Deciding knowledge properties of security protocols

R. Ramanujam * S. P. Suresh †

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 [AN95] 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 $\mathcal{M}(\mathsf{Pr})_{\mathcal{I}} \models \alpha$: that is, whether all *runs* of the model $\mathcal{M}(\mathsf{Pr})$ associated with the protocol Pr, under the *intruder theory* $\mathcal{I}$, satisfy α. There are two important issues here, that of how the model $\mathcal{M}(\mathsf{Pr})$ 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 $\mathcal{M}(\mathsf{Pr})$ is an infinite state system and even simple properties like reachability tend to be undecidable.

*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 $\xi \restriction A$ 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 $\xi \restriction A = \xi' \restriction A$.

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 [NS78]):

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

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.

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

$\mathsf{G}\alpha$ 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 $\mathcal{R}$, about which we will discuss later.

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

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.

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

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 $\mathscr{L}_2$ 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:

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

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:

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

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, n, etc. (which stand for nonces and keys), and by repeatedly applying the tupling and encryption operators.

$$\mathscr{L}_2 \quad ::= \quad A \ \mathsf{has}\ t \mid A!B{:}t \mid A?B{:}t \mid new(n) \mid \neg\alpha \mid \alpha \vee \beta \mid \mathsf{G}\alpha \mid \mathsf{H}\alpha \mid \mathsf{K}_A\alpha$$

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 useof 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 above is undecidable for the logic of knowledge and tense, in the presence of either unbounded nonces or unbounded message length. We therefore work with the syntactic subclass of *tagged protocols* for which the secrecy problem is decidable even in the presence of both unbounded nonces and unbounded message length. (See [RS03] for a summary of the results.) In this paper, we assume a fixed finite set of nonces and show that the verification problem is decidable for the class of tagged protocols, even when message length and number of parallel sessions are unbounded (yielding an infinite state system). We further show that the result extends to the specification logic which can constrain the protocol semantics considerably.

We then consider the protocol logic, which can constrain not only the behaviours but also protocol structure. We show that the verification problem is decidable when the semantics is constrained to generate only so-called *well-typed runs* (which effectively give us finite state environments). Intuitively, well-typed runs are those where the intruder cannot substitute complex terms for atomic terms (or, at least, without getting found out). This rules out attacks where the length of terms used increases without bound, and hence makes verification easier. Knowledge theoreticaly, this amounts to enriching the knowledge of agents with a type system, since the semantics allows only runs which pass such a type inference mechanism.

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 $\mathcal{R}$.

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 [BAN90], which initiated the study of belief logics for authentication. It also gave rise to a host of descendants (see [GNY90] and [AT91], for instance). While BAN logic has been greatly criticised ([Nes90], 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. ([KW96]). 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 ([ABKL93], [ADM03], [Pau98], [Bel03], [HP03], [HvdM03], [GMP92] 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 [DMP03] and [DDMP03]: 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 [DGFvdH04] 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 [HP03], which uses a logic of implicit and explicit knowledge to reason about adversaries. [HP03] 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.

2 Security protocol modelling

We briefly present our model for protocols in this section. A more detailed presentation can be found in [RS05]. 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 [DY83].

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

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

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.

Definition 2.1 *A* **protocol** *is a pair* $\mathsf{Pr} = (\mathsf{C}, \mathsf{R})$ *where* $\mathsf{C} \subseteq \mathcal{T}_0$ *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.

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

A *substitution* σ is a partial map from $\mathcal{T}_0$ to $\mathcal{T}$ such that for all $A \in Ag$, if $\sigma(A)$ is defined then it belongs to Ag, and for all $k \in K_0$, if $\sigma(k)$ is defined then it belongs to K_0. σ is *well-typed* if for all $n \in N$ such that $\sigma(n)$ is defined, $\sigma(n) \in N$. For any $T \subseteq \mathcal{T}$, σ is said to be a T-substitution iff for all $x \in \mathcal{T}_0$, if $\sigma(x)$ is defined then $\sigma(x) \in T$.

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. For technical ease, we code up these details in the form of *events*. 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 $Events(\mathsf{Pr})$. An event is well-typed if the corresponding substitution is. 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 $e' \stackrel{+}{\rightarrow}_\ell e$.

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* $T \subseteq \mathcal{T}$ *and* $t \in \mathcal{T}$*.*

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*,* $\mathsf{analz}(T)$ *(*$\mathsf{synth}(T)$*) is the set of terms* t *such that there is an* analz*-proof (*synth*-proof) of* $T \vdash t$*. For ease of notation,* $\mathsf{synth}(\mathsf{analz}(T))$ *is denoted by* $\overline{T}$*.*

$$\frac{}{T \cup \{t\} \vdash t}\,\mathsf{Ax}_a \qquad \frac{}{T \cup \{t\} \vdash t}\,\mathsf{Ax}_s$$

$$\frac{T \vdash (t_1, t_2)}{T \vdash t_i}\,\mathsf{split}_i (i = 1, 2) \qquad \frac{T \vdash t_1 \quad T \vdash t_2}{T \vdash (t_1, t_2)}\,\mathsf{pair}$$

$$\frac{T \vdash \{t\}_k \quad T \vdash \overline{k}}{T \vdash t}\,\mathsf{decrypt} \qquad \frac{T \vdash t \quad T \vdash k}{T \vdash \{t\}_k}\,\mathsf{encrypt}$$

analz-rules　　　　synth-rules

Figure 1: analz and synth rules.

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

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

A run is well-typed if all the events occurring in it are well-typed. For any $T \subseteq \mathcal{T}$, *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* $\mathcal{R}(\mathsf{Pr})$ *(resp.* $\mathcal{R}_T(\mathsf{Pr})$*).*

3 The logics

We now give the formal semantics of the three logics and state the results. We need to define the indistinguishability relation first. 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. Given an equivalence relation $\sim$ on the set of events, we say that $e_1 \cdots e_k \sim e'_1 \cdots e'_k$ iff $e_i \sim e'_i$ for all $i \leq k$. For $\mathscr{L}_0$ and $\mathscr{L}_2$, we take $\sim$ to be equality, while for $\mathscr{L}_1$ we use a different notion. Say that σ preserves the type of x if $\sigma(x)$ is a nonce (key, agent name) exactly when x is a nonce (respectively, key, agent name). We say that $(\eta, \sigma, lp) \sim (\eta', \sigma', lp')$ iff $\eta = \eta'$, $lp = lp'$, and for all x whose type is preserved by σ or σ', $\sigma(x) = \sigma'(x)$. This is understandable, since $\mathscr{L}_1$ is a logic which works on an abstract name space, and thus can allow for a much coarser indistinguishability relation. We also need this definition to handle ill-typed substitutions in the proof of Theorem 3.2.

The indistinguishability relation on runs is as follows. 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 $(\xi(1) \cdots \xi(i))\lceil A \sim (\xi'(1) \cdots \xi'(i'))\lceil A$, where $\sim$ is the relation appropriate to the logic being considered.

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, $\xi \in \mathcal{R}(\mathsf{Pr})$ 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 $\xi' \in \mathcal{R}(\mathsf{Pr})$ 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 $\xi \in \mathcal{R}_T(\mathsf{Pr})$, $\mathsf{Pr}, (\xi, 0) \models \alpha$.

For giving the semantics of $\mathscr{L}_1$ 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 (as described in the previous section). So along with the protocol we have an assignment $\mathfrak{a} : \mathcal{R} \to \mathcal{S}$, where $\mathcal{S}$ is the set of substitutions as defined in the previous section.

Given any formula α of $\mathscr{L}_1$, a protocol Pr, an assignment $\mathfrak{a}$, a run $\xi \in \mathcal{R}(\mathsf{Pr})$ 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 $\mathscr{L}_1$:

- $(\mathsf{Pr}, \mathfrak{a}), (\xi, i) \models A_r \ \mathsf{has}\ x_{r'}$ iff $\sigma'(x) \in \overline{(\mathit{infstate}(\xi, i))_{\sigma(A)}}$, where $\mathfrak{a}(r) = \sigma$ and $\mathfrak{a}(r') = \sigma'$;
- $(\mathsf{Pr}, \mathfrak{a}), (\xi, i) \models \mathsf{sent}(A_r, B_{r'}, x_{r''})$ iff $\mathit{act}(\xi(i)) = \sigma(A)!\sigma'(B){:}(M)t$ for some M and t and $\sigma''(x) \in ST(t)$, where $\mathfrak{a}(r) = \sigma$, $\mathfrak{a}(r') = \sigma'$ and $\mathfrak{a}(r'') = \sigma''$;
- $(\mathsf{Pr}, \mathfrak{a}), (\xi, i) \models \mathsf{honest}(A_r)$ iff $\sigma(A) \neq I$, where $\mathfrak{a}(r) = \sigma$.

- $(\mathsf{Pr}, \mathfrak{a}), (\xi, i) \models \mathsf{K}_{A_r}\alpha$ iff for all $\xi' \in \mathcal{R}(\mathsf{Pr})$ and $i' \leq |\xi'|$ such that $(\xi, i) \sim_{\sigma(A)} (\xi', i')$, $(\xi', i') \models \alpha$, where $\mathfrak{a}(r) = \sigma$.

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

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

- $\mathsf{Pr}, (\xi, i) \models A \text{ has } t$ iff $t \in \overline{(\mathit{infstate}(\xi, i))_A}$;
- $\mathsf{Pr}, (\xi, i) \models A!B{:}t$ iff $\mathit{act}(\xi(i)) = A!B{:}(M)t'$ for some M and t' and $t \in ST(t')$;
- $\mathsf{Pr}, (\xi, i) \models A?B{:}t$ iff $\mathit{act}(\xi(i)) = A?B{:}t'$ for some t' and $t \in ST(t')$;
- $\mathsf{Pr}, (\xi, i) \models \mathit{new}(n)$ iff $\mathit{act}(\xi(i)) = A!B{:}(M)t'$ for some M and t' and $n \in M$;

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

Theorem 3.1 *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 ([DLMS99] or chapter 3 of [Sur03]), or unboundedly long messages [EG83].*

We therefore consider the subclass of tagged protocols in which add a distinct constant tag to each distinct encrypted subterm occurring in the protocol specification. This prevents some undesirable unification of messages. We now state the main results of the paper, which assert that the verification problem for the logics is decidable for the class of tagged protocols, under some additional restriction on the systems generating the runs of a protocol.

Theorem 3.2 *Fix a finite $T \subseteq \mathcal{T}_0$. The problem of checking for a given tagged protocol* Pr *and a formula α of $\mathscr{L}_0$ or $\mathscr{L}_1$, whether all T-runs of* Pr *satisfy α (in symbols $\mathsf{Pr} \models^T \alpha$), is decidable.*

Note that the result is strong since the existence of ill-typed substitutions forces the protocols to be infinite state systems in general (even when the set of basic terms is finite).

Theorem 3.3 *Fix a finite $T \subseteq \mathcal{T}_0$ and let T denote the set of all* well-typed *T-runs of a given protocol* Pr. *The problem of checking for a given formula α of $\mathscr{L}_2$, whether all runs in T satisfy α (in symbols $\mathsf{Pr} \models^T_{wt} \alpha$), is decidable.*

The restrictions to well-typed runs ensures that the set of events is finite, and hence we do not need to confine our attention to the subclass of tagged protocols. Note that the set of runs is still infinite because events may repeat in a run, and hence the problem is non-trivial.

References

[ABKL93] 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.

[ADM03] Kamel Adi, Mourad Debbabi, and Mohamed Mejri. A new logic for electronic commerce protocols. *Theoretical Computer Science*, 29(3):223–283, 2003.

[AN95] 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.

[AT91] 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.

[BAN90] Michael Burrows, Martin Abadi, and Roger M. Needham. A logic of authentication. *ACM Transactions on Computer Systems*, 8(1):18–36, Feb 1990.

[Bel03] Giampaolo Bella. Inductive Verification of Smartcard Protocols. *Journal of Computer Security*, 11(1):87–132, 2003.

[DDMP03] 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.

[DGFvdH04] 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.

[DLMS99] 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.

[DMP03] 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.

[DY83] Danny Dolev and Andrew Yao. On the Security of public-key protocols. *IEEE Transactions on Information Theory*, 29:198–208, 1983.

[EG83] Shimon Even and Oded Goldreich. On the security of multi-party ping-pong protocols. Technical Report 285, Technion - Israel Institute of Technology, 1983.

[GMP92] Janice Glasgow, Glenn MacEwen, and Prakash Panangaden. A logic for reasoning about security. *ACM Transactions on Computer Systems*, 10(3):226–264, Aug 1992.

[GNY90] 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.

[HP03] 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.

[HvdM03] Joseph Y. Halpern and Ron van der Meyden. A logical reconstruction of SPKI. *Journal of Computer Security*, 11(4):581–613, 2003.

[KW96] 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.

[Low96] 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.

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

[NS78] 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.

[Pau98] Lawrence C. Paulson. The inductive approach to verifying cryptographic protocols. *Journal of computer security*, 6:85–128, 1998.

[RS03] 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.

[RS05] R. Ramanujam and S.P.Suresh. Decidability of context-explicit security protocols. *Journal of Computer Security*, 13(1):135–165, 2005.

[Sur03] 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`.

Appendices

A Properties of the Needham-Schroeder protocol

We look at the Needham-Schroeder protocol in detail now, stating several of its properties in our logics. We show that some crucial properties fail. We also indicate how one can demonstrate some of the properties to be true.

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

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

The *responder role* η_2 is given below:

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

One of 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. An especially strong form of the above is expressed by the following formula:

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

Secrecy for the responder says something similar about y.

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.

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

Authentication for the responder says something similar about the responder.

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

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. This can be evidenced by the following run ξ. 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 $\sigma_2(y) = n$.

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

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. Even if the message received by B has a sender address which is that of an honest agent, B cannot rule out the possibility that the run he is in is ξ above, and thereby cannot rule out either the fact that the first x he received was actually sent to B by A or the fact that the y that he sends purportedly to A will not be known to others. This is the famous Lowe's attack [Low96] on the Needham-Schroeder protocol. Note that the kind of intuitive reasoning which happens above is all about reasoning about indistinguishable runs in the context of a protocol which is common knowledge and constrains the space of possible worlds which are reasoned about.

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 path (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.

As part of ongoing work, we are looking at ways of expressing such combinatorial proofs in terms of the knowledge of the different agents involved. We feel that the protocol logic with the same overall structure can be used to capture the essence of such arguments. But we need to explore the structure of the basic propositions more to get a logic which enables us to carry out such reasoning naturally.

B Proof outlines

In this section, we outline the proofs of Theorems 3.2 and 3.3. We need to present some more definitions before we go on to explain the proof details. The definitions make precise the subclass of protocols we are considering. Protocols presented as a set of roles admits just too many unnatural protocols. We model "natural" protocols by our notion of a well-formed protocol. In an effort to obtain decidability, we choose to rule out some features (namely unifiability of different arbitrary messages occurring in the protocol specification) which are crucially exploited in the undecidability proofs. That leads us to the definition of tagged protocols.

Definition B.1 *A* linear protocol *is a pair* $\mathsf{Pr} = (\mathsf{C}, \eta)$ *where:*

- $\mathsf{C} \subseteq \mathcal{T}_0$ *is the set of* constants *of* Pr*, and*

- *η, the body of the protocol, is a sequence of actions $a_1 b_1 \cdots a_\ell b_\ell$ such that each a_i is a send of the form $A_i!B_i{:}(M_i)t_i$ and the corresponding b_i is a matching receive of the form $B_i?A_i{:}t_i$.*

The roles of a linear protocol are got by projecting the subsequence of A-actions, for each A.

Definition B.2 *A sequence of actions $\eta = a_1 \cdots a_\ell$ is said to be* send-admissible *with respect to a state s iff for all $i \leq \ell$, if a_i is a send action then a_i is enabled at $update(s, a_1 \cdots a_{i-1})$. η is said to be send-admissible with respect to a protocol* Pr *iff it is send-admissible with respect to* $\mathsf{init}(\mathsf{Pr})$. *A* well-formed protocol *is a linear protocol $\mathsf{Pr} = (\mathsf{C}, \eta)$ such that η is send-admissible with respect to* Pr.

Definition B.3 *A well-formed protocol $\mathsf{Pr} = (\mathsf{C}, \eta)$ is called a* tagged protocol *iff for all encrypted subterms t occurring in η, there exists $\mathsf{c}_t \in \mathsf{C}$ such that:*

- *for all $t, t' \in EST(\eta)$, if $\mathsf{c}_t = \mathsf{c}_{t'}$ then $t = t'$, and*
- *for all $t \in EST(\eta)$: $t = \{(\mathsf{c}_t, u)\}_k$ for some u and k.*

We present details which are relevant to the verification problem for $\mathscr{L}_1$. Where appropriate, we also indicate the changes needed to make the proofs go through for $\mathscr{L}_2$.

There are two halves to the decidability proofs. The first half involves handling ill-typed substitutions in runs. This is where the properties of tagged protocols are crucially used. In this half, we show that the verification problem is equivalent to the problem of checking whether there is a well-typed run of Pr satisfying a given α. Since we have also fixed a finite $T \subseteq \mathcal{T}_0$, there are only finitely many T-events that we have to consider. But there might still be infinitely many well-typed T-runs of Pr, since the same event may repeat many times in a run. This feature of the model (repetition of events in a run) makes the involved, combinatorial proofs of the first half easier. But it has to be handled in the second half. In the second half, we show by a good subgraph construction that the problem of checking whether a formula is true in all well-typed runs of Pr is decidable.

Let us briefly outline the first half of the proof. Let us assume that a nonce n_0 is provided to all the agents in the initial state of every protocol. This is supposed to stand for new data that the intruder will come up with during the course of a run. Now, for any substitution σ, define σ_{n_0} (which is easily seen to be well-typed) as follows:

$$\forall x \in \mathcal{T}_0 : \sigma_{\mathsf{n}_0}(x) = \begin{cases} \mathsf{n}_0 & \text{if } x \in N \text{ and } \sigma(x) \notin N \\ \sigma(x) & \text{otherwise} \end{cases}$$

Given a tagged protocol $\mathsf{Pr} = (\mathsf{C}, \eta)$ and a run $\xi = e_1 \cdots e_k$ of Pr with $e_i = (\eta_i, \sigma_i, lp_i)$ for all $i \leq k$, we define e_{n_0} to be $(\eta_i, (\sigma_i)_{\mathsf{n}_0}, lp_i)$. We further define ξ_{n_0} to be $(e_1)_{\mathsf{n}_0} \cdots (e_k)_{\mathsf{n}_0}$. Intuitively, in ξ_{n_0}, all the ill-typed substitutions are replaced by well-typed ones (with the assignment of a default value for some of the nonces). The rationale behind this is as follows: the intruder is introducing complicated terms instead of nonces, hoping that the other agents would decrypt some information and hand it to him. But for this to happen, distinct messages in the protocol specification have to be unifiable, which is exactly what is prevented in tagged protocols. Thus for the purpose of analysis, it suffices to consider a default value for all those nonces which have been substituted with a more complicated term.

The important property to be proved is that ξ_{n_0} is a run of the protocol. This is an involved combinatorial proof, involving a lot of analysis of synth and analz proofs (introduced earlier). All the details are spelled out in Chapter 5 of [Sur03]. It is much easier to check that for any run ξ of the protocol, for any $i \leq |\xi|$, and for any *atomic formula* α, $(\xi, i) \models \alpha$ iff $(\xi_{\mathsf{n}_0}, i) \models \alpha$. Further we can easily check that $\xi \sim_A \xi'$ iff $\xi_{\mathsf{n}_0} \sim_A \xi'_{\mathsf{n}_0}$ (using the special notion of $\sim_A$ that we defined for $\mathscr{L}_1$). It can now be easily seen that $\mathsf{Pr} \models^T \alpha$ iff $\mathsf{Pr} \models^T_{wt} \alpha$. This is the first half of the decidability proofs.

We spell out the details of the second half now. These are also relevant to $\mathscr{L}_2$.

The set of subformulas of a formula is easily defined as follows:

$SF(p) = \{p\}$, for $p \in P$;
$SF(\neg\alpha) = \{\neg\alpha\} \cup SF(\alpha)$;
$SF(\alpha \vee \beta) = \{\alpha \vee \beta\} \cup SF(\alpha) \cup SF(\beta)$;
$SF(\mathsf{F}\alpha) = \{\mathsf{F}\alpha\} \cup SF(\alpha)$;
$SF(\mathsf{P}\alpha) = \{\mathsf{P}\alpha\} \cup SF(\alpha)$;
$SF(\mathsf{K}_A\alpha) = \{\mathsf{K}_A\alpha\} \cup SF(\alpha)$.

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 which satisfies the following conditions:

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 $\mathsf{P}\alpha \in \Psi$;
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 $\mathsf{F}\alpha \in \Psi_1$,
if $\mathsf{F}\alpha \in \Psi_1$ and $\alpha \notin \Psi_1$ then $\mathsf{F}\alpha \in \Psi_2$;
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_1 is an *initial atom* iff for all $\mathsf{P}\alpha \in CL$, if $\mathsf{P}\alpha \in \Psi$ then $\alpha \in \Psi$.

An atom Ψ_1 is 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 $\mathsf{K}_A\alpha \in \Psi_2$.

Definition B.4 *For any set E, the function* $\mathsf{red} : E^* \to E^*$ *is defined as follows:*

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

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

A *molecule* χ is a reduced sequence of pairs $(\mathsf{first}, \Psi_0)(e_1, \Psi_1) \cdots (e_k, \Psi_k)$ where:

- $e_1 \cdots e_k$ is a well-typed T-run of Pr,
- for all $i : 1 \leq i \leq k$, $\Psi_{i-1} \longrightarrow \Psi_i$, and
- for all $i : 0 \leq i \leq k$, and for all $p \in P$, $p \in \Psi_i$ iff $\mathit{infstate}(e_1 \cdots e_i) \models p$.

The set of molecules is denoted by $\mathscr{M}$.

Given a molecule $\chi = (\mathsf{first}, \Psi_0) \cdots (e_k, \Psi_k)$, its *current atom* is Ψ_k and its *current event* is e_k.

We say that $\chi_1 \xrightarrow{e} \chi_2$ iff $\chi_2 = \mathsf{red}(\chi_1 \cdot (e, \Psi))$ for some atom Ψ.

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

- $e_1 \cdots e_k \sim_A e'_1 \cdots e'_\ell$, and
- for all i such that e_i is an A-event, $\Psi_i \cap P_A = \Psi'_i \cap P_A$. 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 initial iff it is of the form (first, Ψ_0) and Ψ_0 is an initial atom. The set of initial molecules is denoted by $\mathscr{I}$.

A molecule χ is said to be a *final molecule* iff its current atom is final. The set of final molecules is denoted by $\mathscr{F}$.

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

Definition B.5 (Good subgraph of $\mathscr{G}$) *A subgraph $\mathscr{G}' = (\mathscr{M}', \mathscr{I}', \mathscr{F}', \longrightarrow, \sim_A)$ of $\mathscr{G}$ is called a good subgraph iff it is a maximal subgraph with the following properties:*

- *$\longrightarrow$ and $\sim_A$ on $\mathscr{G}'$ are just the restrictions of the relations on $\mathscr{G}$ to $\mathscr{M}'$,*
- *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 current atom Ψ, if $\mathsf{L}_A\alpha \in \Psi$ then there exists $\chi' \in \mathscr{M}'$ with current atom Ψ' such that $\chi \sim_A \chi'$ and $\alpha \in \Psi'$, and*

Many of the definitions are on standard lines. The definition of a good subgraph deserves some discussion. Normally in satisfiability proofs, formulas like $\mathsf{L}_A\alpha$ are requirements to be fulfilled, and thus force us to include new worlds in the model. But here we are starting out with a fixed set of worlds. In this setting, formulas like $\mathsf{L}_A\alpha$ force us to throw away worlds. This can happen if there is no indistinguishable world containing α. (We are not allowed to go out of the fixed set of worlds anyway.) On the other hand, formulas like $\mathsf{K}_A\alpha$ force us to include some worlds. This is because 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. To avoid this from happening, we require that a good subgraph be a maximal subgraph with the specified properties. This forces, 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, which can be proved by a relatively straightforward induction.

Lemma B.6 *For all runs $\xi = e_1 \cdots e_k$ of Pr and for all $i \leq |\xi|$, there is a good subgraph $\mathscr{G}'$ which contains the molecule $\chi_i = \mathsf{red}((\mathsf{first}, \Psi_0)(e_1, \Psi_1) \cdots (e_i, \Psi_i))$, where $\Psi_i = \{\alpha \in CL \mid (\xi, i) \models \alpha\}$ for all $i \leq k$, and further $\chi_{i-1} \xrightarrow{e_i} \chi_i$ for all $i : 1 \leq i \leq k$.*

Lemma B.7 *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 $\alpha \in CL$, $(\xi, i) \models \alpha$ iff $\alpha \in \Psi$, where Ψ is the current atom of χ.*

The above two lemmas immediately yield the following lemma.

Lemma B.8 *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 $\chi = (\mathsf{first}, \Psi_0)$ 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 roughly $N_1 = n_2^{n_1}$. The set of atoms is $N_2 = 2^{n_3}$. Thus we can see that the set of molecules (which are sequences of event-atom pairs) 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, it is 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.