirements. For example, in the specification logic, we may say:

$$(\mathsf{sent}(A,B,x) \wedge \mathsf{sent}(A,B,y)) \supset K_A\mathsf{G}(B \;\mathsf{has}\; x \equiv B \;\mathsf{has}\; y)$$

This asserts that when A intends both x and y for B, if the latter gets one it also gets the other. In the protocol logic, we may assert:

$$\mathsf{sent}(A,B,\{(x,y)\}_B) \supset K_A\mathsf{G}(B \;\mathsf{has}\; x \equiv B \;\mathsf{has}\; y)$$

This uses the mechanism of tupling and encryption to assert that anyone who can decrypt the term and get x can also get y, and since B can decrypt the term (sent encrypted with B's public key), B will get both.

The logic $\mathscr{L}_2$ is formally defined below. It uses a set of terms (like t below) which are formed by starting with *basic terms* like m_r, n_r, etc. (which are abstract names for nonces and keys), and by repeatedly applying the tupling and encryption operators.

$$\begin{aligned}\mathscr{L}_2 \;::=\; & A_r \;\mathsf{has}\; t \mid \mathsf{sent}(A_r, B_{r'}, t) \mid \mathsf{received}(A_r, B_{r'}, t) \mid \mathsf{honest}(A_r) \mid \mathsf{new}(x_r) \\ & \mid \neg\alpha \mid \alpha \vee \beta \mid \mathsf{G}\alpha \mid \mathsf{H}\alpha \mid \mathsf{K}_{A_r}\alpha\end{aligned}$$

The notion of knowledge in $\mathscr{L}_2$ is considerably richer than that of $\mathscr{L}_1$, though far simpler than that allowed in $\mathscr{L}_0$. Once again, knowledge here refers to explicit knowledge of data subject to the use of cryptographic primitives, as in $\mathscr{L}_1$, avoiding many of the problems in $\mathscr{L}_0$. But, unlike in $\mathscr{L}_1$, we can refer to term structure and hence constrain the protocol to follow particular mechanisms: for instance, the use of tupling to ensure that knowledge of one term implying knowledge of another, or the use of encryption to infer that a key has been leaked from the knowledge of a term.

The results

The verification problem described earlier is undecidable for the logic of knowledge and tense, in the presence of either unbounded nonces or unbounded message length. We are therefore forced to either work with a syntactic subclass of protocols or restrict the semantics to allow only boundedly many nonces and bounded message length, if we are looking for decidable verification. In this paper, we prove the verification problems for all three logics decidable by assuming a fixed finite set of nonces and bounded message length. It is still a nontrivial result since our model still admits infinitely many runs for a given protocol. We then discuss how we can extend the result (to an appropriate subclass of protocols) even when we relax some of the constraints on the semantics.

Philosophical issues

What is the nature of knowledge asserted in the logics discussed here? Many epistemic notions underlie the models and modalities defined here:

- Agents' states are defined to be simply sets of terms. This corresponds to a database-like explicit information based notion of knowledge; at any state, the agent knows those secrets explicitly present in its database, and on every communication, simply appends received terms.
- The message construction rules synth and analz which specify how a message may be constructed or analyzed into its components define a form of algorithmic knowledge. It is this ability and the limitations thereof which crucially limit the course of events in runs.
- The freshness assumption on atomic terms (nonces and keys) relies on *non-guessability* of such terms. This means that all knowledge assertions made must be seen really to be probabilistic knowledge. When we assert that $\mathsf{K}_A\alpha$ holds, we only say that A knows α to hold with high probability. While this is implicit in the discussion, the effect of cascading such probablistic assertions, and thereby perhaps reduced confidence in them, is glossed over. This is a limitation of the analysis.
- Many common knowledge assumptions are essential to make the analysis work: the protocol itself is common knowledge, and so is the initial distribution of keys and nonces, and thereby *agents' assured initial ignorance of private keys*.
- The analysis can be seen as being carried out in the "mind" of the all-powerful intruder, who can observe all the events that occur (modulo encryption). The fact that the terms communicated follow specific patterns as dictated by the protocol is crucially used by the intruder: thus, another form of algorithmic knowledge, that of pattern matching and unification of symbolic terms, is relevant.
- Interestingly, while the modality in the logic refers to knowledge of propositions, the analysis (and indeed the very intention of the whole exercise) pertains to knowledge of data. Ontologically, these are distinct notions; while treating them as one makes for technical convenience, a case can be made for finer analysis.

- The notion of knowledge we use is standard and subject to the problem of logical omniscience. However, it can be argued that the analysis is mainly from an observer's (or the protocol designer's) viewpoint. Further, the intruder may also be performing a similar analysis, and assumption of logical omniscience on the part of the intruder is conservative. Note that the knowledge employed by the other agents in their inference is mainly algorithmic.

We mention these points not to offer specific answers to the issues raised, but to show that the framework is rich with interesting questions.

Logical issues

The most important issue in the design of the logic is that of expressiveness. We have been guided by two criteria in this matter: that of *decidable verification* and *minimality*. While epistemic logics for security applications have been around for two decades, most of them include a number of primitive formulas as well as special modalities, resulting in logics whose semantics is hard to pin down, and typically the issue of automatic verification is unaddressed. We have concentrated on (what we may call) a bare-bones logical framework, in which many interesting security properties can be expressed. In particular, we can consider protocols like Needham Schroeder protocol and not only specify properties like secrecy for initiator in the specification logic, but carry out the reasoning in the protocol logic. Most of the primitives of BAN-like logics can be translated into formulas of one of the logics defined here.

While we address the verification problem, we have not explored deduction in the logic. In particular, are there proof principles specific to logics of knowledge that help in reasoning about security protocols? Can such logics, for instance, provide alternatives to theorem proving based approaches which use first order or higher order logics? These seem unclear as of now.

An important logical issue pertains to the use of *abstract names*. While the protocol refers to abstract role names, what we wish to specify and verify are systems of concrete agents who play out these roles. To be abstract, the specification logic should also mention only the abstract names, but the semantics must translate them into concrete names used in runs. A difficulty is that the denotation of an abstract name differs at different points in runs, so we have to find a way of resolving the different meanings. This is akin to the problem of rigid designators in first order modal logic. We use the device of adding subscripts to the abstract names to fix the context. These subscripts are assumed to come from a countable set $\mathscr{R}$.

Another aspect of the logic is that while it uses tense modalities interpreted on runs, it is not really a (linear time) temporal logic of knowledge. The difference is mainly due to the semantics; if there is any ordering of protocol events, it is due to the intruder's observations. Hence, any next instant modality is hard to interpret in the sense of local reasoning as embodied in security protocols. Similar remarks apply to until modalities. Reachability and knowledge seem to be the most essentially needed modalities.

Related work

It is difficult to speak of any logic for security protocols without relating the work to BAN logic [7], which initiated the study of belief logics for authentication. It also gave rise to a host of descendants (see [15] and [2], for instance). While BAN logic has been greatly criticised ([23], for example), its high level of abstraction and ease of use is acknowledged, and many protocol errors have been analysed using BAN logic. Verification tools based on BAN logic also exist ([19], for example). One major limitation of BAN logic is its lack of a clear semantics, and even the subsequent attempts at giving it a precise semantics suffer from the fact that the details of the run generation mechanism are not explicated (which has a crucial bearing on decidability). It is this gap between what is specified and what is modelled that has led to what may be termed as "loss of faith" in belief logics.

While there have been many logic based analyses of security protocols ([1], [3], [24], [5], [16], [17], [14] is a sample list), there have been few decidable modal logics which can be considered as specification logics. Recently, there has been important work in this direction by Durgin et al. [12] and Datta et al. [8]: these papers define a modal logic, where properties that hold at the end of a run can be specified. The logic has many basic predicates (like $Knows(A, x)$, $Source(A, x, S)$ etc) including the ones we have, and uses tense modalities. However, these refer to trace properties that hold *within* a run and not epistemic properties (like the ones we study) which refer to collections of runs.

A recent paper by Dixon et al. [9] proposes a temporal logic of knowledge to reason about security protocols. It is strictly more expressive than ours, in its inclusion of next and until modalities and more crucially quantifiers over secrets. Temporal resolution is used as the main inference rule. The status of the verification problem for such a logic is unclear.

The work closest in spirit to ours is due to Halpern and Pucella [16], which uses a logic of implicit and explicit knowledge to reason about adversaries. They go beyond Dolev-Yao adversaries and consider guessing by intruders as well. The major departure in our work is in its focus on decidability of verification. We expect that similar decision procedures may be devised for HP-like logics, but this requires further study.

The notion of explicit knowledge we use, through the has modality, is interesting in that its semantics is given by a deductive system (the synth and analz rules described in the next section) instead of a particular algorithm. Recent work by Pucella [25] explores this idea at a more fundamental level, deriving axiomatizations and complexity results under different settings.

2 Security protocol modelling

We briefly present our model for protocols in this section. A more detailed presentation can be found in [26]. Most of the elements of the model are standard in the literature on modelling security protocols. In particluar, we use the Dolev-Yao adversary model [10].

Terms and actions

We start with a (potentially infinite) set of *agents* Ag, which includes the *intruder* I and the others, who are called *honest agents*. We also start with a set of *keys* K which includes long-term public, private, and shared keys, as well as temporary session keys. For every key k, we denote by $\overline{k}$ its *inverse*. Public keys and their corresponding private keys are inverses of each other, while shared keys are their own inverses. We also assume an initial distribution of long-term keys (for example, A has his private key, everyone's public key, and a shared key with everyone else) which is common knowledge. The set of keys known to A initially is denoted K_A. We also assume a countable set of *nonces* N. $\mathcal{T}_0$, the set of *basic terms*, is defined to be $K \cup N \cup Ag$. The set of *information terms* is defined to be

$$\mathcal{T} \ ::= \ m \mid (t_1, t_2) \mid \{t\}_k$$

where m ranges over $\mathcal{T}_0$ and k ranges over K. These are the terms used in the message exchanges below. We use the standard notion of subterms of a term to define $ST(t)$ for every term t.

We model communication between agents by *actions*. An action is either a *send action* of the form $A!B{:}(M)t$ or a *receive action* of the form $A?B{:}t$. Here A and B are distinct agents, A is honest; and M denotes the set of nonces and keys occurring in t which have been freshly generated during this (send) action. The agent B is (merely) the intended receiver in $A!B{:}(M)t$ and the purported sender in $A?B{:}t$. As we will see later, every send action is an instantaneous receive by the intruder, and similarly, every receive action is an instantaneous send by the intruder.

Protocol specifications

Definition 2.1 *A* **protocol** *is a pair* $\mathsf{Pr} = (\mathsf{C}, \mathsf{R})$ *where* $\mathsf{C} \subseteq \mathscr{T}_0$ *is the set of* constants *of* Pr *(intended to have a fixed interpretation in all runs of* Pr*, unlike fresh nonces and keys); and* R*, the set of* roles *of* Pr*, is a finite nonempty subset of* Ac^+ *each of whose elements is a sequence of A-actions for some honest agent A.*

The semantics of a protocol is given by the set of all its runs. A run is got by instantiating each role of the protocol in an appropriate manner, and forming admissible interleavings of such instantiations. We present the relevant definitions below.

Substitutions and events

A *substitution* σ is a partial map from $\mathscr{T}_0$ to $\mathscr{T}$ such that for all $A \in Ag$, if $\sigma(A)$ is defined then it belongs to Ag, for all $n \in N$, if $\sigma(n)$ is defined then it belongs to N, and for all $k \in K$, if $\sigma(k)$ is defined then it belongs to K. For any $T \subseteq \mathscr{T}$, σ is said to be a T-substitution iff for all $x \in \mathscr{T}_0$, if $\sigma(x)$ is defined then $\sigma(x) \in T$. Substitutions are extended to terms, sets of terms, actions and sequences of actions in a straightforward manner. The set of all substitutions is denoted by $\mathscr{S}$.

Usually, a run of protocol is taken to be any sequence of actions that can possibly be performed by the various agents taking part in the protocol. But for convenience, we model each run as a sequence of *events*, instead. An event of a protocol Pr is a triple (η, σ, lp) such that η is a role of Pr, σ is a substitution, and $1 \leq lp \leq |\eta|$. The set of all events of Pr is denoted *Events*(Pr). A T-event is one which involves a T-substitution. For an event $e = (\eta, \sigma, lp)$ with $\eta = a_1 \cdots a_\ell$, $act(e) \stackrel{\text{def}}{=} \sigma(a_{lp})$. If $lp < |\eta|$ then $(\eta, \sigma, lp) \rightarrow_\ell (\eta, \sigma, lp+1)$. For any event e, $LP(e)$, the *local past* of e, is defined to be the set of all events e' such that $e' \stackrel{+}{\rightarrow}_\ell e$.

Message generation rules

We intend a run of a protocol to be an admissible sequence of events. A very important ingredient of the admissibility criterion is the enabling of events given a particular information state. To treat this formally, we need to define how the agents (particularly the intruder) can build new messages from old. This is formalised by the notion synth and analz derivations.

Definition 2.2 *A* sequent *is of the form* $T \vdash t$ *where* $T \subseteq \mathscr{T}$ *and* $t \in \mathscr{T}$.

An analz*-proof (*synth*-proof)* π *of* $T \vdash t$ *is an inverted tree whose nodes are labelled by sequents and connected by one of the* analz*-rules (*synth*-rules) in Figure 1, whose root is labelled* $T \vdash t$*, and whose leaves are labelled by instances of the* Ax_a *rule (*Ax_s *rule). For a set of terms* T, analz(T) (synth(T)) *is the set of terms* t *such that there is an* analz*-proof (*synth*-proof) of* $T \vdash t$*. For ease of notation,* synth(analz(T)) *is denoted by* $\overline{T}$.

Information states, updates, and runs

An *information state* s is a tuple $(s_A)_{A \in Ag}$ where $s_A \subseteq \mathscr{T}$ for each agent A. $\mathscr{S}$ denotes the set of all information states. Given a protocol $\mathsf{Pr} = (\mathsf{C}, \mathsf{R})$, init(Pr), the *initial state of* Pr is defined to be $(\mathsf{C} \cup K_A)_{A \in Ag}$.

The notion of information state that we use is very rudimentary. In general, a *control state* would include more detail like the number of current sessions each agent is involved in, how far it has progressed in each of them, and so on. But some of this information is already part of the notion of an event, and therefore the above notion of an information state suffices.

Definition 2.3 *The notions of an action enabled at a state and update of a state on an action are defined as follows:*

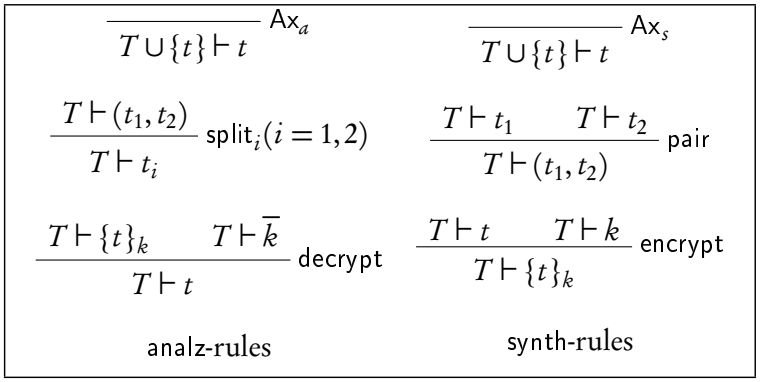

Figure 1: analz and synth rules.

- $A!B{:}(M)t$ *is* enabled *at* s *iff* $t \in \overline{s_A \cup M}$.
- $A?B{:}\,t$ *is* enabled *at* s *iff* $t \in \overline{s_I}$.
- $update(s, A!B{:}(M)t) \stackrel{\text{def}}{=} s'$ *where* $s'_A = s_A \cup M$, $s'_I = s_I \cup \{t\}$, *and for all agents* C *distinct from* A *and* I, $s'_C = s_C$.
- $update(s, A?B{:}\,t) \stackrel{\text{def}}{=} s'$ *where* $s'_A = s_A \cup \{t\}$ *and for all agents* C *distinct from* A, $s'_C = s_C$.

Definition 2.4 *Given a protocol* Pr *and a sequence* $\xi = e_1 \cdots e_k$ *of events of* Pr, $infstate(\xi)$ *is defined to be* $update(\mathsf{init}(\mathsf{Pr}), act(e_1) \cdots act(e_k))$. *An event* e *is said to be* enabled at ξ *iff* $LP(e) \subseteq \{e_1, \ldots, e_k\}$ *and* $act(e)$ *is enabled at* $infstate(\xi)$.

Definition 2.5 *Given a protocol* Pr, *a sequence* $\xi = e_1 \cdots e_k$ *of events of* Pr *is said to be a* run *of* Pr *iff:*

- *for all* $i : 1 \leq i \leq k$, e_i *is enabled at* $e_1 \cdots e_{i-1}$,
- *for all* $i : 1 \leq i \leq k$, $NT(e_i) \cap ST(\mathsf{init}(\mathsf{Pr})) = \emptyset$, *and for all* $i < j \leq k$, $NT(e_i) \cap NT(e_j) = \emptyset$. *(This is the* unique origination property *of runs.)*

For any $T \subseteq \mathscr{T}$, *a run* ξ *is said to be a* T*-run if it is composed only of* T*-events. We denote the set of runs (*T*-runs) of* Pr *by* $\mathscr{R}(\mathsf{Pr})$ *(resp.* $\mathscr{R}_T(\mathsf{Pr})$*).*

An example protocol

To illustrate some of the features of our model, we formally present the Needham-Schroeder protocol, and look at some of its runs.

There are two roles in this protocol. The *initiator role* η_1 is given below:

$$\begin{array}{llllll} 1. & A & ! & B & : (x) & \{A, x\}_{pubk_B} \\ 2. & A & ? & B & : & \{x, y\}_{pubk_A} \\ 3. & A & ! & B & : & \{y\}_{pubk_B} \end{array}$$

The *responder role* η_2 is given below:

$$
\begin{array}{llllll}
1. & B & ? & A & : & \{A,x\}_{pubk_B} \\
2. & B & ! & A & : (y) & \{x,y\}_{pubk_A} \\
3. & B & ? & A & : & \{y\}_{pubk_B}
\end{array}
$$

One of its runs is ξ_1, presented below. In the following, σ is a substitution such that $\sigma(A) = A$, $\sigma(B) = B$, $\sigma(x) = m$ and $\sigma(y) = n$.

$$
\begin{array}{llllll}
(\eta_1,\sigma,1) & A & ! & B & : (m) & \{A,m\}_{pubk_B} \\
(\eta_2,\sigma,1) & B & ? & A & : & \{A,m\}_{pubk_B} \\
(\eta_2,\sigma,2) & B & ! & A & : (n) & \{m,n\}_{pubk_A} \\
(\eta_1,\sigma,2) & A & ? & B & : & \{m,n\}_{pubk_A} \\
(\eta_1,\sigma,3) & A & ! & B & : & \{n\}_{pubk_B} \\
(\eta_2,\sigma,3) & B & ? & A & : & \{n\}_{pubk_B}
\end{array}
$$

It is easy to see that every sent message can be built by the corresponding agent using the date received in earlier communications.

Another run, this time involving the intruder, is ξ_2. In the following, σ_1 is a substitution such that $\sigma_1(A) = A$, $\sigma_1(B) = I$, $\sigma_1(x) = m$, and $\sigma_1(y) = n$; and σ_2 is a substitution such that $\sigma_2(A) = A$, $\sigma_2(B) = B$, $\sigma_2(x) = m$, and $\sigma_2(y) = n$.

$$
\begin{array}{llllll}
(\eta_1,\sigma_1,1) & A & ! & I & : (m) & \{A,m\}_{pubk_I} \\
(\eta_2,\sigma_2,1) & B & ? & A & : & \{A,m\}_{pubk_B} \\
(\eta_2,\sigma_2,2) & B & ! & A & : (n) & \{m,n\}_{pubk_A} \\
(\eta_1,\sigma_1,2) & A & ? & I & : & \{m,n\}_{pubk_A} \\
(\eta_1,\sigma_1,3) & A & ! & I & : & \{n\}_{pubk_I} \\
(\eta_2,\sigma_2,3) & B & ? & A & : & \{n\}_{pubk_B}
\end{array}
$$

Note that as far as B is concerned, he is just having a normal session with A. But A is actually talking to I parallelly (as part of another legitimate session), and uses information from one run in the other cleverly. At the end of the fifth event above, the intruder gets to know n, which was intended to be secret between A and B. Thus B has no guarantee of its nonce remaining secret in the run ξ_2. In fact, even when B is taking part in the legitimate run ξ_1, he cannot rule out the fact that the run taking place is ξ_2, and cannot be assured of the secrecy of n. This is the famous Lowe's attack [20] on the Needham-Schroeder protocol. We'll later formalize some interesting properties of this protocol and discuss which ones are satisfied and which aren't, at the end of the next section.

3 The logics

We now give the formal semantics of the three logics. We need to define the indistinguishability relation first. For any protocol Pr, any run ξ of Pr, and any $A \in Ag$, define $\xi\restriction A$ to be the subsequence of all A-events in ξ. For any two runs ξ and ξ', any two i, i' such that $i \leq |\xi|$ and $i' \leq |\xi'|$, and any $A \in Ag$, we say that $(\xi, i) \sim_A (\xi', i')$ iff $(\xi(1)\cdots\xi(i))\restriction A = (\xi'(1)\cdots\xi'(i'))\restriction A$.

For the semantics of $\mathscr{L}_0$, we assume that a satisfaction relation $\models$ is given which tells us for each information state s and each basic proposition p whether $s \models p$. Given any formula α of $\mathscr{L}_0$, a protocol Pr, $\xi \in \mathscr{R}(\mathsf{Pr})$ and $i \leq |\xi|$, the satisfaction relation $\mathsf{Pr}, (\xi, i) \models \alpha$ is defined by a straightforward induction. We present a few representative cases:

- $\mathsf{Pr}, (\xi, i) \models p$ iff $\mathit{infstate}(\xi, i) \models p$, for $p \in P$;

- $\mathsf{Pr},(\xi,i) \models \mathsf{G}\alpha$ iff for all $j : i \leq j \leq |\xi|$, $\mathsf{Pr},(\xi,j) \models \alpha$;
- $\mathsf{Pr},(\xi,i) \models \mathsf{H}\alpha$ iff for all $j : 1 \leq j \leq i$, $\mathsf{Pr},(\xi,j) \models \alpha$;
- $\mathsf{Pr},(\xi,i) \models \mathsf{K}_A\alpha$ iff for all $\xi' \in \mathscr{R}(\mathsf{Pr})$ and $i' \leq |\xi'|$ such that $(\xi,i) \sim_A (\xi',i')$, $(\xi',i') \models \alpha$.

Given a set of terms T, we say $\mathsf{Pr} \models^T \alpha$ iff for all $\xi \in \mathscr{R}_T(\mathsf{Pr})$, $\mathsf{Pr},(\xi,0) \models \alpha$.

We next present the semantics of $\mathscr{L}_2$. We intend $\mathscr{L}_1$ to have the same semantics. So in this sense $\mathscr{L}_1$ is a strict sublogic of $\mathscr{L}_2$. For giving the semantics of $\mathscr{L}_2$ we need to handle rigid designators (in particular, the subscripts which act as *rigidifiers*.) We follow the relative simple strategy of interpreting each subscript occurring in the formula as a substitution. So along with the protocol we have an assignment $\mathfrak{a} : \mathscr{R} \to \mathscr{S}$, where $\mathscr{S}$ is the set of substitutions as defined in the previous section. For any assignment $\mathfrak{a}$, and any basic term x_r, we define $\mathfrak{a}(x_r)$ to be $\sigma(x)$, where $\mathfrak{a}(r) = \sigma$. $\mathfrak{a}(t)$ for an arbitrary term t is defined in the obvious inductive manner.

Given any formula α of $\mathscr{L}_2$, a protocol Pr, an assignment $\mathfrak{a}$, a run $\xi \in \mathscr{R}(\mathsf{Pr})$ and $i \leq |\xi|$, the satisfaction relation $(\mathsf{Pr},\mathfrak{a}),(\xi,i) \models \alpha$ is defined along the same lines as the earlier definition. We present some of the cases which are special to $\mathscr{L}_2$:

- $(\mathsf{Pr},\mathfrak{a}),(\xi,i) \models A_r \text{ has } x_{r'}$ iff $\mathfrak{a}(x_{r'}) \in \overline{(\mathit{infstate}(\xi,i))_{\mathfrak{a}(A_r)}}$;
- $(\mathsf{Pr},\mathfrak{a}),(\xi,i) \models \mathsf{sent}(A_r,B_{r'},t)$ iff $\mathit{act}(\xi(i)) = \mathfrak{a}(A_r)!\mathfrak{a}(B_{r'}):(M)t'$ for some M and t' such that $\mathfrak{a}(t) \in ST(t')$;
- $(\mathsf{Pr},\mathfrak{a}),(\xi,i) \models \mathsf{honest}(A_r)$ iff $\mathfrak{a}(A_r) \neq I$;
- $\mathsf{Pr},(\xi,i) \models \mathsf{new}(x_r)$ iff $\mathit{act}(\xi(i)) = A!B:(M)t'$ for some M and t' and $\mathfrak{a}(x_r) \in M$;
- $(\mathsf{Pr},\mathfrak{a}),(\xi,i) \models \mathsf{K}_{A_r}\alpha$ iff for all $\xi' \in \mathscr{R}(\mathsf{Pr})$ and $i' \leq |\xi'|$ such that $(\xi,i) \sim_{\mathfrak{a}(A_r)} (\xi',i')$, $(\xi',i') \models \alpha$.

Given a set of terms T, we say $\mathsf{Pr} \models^T \alpha$ iff for all $\xi \in \mathscr{R}_T(\mathsf{Pr})$, and for all assignments $\mathfrak{a}$, $(\mathsf{Pr},\mathfrak{a}),(\xi,0) \models \alpha$.

Properties of the Needham-Schroeder protocol

We now consider the Needham-Schroeder protocol as formalised in the previous section and consider some of its properties. One of the most immediate properties that we desire of this protocol is that of *secrecy*. There are two desirable secrecy requirements in this case. *Secrecy for the initiator* says that all fresh nonces that are instantiated for x and not intended for the intruder are not leaked to the intruder. A simple form of the above is expressed by the following formula:

$$\mathsf{G}[(\mathsf{sent}(A_r,B_r,x_r) \wedge \mathsf{honest}(B_r)) \supset \neg(I \text{ has } x_r)].$$

Secrecy for the responder says something similar about y. A stronger version is the following:

$$\mathsf{G}[\mathsf{sent}(B_r,A_r,y_r) \supset \mathsf{K}_{B_r}(\mathsf{honest}(B_r) \supset \neg(I \text{ has } x_r))].$$

Authentication for the initiator says that if the initiator A receives y apparently from B, then B actually sent y sometime in the past. A much stronger version demands that A know the fact.

$$\mathsf{G}[\mathsf{received}(A_r,B_r,y_r) \supset \mathsf{K}_{A_r}(\mathsf{P}\,\mathsf{sent}(B_r,A_r,y_r))].$$

Authentication for the responder says something similar about the responder.

$$\mathsf{G}[\mathsf{received}(B_r, A_r, x_r) \supset \mathsf{K}_{B_r}(\mathsf{P}\ \mathsf{sent}(A_r, B_r, x_r))].$$

The notable feature of the formulas is that they are quite simple and intuitive to write, not requiring us to name any actual terms that are substituted, nor even mention different substitution names. Much stronger requirements can be stated by iterating the knowledge operator many times. But they will not be satisfied by this protocol. In fact, it can be seen that the number of rounds of communication is a means of achieving higher levels of knowledge.

Of the above properties, secrecy and authentication for the responder are not guaranteed by the Needham-Schroeder protocol. It follows from the discussion in the previous section that the simple version of secrecy is violated by the bad run ξ_2, while the stronger version of secrecy is violated even by the good run ξ_1.

It can be informally argued that the initiator enjoys both secrecy and authentication. The proof would proceed by supposing that there is a run ξ in which A sent x to B initially, B is honest, but still I has x at the end. By considering the point i such that $(\xi, i) \models \neg(I\ \mathsf{has}\ x)$ and $(\xi, i+1) \models I\ \mathsf{has}\ x$, we chase back all communication paths (some C must have sent this to I, and some earlier message received by C prompted him to do this send, and so on) and arrive at a contradiction that in all these cases I must already have x to start the chain of communication. Formalizing such an argument in $\mathscr{L}_2$ (or an enhancement of it) is an interesting problem to consider.

Translating BAN logic

We believe that much of the reasoning carried out in BAN-style logics can be carried out in $\mathscr{L}_2$ (possibly with enhancements) with the added benefits of a precise and general protocol modelling. We will illustrate this by expressing a few of the BAN primitives in $\mathscr{L}_2$ and also showing the validity of some inference rules of BAN expressed in $\mathscr{L}_2$. The BAN primitive believes is coded up by the knowledge modality in our logics. The primitives sees and said can be roughly coded using the formulas received and sent. The primitive fresh is translated into the more precise new primitive. BAN also has a primitive which stands for: "k is a good shared key for P and Q", which can be roughly translated into $P\ \mathsf{has}\ k \wedge Q\ \mathsf{has}\ k \wedge \mathsf{G}\neg I\ \mathsf{has}\ k$. There is another primitives standing for: "x is secret between P and Q, and can be used to establish each others identity", which can be roughly translated into $\varphi(P,Q,x) \stackrel{\text{def}}{=} P\ \mathsf{has}\ x \wedge Q\ \mathsf{has}\ x \wedge \mathsf{K}_P(Q\ \mathsf{has}\ x \wedge \bigvee_{A \neq Q} \neg A\ \mathsf{has}\ x) \wedge \mathsf{K}_Q(P\ \mathsf{has}\ x \wedge \bigvee_{A \neq P} \neg A\ \mathsf{has}\ x)$. Now we can see that for any protocol $(\varphi(P,Q,x) \wedge \mathsf{received}(P,R,\{x,t\}_k)) \supset \mathsf{P}\ \mathsf{sent}(Q,S,\{x,t\}_k)$ is valid, for arbitrary agents R and S. This corresponds to one of the inference rules of BAN. Of course, the idea is not to define a precise encoding but to illustrate the potential of our formalism to carry out reasoning of a similar flavour.

4 Decidability

In this section, we state the main technical results concerning the verification problem for our logics, and provide an outline of some of the proofs.

The first result is that the verification problem is undecidable in general, even for very simple logics.

Theorem 4.1 *The verification problems for the logics $\mathscr{L}_0$, $\mathscr{L}_1$, and $\mathscr{L}_2$ are undecidable, when there are either unboundedly many nonces ([11] or chapter 3 of [29]), or unboundedly long messages [13].*

We therefore fix a bound on the number of nonces and on the message length and obtain a decidability result. In the next section, we discuss how this result can be extended when we relax some of the boundedness restrictions.

Theorem 4.2 *Fix a finite $T \subseteq \mathcal{T}_0$. The problem of checking for a given protocol* Pr *and a formula α of $\mathcal{L}_0$ or $\mathcal{L}_1$ or $\mathcal{L}_2$, whether all T-runs of* Pr *satisfy α (in symbols $\mathsf{Pr} \models^T \alpha$), is decidable.*

Note that the result is still nontrivial since the set of runs of a protocol is infinite. In the rest of this section, we outline a proof of Theorem 4.2 for $\mathcal{L}_0$. We will discuss how to extend the result to the other logics at the end of the section.

For the rest of the section, we will fix a formula α_0 of $\mathcal{L}_0$, a set T of basic terms, and a protocol Pr. Our aim is to check whether $\mathsf{Pr} \models^T \alpha_0$.

For any given formula α, the set $SF(\alpha)$ of subformulas of α is defined in the standard manner. Let SF denote $SF(\alpha_0)$. We define $\neg SF$ to be the set $\{\alpha \mid \neg\alpha \in SF\} \cup \{\neg\alpha \mid \alpha \in SF$ and α is not of the form $\neg\beta\}$. We define CL to be $SF \cup \neg SF$.

An *atom* Ψ is any subset of CL such that: for all $\neg\alpha \in CL$, $\neg\alpha \in \Psi$ iff $\alpha \notin \Psi$; for all $\alpha \vee \beta \in CL$, $\alpha \vee \beta \in \Psi$ iff $\alpha \in \Psi$ or $\beta \in \Psi$; for all $\mathsf{F}\alpha \in CL$, if $\alpha \in \Psi$ then $\mathsf{F}\alpha \in \Psi$; for all $\mathsf{P}\alpha \in CL$, if $\alpha \in \Psi$ then $\mathsf{P}\alpha \in \Psi$; and for all $\mathsf{K}_A\alpha \in CL$, if $\mathsf{K}_A\alpha \in \Psi$ then $\alpha \in \Psi$.

Given two atoms Ψ_1 and Ψ_2, we say that $\Psi_1 \longrightarrow \Psi_2$ iff:

- for all $\mathsf{F}\alpha \in CL$: if $\mathsf{F}\alpha \in \Psi_2$ then $\mathsf{F}\alpha \in \Psi_1$, and if $\mathsf{F}\alpha \in \Psi_1$ and $\alpha \notin \Psi_1$ then $\mathsf{F}\alpha \in \Psi_2$; and
- for all $\mathsf{P}\alpha \in CL$: if $\mathsf{P}\alpha \in \Psi_1$ then $\mathsf{P}\alpha \in \Psi_2$, and if $\mathsf{P}\alpha \in \Psi_2$ and $\alpha \notin \Psi_2$ then $\mathsf{P}\alpha \in \Psi_1$.

An atom Ψ is an *initial atom* iff for all $\mathsf{P}\alpha \in CL$, if $\mathsf{P}\alpha \in \Psi$ then $\alpha \in \Psi$; and a *final atom* iff for all $\mathsf{F}\alpha \in CL$, if $\mathsf{F}\alpha \in \Psi$ then $\alpha \in \Psi$.

Given two atoms Ψ_1 and Ψ_2 and an A, we say that $\Psi_1 \sim_A \Psi_2$ for all $\mathsf{K}_A\alpha \in CL$, $\mathsf{K}_A\alpha \in \Psi_1$ iff $\mathsf{K}_A\alpha \in \Psi_2$.

For any set E, the function $\mathsf{red} : E^* \to E^*$ is defined as follows:

- $\mathsf{red}(\varepsilon) = \varepsilon$.
- $\mathsf{red}(\xi \cdot e) = \begin{cases} \mathsf{red}(\xi) \cdot e & \text{if } e \text{ does not occur in } \mathsf{red}(\xi) \\ \mathsf{red}(\xi) & \text{otherwise} \end{cases}$

$\mathsf{red}(\xi)$ is called the *reduced form* of ξ. We call ξ a *reduced sequence* iff $\mathsf{red}(\xi) = \xi$.

A *chain* ζ is a reduced sequence of the form $(\mathsf{first}, \Psi_0)(e_1, \Psi_1)\cdots(e_k, \Psi_k)$ where:

- $e_1 \cdots e_k$ is a T-run of Pr,
- first is an arbitrary fixed symbol not in $Events(\mathsf{Pr})$,
- Ψ_0 is an initial atom and Ψ_k is a final atom,
- for all $j : 1 \le j \le k$, $\Psi_{j-1} \longrightarrow \Psi_j$, and
- for all $j : 0 \le j \le k$, and for all $p \in P$, $p \in \Psi_j$ iff $infstate(e_1 \cdots e_j) \models p$.

For the above chain ζ, we say that its run is $e_1 \cdots e_k$, and its length is k.

A *molecule* is a pair $\chi = (\zeta, i)$ such that $\zeta = (\mathsf{first}, \Psi_0)(e_1, \Psi_1)\cdots(e_k, \Psi_k)$ is a chain and $0 \le i \le k$. For the above molecule χ, we say that its *run* is $e_1 \cdots e_k$, its *chain* is ζ, its *atom* is Ψ_i and its *event* is e_i. The set of all molecules is denoted by $\mathcal{M}$.

We say that $\chi_1 \xrightarrow{e} \chi_2$ iff for some chain ζ and some i smaller than the length of ζ, $\chi_1 = (\zeta, i)$, $\chi_2 = (\zeta, i+1)$, and e is the $i+1$-th event in the run of ζ.

For $\chi = ((\mathsf{first}, \Psi_0)(e_1, \Psi_1)\cdots(e_k, \Psi_k), i)$ and $\chi' = (\mathsf{first}, \Psi'_0)(e'_1, \Psi'_1)\cdots(e'_\ell, \Psi'_\ell), i')$, we say that $\chi \sim_A \chi'$ iff:

- $e_1 \cdots e_i \sim_A e'_1 \cdots e'_{i'}$, and
- for all j such that e_j is an A-event, $\Psi_j \cap P_A = \Psi'_j \cap P_A$. Here P_A denotes the formulas in P which are affected by A (for example: $\mathsf{sent}(A, B, x)$, $A?B{:}\,t$, A has x and so on).

A molecule is an *initial molecule* iff it is of the form $(\zeta, 0)$. The set of initial molecules is denoted by $\mathscr{I}$. A molecule χ is said to be a *final molecule* iff it is of the form (ζ, k) where k is the length of ζ. The set of final molecules is denoted by $\mathscr{F}$.

The tuple $\mathscr{G} = (\mathscr{M}, \mathscr{I}, \mathscr{F}, \longrightarrow, (\sim_A)_{A \in Ag})$ is called the *molecule graph*. We now define the important notion of a *good subgraph* as follows:

Definition 4.3 (Good subgraph of $\mathscr{G}$) *A subgraph $\mathscr{G}' = (\mathscr{M}', \mathscr{I}', \mathscr{F}', \longrightarrow, (\sim_A)_{A \in Ag})$ of $\mathscr{G}$ is called a* good subgraph *iff:*

- *$\longrightarrow$ and $\sim_A$ on $\mathscr{G}'$ are just the restrictions of the relations on $\mathscr{G}$ to $\mathscr{M}'$, for every $A \in Ag$,*
- *every $\chi \in \mathscr{M}'$ is reachable from $\mathscr{I}'$,*
- *every path from every $\chi \in \mathscr{M}'$ ends in $\mathscr{F}'$,*
- *for every $\chi \in \mathscr{M}'$ with atom Ψ, if $\mathsf{L}_A\alpha \in \Psi$ then there exists $\chi' \in \mathscr{M}'$ with atom Ψ' such that $\chi \sim_A \chi'$ and $\alpha \in \Psi'$, and*
- *for all $\chi = (\zeta, i) \in \mathscr{M}'$ with run ξ and all $\chi' = (\zeta', i') \in \mathscr{M}$ with run ξ' such that $(\xi, i) \sim_A (\xi', i')$, there exists $\chi'' = (\zeta'', i'') \in \mathscr{M}'$ with run ξ' such that $\chi \sim_A \chi''$.*

Many of the definitions are on standard lines. The definition of a good subgraph deserves some discussion. The tricky parts of the definition concerns interpreting formulas of the form $\mathsf{K}_A\alpha$ properly. Note that we need to consider all indistinguishable worlds in evaluating a formula of this form. If we do not represent some of them in the subgraph which we construct, then there might be molecules in the subgraph which contain $\mathsf{K}_A\alpha$ only because not enough indistinguishable worlds have been considered, whereas in the model there is no corresponding world where $\mathsf{K}_A\alpha$ is true. That would be a serious mismatch between the actual model and our subgraph. We impose the appropriate conditions on subgraphs to precisely avoid this mismatch from happening. We force, in effect, for every molecule χ in the subgraph (which represents a possible world (ξ, i) of the model), a representative of every other world (ξ', i') A-equivalent to (ξ, i) to be included in $\mathscr{M}'$. This ensures the truth of the following lemmas.

For every run $\xi = e_1 \cdots e_k$ of Pr, $\zeta(\xi)$ is defined to be $\mathsf{red}((\mathsf{first}, \Psi_0)(e_1, \Psi_1)\cdots(e_k, \Psi_k))$, where for all $j \leq k$, $\Psi_j = \{\alpha \in CL \mid (\xi, j) \models \alpha\}$. Let ℓ be the length of ζ. For every $i \leq \ell$, $\chi_i(\xi)$ is defined to be $(\zeta(\xi), i)$.

Define $\mathscr{G}^*$ to be $(\mathscr{M}^*, \mathscr{I}^*, \mathscr{F}^*, \longrightarrow, (\sim_A)_{A \in Ag})$, where $\mathscr{M}^*$ is the set of all $\chi_i(\xi)$ for all runs ξ of Pr, as defined above; $\mathscr{I}^* = \mathscr{I} \cap \mathscr{M}^*$; $\mathscr{F}^* = \mathscr{F} \cap \mathscr{M}^*$; and $\longrightarrow$ and $\sim_A$ on $\mathscr{G}^*$ are just the restrictions of the relations on $\mathscr{G}$ to $\mathscr{M}^*$, for every $A \in Ag$.

Lemma 4.4 *$\mathscr{G}^*$ is a good subgraph.*

Lemma 4.5 *Given a good subgraph $\mathscr{G}'$ with a molecule χ, for every path from an initial node to χ labelled $e_1 \cdots e_i$ and for every path from χ to a final node labelled $e_{i+1} \cdots e_k$, and for every $\alpha \in CL$, $(e_1 \cdots e_k, i) \models \alpha$ iff $\alpha \in \Psi$, where Ψ is the current atom of χ.*

The above two lemmas immediately yield the following lemma.

Lemma 4.6 *For a protocol* Pr *and a run ξ of* Pr, $\mathsf{Pr}, (\xi, 0) \models \alpha$ *iff there is a good subgraph $\mathscr{G}'$ containing an initial node χ with atom Ψ_0 such that $\alpha \in \Psi_0$ and there is a path labelled ξ from χ to a final node.*

We solve the verification problem as follows. Given Pr and α, we guess a good subgraph $\mathscr{G}'$ of the molecule graph of Pr and α, and check whether $\mathscr{G}'$ satisfies $\neg\alpha$. Thus we have a nondeterministic procedure. Below we make a rough estimate of the size of $\mathscr{G}$ given Pr, T and α.

Let us say the number of basic terms mentioned in Pr is n_1, the size of T is n_2 and the size of α is n_3. We can see that the set of T-substitutions (and therefore the set of T-events relevant for our analysis) is of size roughly $N_1 = n_2^{n_1}$. The set of atoms is of size $N_2 = 2^{n_3}$. Thus we can see that the set of molecules is of size roughly $(N_1 \cdot N_2)^{N_1 \cdot N_2}$. This is of the order of $2^{2^{(n^2)}}$ where n is the maximum of n_1, n_2 and n_3. Even though the proposed procedure is nondeterministic, we believe it may be possible to build an appropriate good subgraph deterministically, starting from $\mathscr{G}$. That would yield us a deterministic procedure with the time complexity indicated above.

It is easy to extend this result to the logics $\mathscr{L}_1$ and $\mathscr{L}_2$. The main technical difficulty is to do with the fact that we use abstract names in those two logics, and fix the context using rigidifiers. So in effect they behave like quantifiers. But we can prove a kind of quantifier elimination, which will imply the following: for any formula α of $\mathscr{L}_2$ (and given a protocol Pr), we can find a formula α' which uses only concrete terms (and not abstract terms with rigidifiers) and whose size can be bounded in terms of the size of α, such that $\mathsf{Pr} \models \alpha$ iff $\mathsf{Pr} \models \alpha'$. Note that this result would hold even when we consider an unbounded number of nonces and keys in the semantics. But formulas which use concrete terms are just special cases of formulas from $\mathscr{L}_0$, and the above decidability proof can be applied with few changes. This proves Theorem 4.2.

5 Discussion

We have presented logics based on knowledge and tense modalities, to specify interesting properties of security protocols and reason about them. We have also presented a detailed model for security protocols, and explored the verification problem for the logics.

In particular, we have proved the verification problem decidable when we restrict the semantics to allow only boundedly many nonces. There is another restriction in the model, which is that we substitute atomic terms in the protocol specification with atomic terms of the same type only. A lot of interest has been shown by the security protocol verification community to handling the problem even when these two restrictions are lifted. Many approaches involve allowing only a bounded number of sessions in each run ([28] is an example) — again a semantic restriction. But there have also been attempts to consider syntactic subclasses of protocols, whose structural properties ensure decidability of verification, even under an infinite-state semantics ([18], [21], [6], [27]). The state of the art is that the problem of ill-typed substitution can be handled, in the sense that there are syntactic subclasses for which the verification problem for $\mathscr{L}_2$ is decidable, if we assume boundedly many nonces in addition. But when we allow unboundedly many nonces, even though the verification of some important properties like secrecy is decidable, extending the result to logics like $\mathscr{L}_2$ remains open.

Below, we outline briefly how the decidability can be extended to the case of ill-typed substitutions. We work with the syntactic subclass of *tagged protocols*, in which every distinct encrypted term occurring

in the protocol specification is tagged with a distinct tag. During run generation, these tags are treated as constants and cannot be substituted with other terms. It is assumed that the different agents participating in the protocol can check for the occurrence of the tags at the appropriate places. Tagging prevents the intruder from learning a term t which is an instance of an encrypted term s occurring in the protocol specification, and (in a different session) passing the same t off as an instance of another term $r \neq s$ occurring in the protocol specification. (This is a typical method of learning new secrets by introducing type flaws.) In effect, tagging ensures that the only terms used in different contexts by the intruder are atomic terms learnt by him. Using this fact (and some involved combinatorial reasoning), we show that whenever the intruder introduces a type flaw, he could have learnt the same information without introducing the type flaw. More details can be found in Chapter 5 of [29]. In fact one can show that there is a map μ which maps any run ξ to an equivalent well-typed run, in the sense that $(\xi, i) \models \alpha$ iff $(\mu(\xi), i) \models \alpha$ for any *atomic formula* α. Further it can also be proved that $\xi \sim_A \xi'$ iff $\mu(\xi) \sim_A \mu(\xi')$. From these observations, we obtain the decidability result for $\mathscr{L}_2$ for the subclass of tagged protocols.

References

[1] Martin Abadi, Michael Burrows, Charles Kaufman, and Butler Lampson. Authentication and Delegation with Smart-Cards. *Science of Computer Programming*, 21(2):93–113, October 1993.

[2] Martin Abadi and Mark Tuttle. A Semantics fo a Logic of Authentication. In *Proceedings of the 10th ACM Annual Symposium on Principles of Distributed Computing*, pages 201–216, Aug 1991.

[3] Kamel Adi, Mourad Debbabi, and Mohamed Mejri. A new logic for electronic commerce protocols. *Theoretical Computer Science*, 29(3):223–283, 2003.

[4] Ross Anderson and Roger M. Needham. Programming Satan's computer. In *Computer Science Today*, volume 1000 of *Lecture Notes in Computer Science*, pages 426–441, 1995.

[5] Giampaolo Bella. Inductive Verification of Smartcard Protocols. *Journal of Computer Security*, 11(1):87–132, 2003.

[6] Bruno Blanchet and Andreas Podelski. Verification of Cryptographic Protocols: Tagging Enforces Termination. In Andrew D. Gordon, editor, *Proceedings of FoSSaCS'03*, volume 2620 of *Lecture Notes in Computer Science*, pages 136–152, 2003.

[7] Michael Burrows, Martin Abadi, and Roger M. Needham. A logic of authentication. *ACM Transactions on Computer Systems*, 8(1):18–36, Feb 1990.

[8] Anupam Datta, A. Derek, John C. Mitchell, and Dusko Pavlovic. Secure protocol composition. In *Proceedings of 19th Conference on Mathematical Foundations of Programming Semantics*, volume 83 of *Electronic Notes in Theoretical Computer Science*, 2003.

[9] Clare Dixon, M. Carmel Fernandez Gago, Michael Fisher, and Wiebe van der Hoek. Using Temporal Logics of Knowledge in the Formal Verification of Security Protocols. In *Proceedings of TIME 2004*, July 2004.

[10] Danny Dolev and Andrew Yao. On the Security of public-key protocols. *IEEE Transactions on Information Theory*, 29:198–208, 1983.

[11] Nancy A. Durgin, Patrick D. Lincoln, John C. Mitchell, and Andre Scedrov. The undecidability of bounded security protocols. In *Proceedings of the Workshop on Formal Methods and Security Protocols (FMSP'99)*, 1999.

[12] Nancy A. Durgin, John C. Mitchell, and Dusko Pavlovic. A compositional logic for proving security properties of protocols. *Journal of Computer Security*, 11(4):677–721, 2003.

[13] Shimon Even and Oded Goldreich. On the security of multi-party ping-pong protocols. Technical Report 285, Technion - Israel Institute of Technology, 1983.

[14] Janice Glasgow, Glenn MacEwen, and Prakash Panangaden. A logic for reasoning about security. *ACM Transactions on Computer Systems*, 10(3):226–264, Aug 1992.

[15] Li Gong, Roger Needham, and Raphael Yahalom. Reasoning About Belief in Cryptographic Protocols. In Deborah Cooper and Teresa Lunt, editors, *Proceedings 1990 IEEE Symposium on Research in Security and Privacy*, pages 234–248. IEEE Computer Society, 1990.

[16] Joseph Y. Halpern and Riccardo Pucella. Modeling adversaries in a logic for security protocol analysis. In *Formal Aspects of Security, First International Conference, FASec 2002*, volume 2629 of *Lecture Notes in Computer Science*, pages 115–132, 2003.

[17] Joseph Y. Halpern and Ron van der Meyden. A logical reconstruction of SPKI. *Journal of Computer Security*, 11(4):581–613, 2003.

[18] James Heather, Gavin Lowe, and Steve Schneider. How to Prevent Type Flaw Attacks on Security Protocols. In *Proceedings of the 13th IEEE Computer Security Foundations Workshop (CSFW 13)*, pages 255–268, July 2000.

[19] Darrell Kindred and Jeannette Wing. Fast, automatic checking of security protocols. In *Proceedings of the 2nd USENIX workshop on e-commerce*, pages 41–52, 1996.

[20] Gavin Lowe. Breaking and fixing the Needham-Schroeder public key protocol using FDR. In *Proceedings of TACAS'96*, volume 1055 of *Lecture Notes in Computer Science*, pages 147–166, 1996.

[21] Gavin Lowe. Towards a completeness result for model checking of security protocols. *Journal of computer security*, 7:89–146, 1999.

[22] Roger M. Needham and Michael D. Schroeder. Using Encryption for Authentication in Large Networks of Computers. *Communications of the ACM*, 21(12):993–999, 1978.

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

[24] Lawrence C. Paulson. The inductive approach to verifying cryptographic protocols. *Journal of computer security*, 6:85–128, 1998.

[25] Riccardo Pucella. Deductive Algorithmic Knowledge. In *Proceedings of the Eighth International Symposium on Artificial Intelligence and Mathematics*, 2004.

[26] R. Ramanujam and S.P.Suresh. Decidability of context-explicit security protocols. *Journal of Computer Security*, 13(1):135–165, 2005.

[27] R. Ramanujam and S.P. Suresh. Tagging makes secrecy decidable for unbounded nonces as well. In *Proceedings of 23rd FST&TCS*, volume 2914, pages 363–374, Mumbai, India, December 2003.

[28] Michaël Rusinowitch and Mathieu Turuani. Protocol insecurity with finite number of sessions is NP-complete. *Theoretical Computer Science*, 299:451–475, 2003.

[29] S.P. Suresh. *Foundations of Security Protocol Analysis*. PhD thesis, The Institute of Mathematical Sciences, Chennai, India, November 2003. Madras University. Available at `http://www.cmi.ac.in/~spsuresh`.