Using Temporal Logics of Knowledge in the Formal Verification of Security Protocols

Clare Dixon, Mari-Carmen Fernández Gago, Michael Fisher and Wiebe van der Hoek

Department of Computer Science
University of Liverpool, Liverpool L69 7ZF, United Kingdom

FAX: (+44) 151 79 43715
EMAIL: {C.Dixon,M.C.Gago,M.Fisher,wiebe}@csc.liv.ac.uk
WWW: http://www.csc.liv.ac.uk/{~clare,~mari,~michael,~wiebe}

Abstract

Temporal logics of knowledge are useful for reasoning about situations where the knowledge of an agent or component is important, and where change may occur in this knowledge over time. Here we use temporal logics of knowledge to reason about security protocols. We show how to specify the Needham-Schroeder protocol using temporal logics of knowledge and prove various properties using a resolution calculus for this logic.

1 Introduction

Improved communication infrastructures encourage parties to interchange more and more sensitive data, such as payment instructions in e-commerce, strategic information between commercial partners, or personal information in, for instance, medical applications. Issues such as authentication of the partners in a protocol, together with the confidentiality of information therefore become increasingly important: cryptographic protocols are used to distribute keys and authenticate agents and data over hostile networks. Although the protocols used often look very intricate, many examples are known of sensitive applications that were 'cracked' and had to be furnished with new, 'improved' protocols. It is obvious that in such information-sensitive applications as above, one prefers to *formally prove* that certain information can not be eavesdropped by unwanted third parties.

The application of logical tools to the analysis of security protocols was pioneered by Burrows, Abadi and Needham. In [1] and [10] specific epistemic logics, collectively referred to as BAN logics, are proposed to deal with authentication issues. At the same time, standard epistemic logics ([12, 18]) have been applied successfully to reason about communication protocols, cf. the derivation of the alternating bit protocol in [14] or, more recently, the analysis of TCP [22]). In such an analysis, an epistemic language is useful in order to express that some receiver indeed *knows* some message at a specific state of the protocol, or that a sender *knows* that the receiver knows the message. In this setting, contrary to the security framework of BAN, the implicit assumption is always that the network is not hostile. In spite of the potential, more recent work on epistemic logic has mainly focused on theoretical issues such as variants of modal logic, completeness, and derived notions such as *distributed knowledge* and *common knowledge*.

Logics of knowledge are useful for specifying systems where statements such as

If John knows his own private key and John receives a message encrypted in his public key then John knows the contents of the message are required.

Statements such as the above represent static situations, in other words they describe the state of the knowledge within the world, but not how this knowledge evolves. For this, we incorporate *temporal logic*. Temporal logics have been shown to be useful for specifying dynamic systems that change over time [16]. By combining temporal and epistemic logics, we provide a logical framework in which systems requiring both dynamic aspects and informational aspects relating to knowledge can be described. This is particularly important in security protocols, where one wants to ensure that certain knowledge is obtained over time or, at least, that ignorance of potential intruders persists over the whole run of the protocol. Such *temporal logics of knowledge* have been used in the specification and verification of distributed and multi-agent systems [8, 13, 18], in analysing security protocols [23, 9], and in characterising knowledge games such as the muddy children problem [4].

The logic, $KL_{(n)}$, that we consider here is the fusion of a linear time temporal logic (comprising finite past and infinite future) with the multi-modal logic S5 (see, for example, [12] for more details about this logic). In order to prove that a particular property φ follows from a problem specification ψ, for example a security protocol, where both φ and ψ are formulae of $KL_{(n)}$ we must prove that $\vdash \psi \Rightarrow \varphi$. Since we carry out proofs using clausal resolution for temporal logics of knowledge [3, 4], which is a refutation method, we actually check that the combination of ψ and $\neg\varphi$ is unsatisfiable. This resolution calculus uses a translation to a normal form in order to separate modal and temporal components, a novel resolution method applied to the temporal part and modal resolution rules applied to the modal part. Information is carried between the two components using clauses containing only literals.

Given the above background, in this paper, we bring together specification using temporal logics of knowledge and verification using clausal resolution, and apply these to the problem of formally analysing security protocols. In order to show how such protocols can be specified and verified, we consider one very well known protocol, namely the Needham-Schroeder protocol [20]. This protocol has been widely studied with particular problems uncovered via formal analysis, for example [17].

The paper is structured as follows. In Section 2 we describe the Needham-Schroeder protocol. In Section 3 we give the syntax and semantics of a linear-time temporal logic of knowledge and in Section 4 we present a clausal resolution method for this logic. In Section 5 we show how the Needham-Schroeder protocol can be specified in this logic.

2 The Needham-Schroeder protocol with public keys

The Needham-Schroeder protocol with public keys [20] intends to establish authentication between an agent A who initiates a protocol and an agent B who responds to A.

The complete protocol consists of seven messages, but we focus on a simplified version consisting of only three messages. The messages that we omit are those whereby the agents request other agent's public keys from a server. Note that omitting these steps is equivalent to assuming that each agent always knows the others' public keys.

The protocol can then be described as the three following steps:

Message	Direction	Contents
Message 1	$A \rightarrow B$:	$\{N_A, A\}_{pub\_key(B)}$
Message 2	$B \rightarrow A$:	$\{N_B, N_A\}_{pub\_key(A)}$
Message 3	$A \rightarrow B$:	$\{N_B\}_{pub\_key(B)}$

Note that message contents of the form $\{X, Y\}_{pub\_key(Z)}$ represent messages containing both X and Y but then encrypted with Z's public key. Elements of the form N_X are special items of data, called *nonces*. Typically, agents in the protocol will generate their own unique nonce (often encrypted) which is, at least initially, unknown to all other agents.

Message 1: A sends B an encrypted nonce together with A's identity, all encrypted with B's public key.

Message 2: When B receives Message 1, it decrypts it to obtain N_A. Then B returns to A the nonce N_A and generates another nonce of his own, N_B, and sends it back, this time encrypted with A's public key.

Message 3: When A receives Message 2, it returns B's nonce, this time encrypted with B's public key in order to prove A's authenticity.

It would seem that A should be sure he is talking to B, since only B 'should' be able to decrypt Message 1. In the same way, B seems to be sure that he is talking to A since only A 'should' be able to decrypt Message 2. However, we will see later that this is not always the case.

3 Syntax and Semantics

The logic, $KL_{(n)}$, a *temporal logic of knowledge* we consider is the fusion of linear-time temporal logic with multi-modal S5. We first give the syntax and semantics of $KL_{(n)}$, where each modal relation is restricted to be an equivalence relation [15]. The temporal component is interpreted over a discrete linear model of time with finite past and infinite future; an obvious choice for such a flow of time is $(\mathbb{N}, <)$, i.e., natural numbers ordered by the usual 'less than' relation. This logic has been studied in detail [15] and is the most commonly used temporal logic of knowledge.

3.1 Syntax

Formulae are constructed from a set $\mathcal{P} = \{p, q, r, \ldots\}$ of *primitive propositions*. The language $KL_{(n)}$ contains the standard propositional connectives $\neg$ (not), $\vee$ (or), $\wedge$ (and) and $\Rightarrow$ (implies). For knowledge we assume a set of agents $Ag = \{1, \ldots n\}$ and introduce a set of unary modal connectives K_i, for $i \in Ag$, where a formula $K_i\phi$ is read as "agent i knows ϕ". For the temporal dimension we take the usual [11] set of future-time temporal connectives $\bigcirc$ (*next*), $\diamondsuit$ (*sometime* or *eventually*), $\square$ (*always*), $\mathcal{U}$ (*until*) and $\mathcal{W}$ (*unless* or *weak until*).

The set of well-formed formulae of $KL_{(n)}$, WFF_K is defined as follows:

- **false**, **true** and any element of $\mathcal{P}$ is in WFF_K;
- if A and B are in WFF_K then so are (where $i \in Ag$)

$$\begin{array}{lllll} \neg A & A \vee B & A \wedge B & A \Rightarrow B & K_i A \\ \diamondsuit A & \square A & A\,\mathcal{U}\,B & A\,\mathcal{W}\,B & \bigcirc A \end{array}$$

We define some particular classes of formulae that will be useful later.

Definition 1 *A* literal *is either p, or $\neg p$, where $p \in \mathcal{P}$.*

Definition 2 *A* modal literal *is either $K_i l$ or $\neg K_i l$ where l is a literal and $i \in Ag$.*

Notation: in the following, l are literals, m are either literals or modal literals and D are disjunctions of literals or modal literals.

3.2 Semantics

First, we assume that the world may be in any of a set, S, of *states*.

Definition 3 *A* timeline *t, is an infinitely long, linear, discrete sequence of states, indexed by the natural numbers. Let TLines be the set of all timelines.*

Definition 4 *A* point *q, is a pair $q = (t, u)$, where $t \in$ TLines is a timeline and $u \in \mathbb{N}$ is a temporal index into t. Let Points be the set of all points.*

Definition 5 *A* valuation *π, is a function $\pi : Points \times \mathcal{P} \to \{T, F\}$.*

Definition 6 *A* model *M, is a structure $M = \langle TL, R_1, \ldots, R_n, \pi \rangle$, where:*

- *$TL \subseteq TLines$ is a set of timelines, with a distinguished timeline t_0;*
- *R_i, for all $i \in Ag$ is the agent accessibility relation over Points, i.e., $R_i \subseteq Points \times Points$ where each R_i is an equivalence relation;*
- *π is a valuation.*

As usual, we define the semantics of the language via the satisfaction relation '$\models$'. For $KL_{(n)}$, this relation holds between pairs of the form $\langle M, p \rangle$ (where M is a model and p is a point in $TL \times \mathbb{N}$), and formulae in WFF_K. The rules defining the satisfaction relation are given below.

$$
\begin{array}{lll}
\langle M,(t,u)\rangle \models \textbf{true} & & \\
\langle M,(t,u)\rangle \not\models \textbf{false} & & \\
\langle M,(t,u)\rangle \models p & \text{iff} & \pi((t,u),p) = T \text{ (where } p \in \mathcal{P}) \\
\langle M,(t,u)\rangle \models \neg A & \text{iff} & \langle M,(t,u)\rangle \not\models A \\
\langle M,(t,u)\rangle \models A \lor B & \text{iff} & \langle M,(t,u)\rangle \models A \text{ or } \langle M,(t,u)\rangle \models B \\
\langle M,(t,u)\rangle \models \bigcirc A & \text{iff} & \langle M,(t,u+1)\rangle \models A \\
\langle M,(t,u)\rangle \models \square A & \text{iff} & \forall u' \in \mathbb{N},\ \text{if } (u \leq u') \text{ then } \langle M,(t,u')\rangle \models A \\
\langle M,(t,u)\rangle \models \lozenge A & \text{iff} & \exists u' \in \mathbb{N} \text{ such that } (u \leq u') \text{ and } \langle M,(t,u')\rangle \models A \\
\langle M,(t,u)\rangle \models A\,\mathcal{U}\,B & \text{iff} & \exists u' \in \mathbb{N} \text{ such that } (u' \geq u) \text{ and} \\
 & & \quad \langle M,(t,u')\rangle \models B, \text{ and } \forall u'' \in \mathbb{N}, \\
 & & \quad \text{if } (u \leq u'' < u') \text{ then } \langle M,(t,u'')\rangle \models A \\
\langle M,(t,u)\rangle \models A\,\mathcal{W}\,B & \text{iff} & \langle M,(t,u)\rangle \models A\,\mathcal{U}\,B \text{ or } \langle M,(t,u)\rangle \models \square A \\
\langle M,(t,u)\rangle \models K_i A & \text{iff} & \forall t' \in TL.\ \forall u' \in \mathbb{N}.\ \text{if } ((t,u),(t',u')) \in R_i \\
 & & \quad \text{then } \langle M,(t',u')\rangle \models A
\end{array}
$$

For any formula A, if there is some model M and timeline t such that $\langle M, (t, 0)\rangle \models A$, then A is said to be satisfiable. If for any formula A, for all models M there exists a timeline t such that $\langle M, (t, 0)\rangle \models A$ then A is said to be valid. Note, this is the anchored version of the (temporal) logic, i.e. validity and satisfiability are evaluated at the beginning of time (see for example [5]).

As agent accessibility relations in $KL_{(n)}$ models are equivalence relations, the axioms of the normal modal system S5 are valid in $KL_{(n)}$ models. The system S5 is widely recognised as the logic of idealised *knowledge*, and for this reason $KL_{(n)}$ is often termed a *temporal logic of knowledge*.

4 Resolution for Temporal Logics of Knowledge

The resolution calculus is clausal requiring a translation to a normal form to separate modal and temporal components and to put formulae in a particular form. A set of temporal resolution rules applied to the temporal part and modal resolution rules are applied to the modal part. Information is carried between the two components using clauses containing literals. Full details of resolution based proof methods for temporal logics of knowledge are given in [3, 4].

4.1 Normal Form

Formulae in $KL_{(n)}$ can be transformed into a normal form SNF$_K$ (Separated Normal Form for temporal logics of knowledge). For the purposes of the normal form we introduce a symbol **start** such that $\langle M, (t_0, 0)\rangle \models$ **start**. This is not necessary but allows the normal form to be implications. An alternative would be to let initial clauses (see Figure 1) be a disjunction of literals. The translation to SNF$_K$ removes many of the temporal operators that do not appear in the normal form by rewriting using their fixpoint definitions. Also the translation uses the renaming technique [21] where complex subformulae are replaced by new propositions and the truth value of these propositions is linked to the formulae they replaced in all states. To achieve this we introduce the $\Box^*$ operator, which allows nesting of K_i and $\Box$ operators. The $\Box^*$ operator is defined in terms of the C (or *common knowledge*) and E (or *everybody knows*) operators. We define E by $E\phi \;\Leftrightarrow\; \bigwedge_{i\in Ag} K_i\phi$. The common knowledge operator, C, is then defined as the maximal fixpoint of the formula $C\phi \;\Leftrightarrow\; E(\phi \wedge C\phi)$. Finally, the $\Box^*$ operator is defined as the maximal fixpoint of $\Box^*\phi \;\Leftrightarrow\; \Box(\phi \wedge C\Box^*\phi)$.

Thus we reason about reachable points from the initial point in the distinguished timeline t_0 (where **start** is satisfiable), i.e. the points we require in the proof.

4.1.1 Definition of the Normal Form

Formulae in SNF$_K$ are of the general form

$$\Box^* \bigwedge_j T_j$$

where each T_j, known as a *clause*, must be in one of the varieties described in Figure 1 where k_a, l_b, and l are literals and m_{ib} are either literals, or modal literals involving the K_i operator. Thus a K_i clause (also known as a modal clause) may not contain modal literals K_il_1 and K_jl_2 (or K_il_1 and $\neg K_jl_2$) where $i \neq j$. Each K_i clause involves literals, or modal literals involving the K_i operator where at least one of the disjuncts is a modal literal. The outer '$\Box^*$' operator that surrounds the conjunction of clauses is usually omitted. Similarly, for convenience the conjunction is dropped and we consider just the set of clauses T_j.

To apply the temporal resolution rule (see Section 4.2), one or more step clauses may need to be combined. Consequently, a variant on SNF$_K$ called *merged-SNF$_K$)* [6], is also defined. Given a set of step clauses in SNF$_K$, any step clause in SNF$_K$ is also a clause in SNF$_K$. Any literal clause of the form **true** $\Rightarrow F$ is written into a merged- SNF$_K$ clause as **true** $\Rightarrow \bigcirc F$. Any two merged-SNF$_K$ clauses may be combined to produce a merged-SNF$_K$ clause as follows

$$\begin{array}{rcl} A & \Rightarrow & \bigcirc C \\ B & \Rightarrow & \bigcirc D \\ \hline (A \wedge B) & \Rightarrow & \bigcirc (C \wedge D) \end{array}$$

where A and B are conjunctions of literals and C and D are conjunctions of disjunctions of literals.

$$\boxed{\begin{array}{lcll}
\textbf{start} & \Rightarrow & \bigvee_{b=1}^{r} l_b & \text{(an } \textit{initial} \text{ clause)} \\
\bigwedge_{a=1}^{g} k_a & \Rightarrow & \bigcirc \bigvee_{b=1}^{r} l_b & \text{(a } \textit{step} \text{ clause)} \\
\bigwedge_{a=1}^{g} k_a & \Rightarrow & \Diamond l & \text{(a } \textit{sometime} \text{ clause)} \\
\textbf{true} & \Rightarrow & \bigvee_{b=1}^{r} m_{ib} & \text{(a } K_i\text{–clause)} \\
\textbf{true} & \Rightarrow & \bigvee_{b=1}^{r} l_b & \text{(a literal clause)}
\end{array}}$$

Figure 1: Clauses in SNF$_K$

4.1.2 Translation to Normal Form

The translation to SNF$_K$ is carried out by renaming complex subformulae with new propositional variables and linking the truth of the subformula to that of the proposition at all moments. Temporal operators are removed using their fixpoint definitions. Classical and temporal equivalences (see for example [5]) are also used to get formulae into the correct format. See [4, 7] for more details.

4.2 Resolution Rules

The resolution rules presented are split into four groups: those concerned with initial resolution, modal resolution, step resolution and temporal resolution. As well as the resolution rules presented, simplification and subsumption also takes place. So for example the step clause $a \Rightarrow \bigcirc(b \vee b \vee c)$ is automatically rewritten as $a \Rightarrow \bigcirc(b \vee c)$.

Initial Resolution An initial clause may be resolved with either a literal clause or an initial clause as follows

$$[\text{IRES1}] \quad \begin{array}{lcl} \textbf{true} & \Rightarrow & (A \vee l) \\ \textbf{start} & \Rightarrow & (B \vee \neg l) \\ \hline \textbf{start} & \Rightarrow & (A \vee B) \end{array} \qquad [\text{IRES2}] \quad \begin{array}{lcl} \textbf{start} & \Rightarrow & (A \vee l) \\ \textbf{start} & \Rightarrow & (B \vee \neg l) \\ \hline \textbf{start} & \Rightarrow & (A \vee B) \end{array}$$

Modal Resolution During modal resolution we apply the following rules which are based on the modal resolution system introduced by Mints [19]. In the following we may only resolve two K_i clauses together if they relate to the same i, i.e. we may not resolve a clause containing K_1 with a clause containing K_2. We may resolve a literal or modal literal and its negation or the formulae $K_i l$ and $K_i \neg l$ as we cannot both know something and know its negation.

$$[\text{MRES1}] \quad \begin{array}{lcl} \textbf{true} & \Rightarrow & D \vee m \\ \textbf{true} & \Rightarrow & D' \vee \neg m \\ \hline \textbf{true} & \Rightarrow & D \vee D' \end{array} \qquad [\text{MRES2}] \quad \begin{array}{lcl} \textbf{true} & \Rightarrow & D \vee K_i l \\ \textbf{true} & \Rightarrow & D' \vee K_i \neg l \\ \hline \textbf{true} & \Rightarrow & D \vee D' \end{array}$$

Next, as we have the T axiom, $\vdash K_i p \Rightarrow p$, we can resolve formulae such as $K_i l$ with $\neg l$ (giving MRES3). The rule MRES4 requires the function $\mathrm{mod}_i(D')$, defined below, and is justified due to the external K_i operator surrounding each clause (due to the $\Box^*$ operator), i.e, we distribute K_i into the second clause and resolve $\neg K_i l$ with $K_i l$. The "mod_i" function ensures that during this distribution at most one K_i or $\neg K_i$ operator applies to each literal due to the equivalences $\neg K_i \neg K_i \varphi \Leftrightarrow K_i \varphi$ and $\neg K_i K_i \varphi \Leftrightarrow \neg K_i \varphi$ in S5.

$$\text{[MRES3]}\quad \begin{array}{rcl} \mathbf{true} & \Rightarrow & D \vee K_i l \\ \mathbf{true} & \Rightarrow & D' \vee \neg l \\ \hline \mathbf{true} & \Rightarrow & D \vee D' \end{array} \qquad \text{[MRES4]}\quad \begin{array}{rcl} \mathbf{true} & \Rightarrow & D \vee \neg K_i l \\ \mathbf{true} & \Rightarrow & D' \vee l \\ \hline \mathbf{true} & \Rightarrow & D \vee \mathrm{mod}_i(D') \end{array}$$

Definition 7 *The function $\mathrm{mod}_i(D)$, defined on disjunctions of literals or modal literals D, is defined as follows.*

$$\begin{array}{rcl} \mathrm{mod}_i(A \vee B) & = & \mathrm{mod}_i(A) \vee \mathrm{mod}_i(B) \\ \mathrm{mod}_i(l) & = & \neg K_i \neg l \end{array} \qquad \begin{array}{rcl} \mathrm{mod}_i(K_i l) & = & K_i l \\ \mathrm{mod}_i(\neg K_i l) & = & \neg K_i l \end{array}$$

Finally, we require the following rewrite rule to allow us to obtain the most comprehensive set of literal clauses for use during initial, step and temporal resolution

$$\text{[MRES5]}\quad \begin{array}{rcl} \mathbf{true} & \Rightarrow & L \vee K_i l_1 \vee K_i l_2 \vee \ldots \\ \hline \mathbf{true} & \Rightarrow & L \vee l_1 \vee l_2 \vee \ldots \end{array}$$

Here, L is a disjunction of literals.

Step Resolution 'Step' resolution consists of the application of standard classical resolution to formulae representing constraints at a particular moment in time, together with simplification rules for transferring contradictions within states to constraints on previous states (standard simplification and subsumption rules are also applied). The following resolution rules may be applied by resolving two step clauses or a step clause with a literal clause.

$$\text{[SRES1]}\quad \begin{array}{rcl} P & \Rightarrow & \bigcirc(A \vee l) \\ Q & \Rightarrow & \bigcirc(B \vee \neg l) \\ \hline (P \wedge Q) & \Rightarrow & \bigcirc(A \vee B) \end{array}$$

$$\text{[SRES2]}\quad \begin{array}{rcl} \mathbf{true} & \Rightarrow & (A \vee l) \\ Q & \Rightarrow & \bigcirc(B \vee \neg l) \\ \hline Q & \Rightarrow & \bigcirc(A \vee B) \end{array}$$

Once a contradiction within a state is found, the following rule can be used to generate additional literal clauses.

$$\text{[SRES3]}\quad \begin{array}{rcl} P & \Rightarrow & \bigcirc\mathbf{false} \\ \hline \mathbf{true} & \Rightarrow & \neg P \end{array}$$

This rule states that if, by satisfying P, a contradiction is produced, then P must never be satisfied in *any* moment. The new constraint therefore represents $\Box^* \neg P$

Termination Each cycle of initial, modal or step resolution terminates when either no new resolvents are derived, or **false** is derived in the form of either **start** $\Rightarrow$ **false** or **true** $\Rightarrow$ **false**.

Temporal Resolution The temporal resolution rule is as follows, where we resolve a sometime clause, $Q \Rightarrow \Diamond l$, with a condition, $\bigvee_{k=0}^{n} A_k$, that implies $\Box \neg l$ in the next moment (known as a *loop formula for* $\neg l$).

$$\frac{\begin{array}{rcl} \bigvee_{k=0}^{n} A_k & \Rightarrow & \bigcirc \Box \neg l \\ Q & \Rightarrow & \Diamond l \end{array}}{\begin{array}{rcl} Q & \Rightarrow & (\bigwedge\limits_{i=0}^{n} \neg A_i)\, \mathcal{W}\, l \end{array}}$$

This resolvent states that once Q is satisfied then none of the A_i should be satisfied unless l is satisfied. A systematic way of deriving $\bigvee_{k=0}^{n} A_k$ from the set of step clauses such that

$$\bigvee_{k=0}^{n} A_k \Rightarrow \bigcirc \Box \neg l$$

is described in [2] but is beyond the scope of this paper.

Translating the resolvent into SNF_K we obtain the following clauses for each i where w_l is a new proposition.

$$\begin{array}{rcl} \mathbf{true} & \Rightarrow & \neg Q \lor l \lor \neg A_i \\ \mathbf{true} & \Rightarrow & \neg Q \lor l \lor w_l \\ w_l & \Rightarrow & \bigcirc (l \lor \neg A_i) \\ w_l & \Rightarrow & \bigcirc (l \lor w_l) \end{array}$$

4.3 The temporal resolution algorithm

Given any temporal formula ψ to be shown unsatisfiable the following steps are performed.

1. Translate ψ into a set of SNF_K clauses ψ_s.
2. Perform modal and step resolution (including simplification and subsumption) until either
   (a) **true** $\Rightarrow$ **false** is derived - terminate noting ψ unsatisfiable; or
   (b) no new resolvents are generated - continue to step 3.
3. Select an eventuality from the right hand side of a sometime clause within ψ_s, for example $\Diamond l$. Search for loop formulae in $\neg l$ and generate the appropriate resolvents. If no new formulae have been generated try the next sometime clause otherwise (if new formulae have been generated) go to step 4. If there are no eventualities for which new resolvents can be derived, go to step 5.
4. Add the new resolvents to the clause-set and perform initial resolution until either
   (a) **start** $\Rightarrow$ **false** is derived - terminate noting ψ unsatisfiable; or
   (b) no new resolvents are generated - continue at step 2.
5. Perform initial resolution until either
   (a) **start** $\Rightarrow$ **false** is derived - terminate noting ψ unsatisfiable; or
   (b) no new resolvents are generated -terminate declaring ψ satisfiable.

4.4 Correctness

Firstly we can show that the transformation into SNF$_K$ preserves satisfiability.

Theorem 1 *A $KL_{(n)}$ formula A is satisfiable if, and only if, $\tau_0[A]$ is satisfiable (where τ_0 is the translation into SNF$_K$).*

Proofs analogous to those in [4, 7] will suffice.

Theorem 2 *(Soundness) Let S be a satisfiable set of SNF$_K$ clauses and T be the set of clauses obtained from S by an application of one of the resolution rules. Then T is also satisfiable.*

This can be shown by showing an application of each resolution rule preserves satisfiability. (see [4])

Theorem 3 *(Completeness) If a set of SNF$_K$ clauses is unsatisfiable then it has a refutation by the temporal resolution procedure given in this paper.*

This is carried out by constructing a graph to represent all possible models for the set of clauses. Deletions in the graph represent the application of of the resolution rules. An empty graph corresponds with the generation of false (see [4]).

5 Specifying the Needham-Schroeder Protocol in $KL_{(n)}$

In this section, we will use $KL_{(n)}$ to specify the Needham-Schroeder protocol. In particular, we will provide axioms describing the key aspects of both the system and the protocol. In order to do this we use the following syntactic conventions. Let M_1 and M_2 be variables over messages, *Key* be a variable over keys and $X, Y, \ldots$ be variables over agents. Moreover, for every agent, X, we assume there are keys $pub\_key(X)$ and $priv\_key(X)$, while in this protocol A and B are constants representing two specific agents. We identify the following predicates:

- $send(A, Msg, Key)$ is satisfied if agent A sends message *Msg* encrypted by *Key*;
- $rcv(A, Msg, Key)$ is satisfied if agent A receives message *Msg* encrypted by *Key*;
- $Msg(M_1)$ is satisfied if M_1 is a message;
- $val\_pub\_key(X, V)$ is satisfied if the value public key of X is V
- $val\_priv\_key(X, V)$ is satisfied if the value public key of X is V
- $val\_nonce(N_A, V)$ is satisfied if the value of nonce N_A is V
- $contains(M_1, M_2)$ is satisfied if the message M_2 is contained within M_1.

To simplify the description, we allow quantification and equality over the sets of agents, messages and keys. As we assume a finite set of agents, messages and keys this logic remains essentially propositional.

There are a few general assumptions:

- initially, only agent A knows the content of its own nonce N_A and only B knows the content of its own nonce N_B;
- if agent A sends messages containing its nonce or B's nonce they are encrypted with B's public key;

- if agent B sends messages containing its nonce or A's nonce they are encrypted with A's public key;
- messages sent are not guaranteed to arrive at the required destination;
- if a message does arrive at an agent, then that message must have been previously sent by an agent;
- knowledge of messages persists, i.e. agents do not forget;
- if a message is received, and the receiver knows the private key required, then the receiver will know the content of the message;

We allow a simple representation of public and private keys in terms of the unary functions '$\mathit{pub\_key}(X)$' and '$\mathit{priv\_key}(X)$'. All agents know the public keys of all other agents, but each agent's private key is only known by that agent. There is also an axiom below (Axiom 14) explaining that, if an agent receives a message, and that message is encrypted by a public key whose private key the agent knows, then the agent will then know the contents of the message.

5.1 Specifying Structural Assumptions

We begin with various structural assumptions concerning keys and message contents.

1. $\forall X, \mathit{Key}, M_1.\ \mathit{send}(X, M_1, \mathit{Key}) \Rightarrow \neg \mathit{contains}(M_1, \mathit{priv\_key}(X))$

   — agents will not reveal their private key to others

2. $\forall X, V_1, V_2.\ [\mathit{val\_pub\_key}(X, V_1) \Leftrightarrow \Box \mathit{val\_pub\_key}(X, V_1)] \wedge$
   $[\mathit{val\_priv\_key}(X, V_2) \Leftrightarrow \Box \mathit{val\_priv\_key}(X, V_2)] \wedge$
   $[\mathit{val\_nonce}(X, V_1) \Leftrightarrow \Box \mathit{val\_nonce}(X, V_1)]$

   — the public keys, private keys and nonces of all the agents remain the same during the protocol

3. $\forall X, Y, V.\ (\mathit{val\_pub\_key}(X, V) \wedge \mathit{val\_pub\_key}(Y, V)) \Rightarrow X = Y)$

   — no two agents have the same public keys

4. $\forall \mathit{Key}, M_1.$
   $(\mathit{send}(A, M_1, \mathit{Key}) \wedge [\mathit{contains}(M_1, N_A) \vee \mathit{contains}(M_1, N_B)]) \Rightarrow (\mathit{Key} = \mathit{pub\_key}(B))$

   — if agent A sends out messages containing N_A or N_B the messages must be encrypted with B's public key.

5. $\forall \mathit{Key}, M_2.\ (\mathit{send}(B, M_2, \mathit{Key}) \wedge [\mathit{contains}(M_2, N_A) \vee \mathit{contains}(M_2, N_B)]) \Rightarrow (\mathit{Key} = \mathit{pub\_key}(A))$

   — if agent B sends out messages containing N_A or N_B the messages must be encrypted with A's public key.

5.2 Specifying Scenario Assumptions

As we will be concerned with one particular scenario, namely the simple interaction between agents given above, we instantiate message contents, keys and names for this scenario.

6. $Msg(M_1) \Leftrightarrow ((M_1 = m_1) \vee (M_1 = m_2) \vee (M_1 = m_3))$
   — in this particular scenario, we just use three messages, m_1, m_2 and m_3.
7. $\forall X, Y, Z.\ \ (contains(m_1, X) \Leftrightarrow ((X = A) \vee (X = N_A)))\wedge$
   $(contains(m_2, Y) \Leftrightarrow ((Y = N_A) \vee (Y = N_B)))\wedge$
   $(contains(m_3, Z) \Leftrightarrow (Z = N_B))$
   — message m_1 contains only N_A and A, message m_2 contains only N_B and N_A and message m_3 contains only N_B
8. **start** $\Rightarrow$ $val\_priv\_key(A, a_v) \wedge val\_priv\_key(B, b_v) \wedge val\_priv\_key(C, c_v)\wedge$
   $\neg val\_priv\_key(A, b_v) \wedge \neg val\_priv\_key(A, c_v) \wedge \neg val\_priv\_key(B, a_v)\wedge$
   $\neg val\_priv\_key(B, c_v) \wedge \neg val\_priv\_key(C, a_v) \wedge \neg val\_priv\_key(C, b_v)\wedge$
   $val\_pub\_key(A, a) \wedge val\_pub\_key(B, b) \wedge val\_pub\_key(C, c))\wedge$
   $\neg val\_pub\_key(A, b) \wedge \neg val\_pub\_key(A, c) \wedge \neg val\_pub\_key(B, a)\wedge$
   $\neg val\_pub\_key(B, c) \wedge \neg val\_pub\_key(C, a)) \wedge \neg val\_pub\_key(C, b))\wedge$
   $val\_nonce(N_A, a_n) \wedge val\_nonce(N_B, b_n) \wedge val\_nonce(N_C, c_n)\wedge$
   $\neg val\_nonce(N_A, b_n) \wedge \neg val\_nonce(N_A, c_n) \wedge \neg val\_nonce(N_B, a_n)\wedge$
   $\neg val\_nonce(N_B, c_n) \wedge \neg val\_nonce(N_C, a_n) \wedge \neg val\_nonce(N_C, b_n)\wedge$

5.3 Specifying Basic Knowledge Axioms

We here specify the basic attributes of an agent's knowledge.

9. **start** $\Rightarrow \forall X.(\exists V.\ K_X val\_nonce(N_X, V))\ \wedge\ [\forall Y, Z.\ (Y \neq X) \Rightarrow \neg K_Y val\_nonce(N_X, Z)]$
   — initially agents only know their own nonces.
10. $\forall X.\ (\exists V. K_X val\_priv\_key(Y, V) \Leftrightarrow (X = Y))$ — agents only know their own private keys
11. $\forall X.\ K_X val\_pub\_key(A, a) \wedge K_X val\_pub\_key(B, b) \wedge K_X val\_pub\_key(C, c)$ — all agents know all the public keys.
12. $\forall X, N, V.\ K_X val\_nonce(N, V)\ \Rightarrow\ \bigcirc K_X val\_nonce(N, V)$ — agents never forget nonces they know
13. $\forall X, Y, V.\ K_X val\_priv\_key(Y, V)\ \Rightarrow\ \bigcirc K_X val\_priv\_key(Y, V)$ — agents never forget private keys they know

5.4 Specifying Communication Axioms

We now specify the communication between agents, and how this affects the agent's knowledge. For convenience, we allow the use of past-time temporal operators, in particular "$\circledcirc$", meaning in the previous moment in time, and "$\Diamond$", meaning at some time in the past. These operators have the following semantics.

$$\langle M, (t, u)\rangle \models \circledcirc A \quad \text{iff} \quad u > 0 \text{ and } \langle M, (t, u-1)\rangle \models A$$
$$\langle M, (t, u)\rangle \models \Diamond A \quad \text{iff} \quad \exists u' \in \mathbb{N} \text{ such that } (0 \leq u' < u) \text{ and } \langle M, (t, u')\rangle \models A$$

14. $\forall X, M_1, N_1 \bigcirc((Msg(M_1) \wedge contains(M_1, N_1)) \Rightarrow (\exists V_1 K_X val\_nonce(N_1, V_1) \Leftrightarrow$ $\circledcirc[K_X val\_nonce(N_1, V_1) \vee (\exists Y.\exists V.\ rcv(X, M_1, pub\_key(Y)) \wedge K_X val\_priv\_key(Y, V))]))$

    — for all moments except the first moment if M_1 is a message which contains N_1 an agent knows the content of N_1 either if it already knew the content of N_1, or if it received an encrypted version of M_1 that it could decode.

15. $\forall X, Key, M_1.\ rcv(X, M_1, Key) \Rightarrow \exists Y.\ \blacklozenge\, send(Y, M_1, Key)$

    — if an agent receives a message, then there was some agent that previously sent that message

16. $\forall X, Key, M_1, N_1\ \ (send(X, M_1, Key) \wedge contains(M_1, N_1) \quad \Rightarrow \quad \exists V_1.\ K_X val\_nonce(N_1, V_1) \vee$ $\blacklozenge\, rcv(X, M_1, Key)$

    — if an agent sends a message M_1 encrypted with Key, then it must either know the contents M_1 or just be forwarding the encrypted message as a whole

Note that there is no axiom such as

$$send(\ldots) \Rightarrow \Diamond rcv(\ldots)$$

as messages cannot be guaranteed to be delivered. However, we do have a related axiom (Axiom 15) which says that, if a message is received, then some other agent must have sent it previously.

5.5 Axioms in Normal Form

We can translate all the above axioms into the normal form. As an example, we will assume three agents A, B and C and translate axioms 6, 7, 9, 10, and 11, as in Fig. 2.

We can also write axiom 14 into normal form as follows for all agents X.

$$\begin{array}{lll}
(\bigcirc(Msg(M_1) \wedge contains(M_1, N_1)) \Rightarrow & & \\
\quad (\exists V_1 \bigcirc K_X val\_nonce(N_1, V_1) & \Leftrightarrow & [K_X val\_nonce(N_1, V_1) \vee \\
 & & (\exists Y.\exists V\ rcv(X, M_1, pub\_key(Y)) \wedge K_X val\_priv\_key(Y, V))])) \\
(\bigcirc(Msg(M_1) \wedge contains(M_1, N_1)) \Rightarrow & & \\
\quad (\exists V_1 \bigcirc K_X val\_nonce(N_1, V_1) & \Leftrightarrow & [K_X val\_nonce(N_1, V_1) \vee \\
 & & (\exists V.(rcv(X, M_1, pub\_key(A)) \wedge K_X val\_priv\_key(A, V)) \vee \\
 & & \exists V.(rcv(X, M_1, pub\_key(B)) \wedge K_X val\_priv\_key(B, V)) \vee \\
 & & \exists V.(rcv(X, M_1, pub\_key(C)) \wedge K_X val\_priv\_key(C, V)))]))
\end{array}$$

We provide a version below where X is instantiated as B, M_1 is instantiated as m_1, N_1 is instantiated as N_A and d_1, d_2, d_3, d_4, e_1, e_2, e_3, f_1, f_2,and f_3 are new propositions. Note due to axiom 14 containing the $\Leftrightarrow$ operator we must rename formulae below this using $\Leftrightarrow$ rather than just $\Rightarrow$.

In the following d_1 is a new name for the following

$$d_1 \Leftrightarrow (rcv(B, m_1, pub\_key(A)) \wedge (K_B val\_priv\_key(A, a_v) \vee K_B val\_priv\_key(A, b_v) \vee K_B val\_priv\_key(A, c_v)))$$

and similarly for d_2 and d_3 where A in the above are replaced by B and C respectively. d_4 is a new name for the disjunction of these formulae i.e.

$$d_4 \Leftrightarrow d_1 \vee d_2 \vee d_3$$

f_1 is a new name for B's knowledge that the value of N_A is a_n i.e.

$$f_1 \Leftrightarrow K_B val\_nonce(N_A, a_n)$$

$6a.$ $\mathbf{true} \Rightarrow Msg(m_1)$

$6b.$ $\mathbf{true} \Rightarrow Msg(m_2)$

$6c.$ $\mathbf{true} \Rightarrow Msg(m_3)$

$7a.$ $\mathbf{true} \Rightarrow contains(m_1, A)$

$7b.$ $\mathbf{true} \Rightarrow contains(m_1, N_A)$

$7c.$ $\mathbf{true} \Rightarrow contains(m_2, N_A)$

$7d.$ $\mathbf{true} \Rightarrow contains(m_2, N_B)$

$7e.$ $\mathbf{true} \Rightarrow contains(m_3, N_B)$

$9a.$ $\mathbf{start} \Rightarrow (K_A val\_nonce(N_A, a_n) \lor K_A val\_nonce(N_A, b_n) \lor K_A val\_nonce(N_A, c_n))$

$9b.$ $\mathbf{start} \Rightarrow (K_B val\_nonce(N_B, a_n) \lor K_B val\_nonce(N_B, b_n) \lor K_B val\_nonce(N_B, c_n))$

$9c.$ $\mathbf{start} \Rightarrow (K_C val\_nonce(N_C, a_n) \lor K_C val\_nonce(N_C, b_n) \lor K_C val\_nonce(N_C, c_n))$

$9d.$ $\mathbf{start} \Rightarrow \neg K_A val\_nonce(N_B, b_n)$

$9e.$ $\mathbf{start} \Rightarrow \neg K_A val\_nonce(N_C, c_n)$

$9f.$ $\mathbf{start} \Rightarrow \neg K_B val\_nonce(N_A, a_n)$

$9g.$ $\mathbf{start} \Rightarrow \neg K_B val\_nonce(N_C, c_n)$

$9h.$ $\mathbf{start} \Rightarrow \neg K_C val\_nonce(N_A, a_n)$

$9i.$ $\mathbf{start} \Rightarrow \neg K_C val\_nonce(N_B, b_n)$

$10a.$ $\mathbf{true} \Rightarrow K_A val\_priv\_key(A, a_v) \lor K_A val\_priv\_key(A, b_v) \lor K_A val\_priv\_key(A, c_v)$

$10b.$ $\mathbf{true} \Rightarrow K_B val\_priv\_key(B, a_v) \lor K_B val\_priv\_key(B, b_v) \lor K_B val\_priv\_key(B, c_v)$

$10c.$ $\mathbf{true} \Rightarrow K_C val\_priv\_key(C, a_v) \lor K_C val\_priv\_key(C, b_v) \lor K_C val\_priv\_key(C, c_v)$

$10d.$ $\mathbf{true} \Rightarrow \neg K_A val\_priv\_key(B, a_v)$

$10e.$ $\mathbf{true} \Rightarrow \neg K_A val\_priv\_key(B, b_v)$

$10f.$ $\mathbf{true} \Rightarrow \neg K_A val\_priv\_key(B, c_v)$

$10g.$ $\mathbf{true} \Rightarrow \neg K_A val\_priv\_key(C, a_v)$

$10h.$ $\mathbf{true} \Rightarrow \neg K_A val\_priv\_key(C, b_v)$

$10i.$ $\mathbf{true} \Rightarrow \neg K_A val\_priv\_key(C, c_v)$

$10j.$ $\mathbf{true} \Rightarrow \neg K_B val\_priv\_key(A, a_v)$

$10k.$ $\mathbf{true} \Rightarrow \neg K_B val\_priv\_key(A, b_v)$

$10l.$ $\mathbf{true} \Rightarrow \neg K_B val\_priv\_key(A, c_v)$

$10m.$ $\mathbf{true} \Rightarrow \neg K_B val\_priv\_key(C, a_v)$

$10n.$ $\mathbf{true} \Rightarrow \neg K_B val\_priv\_key(C, b_v)$

$10o.$ $\mathbf{true} \Rightarrow \neg K_B val\_priv\_key(C, c_v)$

$10p.$ $\mathbf{true} \Rightarrow \neg K_C val\_priv\_key(A, a_v)$

$10q.$ $\mathbf{true} \Rightarrow \neg K_C val\_priv\_key(A, b_v)$

$10r.$ $\mathbf{true} \Rightarrow \neg K_C val\_priv\_key(A, c_v)$

$10s.$ $\mathbf{true} \Rightarrow \neg K_C val\_priv\_key(B, a_v)$

$10t.$ $\mathbf{true} \Rightarrow \neg K_C val\_priv\_key(B, b_v)$

$10u.$ $\mathbf{true} \Rightarrow \neg K_C val\_priv\_key(B, c_v)$

$11a.$ $\mathbf{true} \Rightarrow K_A val\_pub\_key(A, a)$

$11b.$ $\mathbf{true} \Rightarrow K_A val\_pub\_key(B, b)$

$11c.$ $\mathbf{true} \Rightarrow K_A val\_pub\_key(C, c)$

$11d.$ $\mathbf{true} \Rightarrow K_B val\_pub\_key(A, a)$

$11e.$ $\mathbf{true} \Rightarrow K_B val\_pub\_key(B, b)$

$11f.$ $\mathbf{true} \Rightarrow K_B val\_pub\_key(C, c)$

$11g.$ $\mathbf{true} \Rightarrow K_C val\_pub\_key(A, a)$

$11h.$ $\mathbf{true} \Rightarrow K_C val\_pub\_key(B, b)$

$11i.$ $\mathbf{true} \Rightarrow K_C val\_pub\_key(C, c)$

Figure 2: Normal form translation of axioms 6, 7, 9, 10, and 11

where f_2 and f_3 are similarly defined except a_n is replaced by b_n and c_n respectively. The right hand side of the implication becomes

$$(\exists V_1 \bigcirc K_X \mathit{val\_nonce}(N_1, V_1) \;\Leftrightarrow\; (K_X \mathit{val\_nonce}(N_1, V_1) \lor d_4))$$

Removing the existential quantifier we obtain

$$(\bigcirc(\mathit{Msg}(M_1) \land \mathit{contains}(M_1, N_1)) \Rightarrow (e_1 \lor e_2 \lor e_3)$$

where

$$e_1 \Rightarrow (\bigcirc f_1 \Leftrightarrow (f_1 \lor d_4))$$

and similarly for e_2 and e_3 where f_1 is replaced by f_2 and f_3 respectively.

14 $(\neg e_1 \land \neg e_2 \land \neg e_3) \Rightarrow \bigcirc(\neg \mathit{Msg}(m_1) \lor \neg \mathit{contains}(m_1, N_A))$

14 $(e_1 \land \neg f_1 \land \neg d_4) \Rightarrow \bigcirc \neg f_1$

14 $(e_1 \land f_1) \Rightarrow \bigcirc f_1$

14 $(e_1 \land d_4) \Rightarrow \bigcirc f_1$

14 $(e_2 \land \neg f_2 \land \neg d_4) \Rightarrow \bigcirc \neg f_2$

14 $(e_2 \land f_2) \Rightarrow \bigcirc f_2$

14 $(e_2 \land d_4) \Rightarrow \bigcirc f_2$

14 $(e_3 \land \neg f_3 \land \neg d_4) \Rightarrow \bigcirc \neg f_3$

14 $(e_3 \land f_3) \Rightarrow \bigcirc f_3$

14 $(e_3 \land d_4) \Rightarrow \bigcirc f_3$

14 $\mathbf{true} \Rightarrow \neg f_1 \lor K_B \mathit{val\_nonce}(N_A, a_n)$

14 $\mathbf{true} \Rightarrow f_1 \lor \neg K_B \mathit{val\_nonce}(N_A, a_n)$

14 $\mathbf{true} \Rightarrow \neg f_2 \lor K_B \mathit{val\_nonce}(N_A, b_n)$

14 $\mathbf{true} \Rightarrow f_2 \lor \neg K_B \mathit{val\_nonce}(N_A, b_n)$

14 $\mathbf{true} \Rightarrow \neg f_3 \lor K_B \mathit{val\_nonce}(N_A, c_n)$

14 $\mathbf{true} \Rightarrow f_3 \lor \neg K_B \mathit{val\_nonce}(N_A, c_n)$

14 $\mathbf{true} \Rightarrow \neg d_4 \lor d_1 \lor d_2 \lor d_3$

14 $\mathbf{true} \Rightarrow d_4 \lor \neg d_1$

14 $\mathbf{true} \Rightarrow d_4 \lor \neg d_2$

14 $\mathbf{true} \Rightarrow d_4 \lor \neg d_3$

14 $\mathbf{true} \Rightarrow \neg d_1 \lor \mathit{rcv}(B, m_1, \mathit{pub\_key}(A))$

14 $\mathbf{true} \Rightarrow \neg d_1 \lor (K_B \mathit{val\_priv\_key}(A, a_v) \lor K_B \mathit{val\_priv\_key}(A, b_v) \lor K_B \mathit{val\_priv\_key}(A, c_v)$

14 $\mathbf{true} \Rightarrow \neg \mathit{rcv}(B, m_1, \mathit{pub\_key}(A)) \lor d_1 \lor \neg K_B \mathit{val\_priv\_key}(A, a_v)$

14 $\mathbf{true} \Rightarrow \neg \mathit{rcv}(B, m_1, \mathit{pub\_key}(A)) \lor d_1 \lor \neg K_B \mathit{val\_priv\_key}(A, b_v)$

14 $\mathbf{true} \Rightarrow \neg \mathit{rcv}(B, m_1, \mathit{pub\_key}(A)) \lor d_1 \lor \neg K_B \mathit{val\_priv\_key}(A, c_v)$

14 $\mathbf{true} \Rightarrow \neg d_2 \lor \mathit{rcv}(B, m_1, \mathit{pub\_key}(B))$

14 $\mathbf{true} \Rightarrow \neg d_2 \lor (K_B \mathit{val\_priv\_key}(B, a_v) \lor K_B \mathit{val\_priv\_key}(B, b_v) \lor K_B \mathit{val\_priv\_key}(B, c_v)$

14 $\mathbf{true} \Rightarrow \neg \mathit{rcv}(B, m_1, \mathit{pub\_key}(B)) \lor d_2 \lor \neg K_B \mathit{val\_priv\_key}(B, a_v)$

14 $\mathbf{true} \Rightarrow \neg \mathit{rcv}(B, m_1, \mathit{pub\_key}(B)) \lor d_2 \lor \neg K_B \mathit{val\_priv\_key}(B, b_v)$

14 $\mathbf{true} \Rightarrow \neg \mathit{rcv}(B, m_1, \mathit{pub\_key}(B)) \lor d_2 \lor \neg K_B \mathit{val\_priv\_key}(B, c_v)$

14 $\mathbf{true} \Rightarrow \neg d_3 \lor \mathit{rcv}(B, m_1, \mathit{pub\_key}(C))$

14 $\mathbf{true} \Rightarrow \neg d_3 \lor (K_B \mathit{val\_priv\_key}(C, a_v) \lor K_B \mathit{val\_priv\_key}(C, b_v) \lor K_B \mathit{val\_priv\_key}(C, c_v)$

14 $\mathbf{true} \Rightarrow \neg \mathit{rcv}(B, m_1, \mathit{pub\_key}(C)) \lor d_3 \lor \neg K_B \mathit{val\_priv\_key}(C, a_v)$

14 $\mathbf{true} \Rightarrow \neg \mathit{rcv}(B, m_1, \mathit{pub\_key}(C)) \lor d_3 \lor \neg K_B \mathit{val\_priv\_key}(C, b_v)$

14 $\mathbf{true} \Rightarrow \neg \mathit{rcv}(B, m_1, \mathit{pub\_key}(C)) \lor d_3 \lor \neg K_B \mathit{val\_priv\_key}(C, c_v)$

Note further that the following is a consequence of axiom 15.

$$15a.\ \textbf{start} \Rightarrow \neg rcv(B, m_1, pub\_key(B))$$

6 Verifying Properties of the Specification

Once we have the above axioms relating to the specific scenario, we can attempt to prove various statements. Refutations tend to be quite long, so we will only provide detail for the simpler examples.

6.1 B's Knowledge on Receipt of N_A

The first example will capture the statement

once B receives the nonce of A encoded by B's public key then B knows the nonce of A

This can be translated into $KL_{(n)}$ as

$$\square(rcv(B, m1, pub\_key(B)) \Rightarrow \bigcirc \exists V.\ K_B val\_nonce(N_A, V))$$

or removing the existential quantifier as follows.

$$\square(rcv(B, m1, pub\_key(B)) \Rightarrow \bigcirc(K_B val\_nonce(N_A, a_n) \vee K_B val\_nonce(N_A, b_n) \vee K_B val\_nonce(N_A, c_n))$$

We will now show how to establish this using the resolution method outlined earlier. To prove the above, we negate the statement and derive the following set of clauses.

$$\begin{array}{lrcl}
p1. & \textbf{start} & \Rightarrow & x \\
p2. & x & \Rightarrow & \Diamond y \\
p3. & \textbf{true} & \Rightarrow & \neg y \vee rcv(B, m_1, pub\_key(B)) \\
p4. & y & \Rightarrow & \bigcirc z \\
p5. & \textbf{true} & \Rightarrow & \neg z \vee \neg K_B val\_nonce(N_A, a_n) \\
p6. & \textbf{true} & \Rightarrow & \neg z \vee \neg K_B val\_nonce(N_A, b_n) \\
p7. & \textbf{true} & \Rightarrow & \neg z \vee \neg K_B val\_nonce(N_A, c_n)
\end{array}$$

In the following we will need an instantiation of axiom 13

$$K_B val\_priv\_key(B, b_v) \Rightarrow \bigcirc K_B val\_priv\_key(B, b_v)$$

which is translated into normal form as follows

$$\begin{array}{lrcl}
13 & p_1 & \Rightarrow & \bigcirc p_1 \\
13 & \textbf{true} & \Rightarrow & \neg p_1 \vee K_B val\_priv\_key(B, b_v) \\
13 & \textbf{true} & \Rightarrow & p_1 \vee \neg K_B val\_priv\_key(B, b_v)
\end{array}$$

The proof takes place as follows.

$p8.$	**true**	$\Rightarrow$	$\neg z \lor \neg f_1$	$[ax14, p5\ \mathit{MRES}1]$
$p9.$	**true**	$\Rightarrow$	$\neg z \lor \neg f_2$	$[ax14, p6\ \mathit{MRES}1]$
$p10.$	**true**	$\Rightarrow$	$\neg z \lor \neg f_3$	$[ax14, p7\ \mathit{MRES}1]$
$p11.$	$(e_1 \land d_4)$	$\Rightarrow$	$\bigcirc \neg z$	$[ax14, p8\ \mathit{SRES}2]$
$p12.$	$(e_2 \land d_4)$	$\Rightarrow$	$\bigcirc \neg z$	$[ax14, p9\ \mathit{SRES}2]$
$p13.$	$(e_3 \land d_4)$	$\Rightarrow$	$\bigcirc \neg z$	$[ax14, p10\ \mathit{SRES}2]$
$p14.$	$(e_1 \land d_4 \land y)$	$\Rightarrow$	$\bigcirc$**false**	$[p4, p11\ \mathit{SRES}1]$
$p15.$	$(e_2 \land d_4 \land y)$	$\Rightarrow$	$\bigcirc$**false**	$[p5, p12\ \mathit{SRES}1]$
$p16.$	$(e_3 \land d_4 \land y)$	$\Rightarrow$	$\bigcirc$**false**	$[p6, p13\ \mathit{SRES}1]$
$p17.$	**true**	$\Rightarrow$	$(\neg e_1 \lor \neg d_4 \lor \neg y)$	$[p14\ \mathit{SRES}3]$
$p18.$	**true**	$\Rightarrow$	$(\neg e_2 \lor \neg d_4 \lor \neg y)$	$[p15\ \mathit{SRES}3]$
$p19.$	**true**	$\Rightarrow$	$(\neg e_3 \lor \neg d_4 \lor \neg y)$	$[p16\ \mathit{SRES}3]$
$p20.$	$(\neg e_1 \land \neg e_2 \land \neg e_3)$	$\Rightarrow$	$\bigcirc \neg \mathit{contains}(m_1, N_A)$	$[ax6a, ax14\ \mathit{SRES}2]$
$p21.$	$(\neg e_1 \land \neg e_2 \land \neg e_3)$	$\Rightarrow$	$\bigcirc$**false**	$[ax7b, p20\ \mathit{SRES}2]$
$p22.$	**true**	$\Rightarrow$	$(e_1 \lor e_2 \lor e_3)$	$[p21\ \mathit{SRES}3]$
$p23.$	**true**	$\Rightarrow$	$(e_2 \lor e_3 \lor \neg d_4 \lor \neg y)$	$[p17, p22\ \mathit{MRES}1]$
$p24.$	**true**	$\Rightarrow$	$(e_3 \lor \neg d_4 \lor \neg y)$	$[p18, p23\ \mathit{MRES}1]$
$p25.$	**true**	$\Rightarrow$	$(\neg d_4 \lor \neg y)$	$[p19, p24\ \mathit{MRES}1]$
$p26.$	**true**	$\Rightarrow$	$(\neg d_2 \lor \neg y)$	$[ax14, p25\ \mathit{MRES}1]$
$p27.$	**true**	$\Rightarrow$	$(\neg \mathit{rcv}(B, m_1, \mathit{pub\_key}(B)) \lor$ $\neg K_B \mathit{val\_priv\_key}(B, b_v) \lor \neg y)$	$[ax14, p26\ \mathit{MRES}1]$
$p28.$	**true**	$\Rightarrow$	$(\neg K_B \mathit{val\_priv\_key}(B, b_v) \lor \neg y)$	$[p3, p27\ \mathit{MRES}1]$
$p29.$	**true**	$\Rightarrow$	$(\neg p_1 \lor \neg y)$	$[ax13, p28\ \mathit{MRES}1]$
$p30.$	p_1	$\Rightarrow$	$\bigcirc \neg y$	$[ax13, p29\ \mathit{SRES}2]$

we can apply the temporal resolution rule with clauses $p30$ and axioms 13 together giving

$$p_1 \Rightarrow \bigcirc \square \neg y$$

for resolution with $p2$.

$p31.$	**true**	$\Rightarrow$	$\neg x \lor y \lor \neg p_1$	$[p30, ax13, p2\ \mathit{TRES}]$
$p32.$	**true**	$\Rightarrow$	$\neg x \lor y \lor w_y$	$[p30, ax13, p2\ \mathit{TRES}]$
$p33.$	w_y	$\Rightarrow$	$\bigcirc (y \lor \neg p_1)$	$[p30, ax13, p2\ \mathit{TRES}]$
$p34.$	w_y	$\Rightarrow$	$\bigcirc (y \lor w_l)$	$[p30, ax13, p2\ \mathit{TRES}]$
$p35.$	**true**	$\Rightarrow$	$\neg x \lor y \lor \neg K_B \mathit{val\_priv\_key}(B, b_v)$	$[ax13, p31\ \mathit{MRES}1]$
$p36.$	**true**	$\Rightarrow$	$\neg x \lor y \lor K_B \mathit{val\_priv\_key}(B, a_v) \lor K_B \mathit{val\_priv\_key}(B, c_v)$	$[ax10b, p35\ \mathit{MRES}1]$
$p37.$	**true**	$\Rightarrow$	$\neg x \lor y \lor \mathit{val\_priv\_key}(B, a_v) \lor \mathit{val\_priv\_key}(B, c_v)$	$[p36\ \mathit{SRES}5]$
$p38.$	**true**	$\Rightarrow$	$\neg x \lor \mathit{rcv}(B, m_1, \mathit{pub\_key}(B)) \lor \mathit{val\_priv\_key}(B, a_v)$ $\lor \mathit{val\_priv\_key}(B, c_v)$	$[p37, p3\ \mathit{MRES}1]$
$p39.$	**start**	$\Rightarrow$	$\neg x \lor \mathit{val\_priv\_key}(B, a_v) \lor \mathit{val\_priv\_key}(B, c_v)$	$[ax15a, p38\ \mathit{IRES}1]$
$p40.$	**start**	$\Rightarrow$	$\neg x \lor \mathit{val\_priv\_key}(B, c_v)$	$[ax8, p39\ \mathit{IRES}2]$
$p41.$	**start**	$\Rightarrow$	$\neg x$	$[ax8, p40\ \mathit{IRES}2]$
$p42.$	**start**	$\Rightarrow$	**false**	$[p1, p41\ \mathit{IRES}2]$

6.2 C's Ignorance

A key part of this protocol is that information is transferred between agents *A* and *B* without agent *C* ever being able to intercept sensitive information. We can verify this by showing that, in the scenario above, *C* will never know the value of A's nonce, i.e.

$$\forall V.\ \Box \neg K_C value\_nonce(N_A, V)$$

Rather than giving a full refutation, we will indicate how this can be proved.

As axiom 14 states, for an agent (in this case C) to know the value of N_A then either it knew it originally, or it received a message that contained it.

In the first case, agent C did not know the value of N_A originally.

In the second case, by axioms 3 and 4, agents A and B only ever send out messages containing N_A encrypted by $pub\_key(B)$ or $pub\_key(A)$, respectively.

Consequently, C can never know the value of N_A.

6.3 Confirmation of B's Knowledge

Once A receives m_2 (which, in turn, contains N_A) back, then it can infer that B knows N_A, i.e.

$$rcv(A, m_2, pub\_key(A)) \Rightarrow \bigcirc K_A K_B N_A)$$

Recall that axiom 15 states that if an agent receives a message, then there must have been a previous corresponding send. Since A did not send the message, the only choice is

$$\blacklozenge\, send(B, m_2, pub\_key(A)) \;\vee\; \blacklozenge\, send(C, m_2, pub\_key(A))$$

However, in order to send a message, an agent must know the contents of that message. Since we know (from above) that C doesn't know the value of N_A, then C can not have sent the message. Consequently, B must have sent the message and so A can infer that $K_B N_A$. Form this, we can infer $K_A K_B N_A$.

7 Lowe's attack on the Protocol

In the following section we describe an attack on the protocol as suggested by Lowe [17]. First we outline some general assumptions about the intruder. The intruder is able to:-

- overhear and intercept messages being passed in the system,
- decrypt messages that are encrypted with its own public key, then it can learn them,
- introduce new messages with the nonces it knows,
- replay any message he has seen, even if he was not able to decrypt it.
- if the intruder cannot learn the message, then it can remember the encrypted part and can pass it to other agents.

Lowe considers a situation where A could run the protocol with an enemy C, then C could pretend is he A and could start a new run of the protocol with B [17]. The situation could be described as follows:

Message	Direction	Contents
Message 1	$A \rightarrow C$:	$(N_A, A)_{pub\_key(C)}$
Message 2	$C(A) \rightarrow B$:	$(N_A, A)_{pub\_key(B)}$
Message 3	$B \rightarrow C(A)$:	$(N_A, N_B)_{pub\_key(A)}$
Message 4	$C(A) \rightarrow A$:	$(N_A, N_B)_{pub\_key(A)}$
Message 5	$A \rightarrow C$:	$(N_B)_{pub\_key(C)}$
Message 6	$C(A) \rightarrow B$:	$(N_B)_{pub\_key(B)}$

In message 1 A starts running the protocol with C, sending the nonce N_A, encrypted with C's public key. In message 2 the intruder impersonates A (denoted above as $C(A)$) to start a run of the protocol with B, sending the same nonce that A sent before but encrypted with B's public key. Then B replies by sending a new nonce N_B to C, but it encrypts it with A's public key (message 3). C cannot decrypt the message, but it forwards it to A in order to obtain N_B (message 4). In message 5 A sends N_B to C with C's public key, therefore he now knows N_B. Then he returns N_B to B making him believe that the protocol has been correctly run with A. However, now A, B and C all know the nonces N_A and N_B.

In the previous set of axioms this attack is disallowed due to axiom 4, i.e. A is only allowed to send messages containing N_A or N_B enctypted in B's public key. If axiom 4 is not present, then agent A is permitted to send out a message containing N_A with a public key different from that of B. Consequently, we are not then able to prove $\forall V.\ \Box\neg K_C value\_nonce(N_A, V)$ and so we cannot be sure that B, and only B, knows N_A.

8 Concluding Remarks

We have used a combination of non-classical logics, in particular the fusion of linear time temporal logic with the multi-modal logic S5 (representing knowledge) to represent and reason about security protocols. In particular we have specified the Needham-Schroeder Protocol using temporal logics of knowledge giving axioms relating to communication mechanisms, knowledge etc. We have proved various properties of this specification by using a resolution calculus for this logic and briefly discussed a well known attack on this protocol.

Future work involves the specification and verification of other protocols in this logic. The consideration of further protocols would also help us identify other suitable combinations of temporal and modal logics. We would like to develop tools to carry out the verification of the required properties. At the moment we have a prototype resolution theorem prover for the single modal version of temporal logics of knowledge but this needs extending to deal with the multi-modal case in order to prove theorems automatically. Further, we would like to develop programs that animate the reason for proof failure to assist the designers of protocols to detect flaws.

References

[1] M. Burrows, M. Abadi and R. Needham, A Logic of Authentication, *ACM Transactions on Computer Systems,* vol. 8, p. 18–36, 1990.

[2] C. Dixon. Temporal Resolution using a Breadth-First Search Algorithm. *Annals of Mathematics and Artificial Intelligence*, 22:87–115, 1998.

[3] C. Dixon and M. Fisher. Resolution-Based Proof for Multi-Modal Temporal Logics of Knowledge. In S. Goodwin and A. Trudel, editors, *Proceedings of TIME-00 the Seventh International Workshop on Temporal Representation and Reasoning*, Cape Breton, Nova Scotia, Canada, July 2000. IEEE Press.

[4] C. Dixon, M. Fisher, and M. Wooldridge. Resolution for Temporal Logics of Knowledge. *Journal of Logic and Computation*, 8(3):345–372, 1998.

[5] E. A. Emerson. Temporal and Modal Logic. In J. van Leeuwen, editor, *Handbook of Theoretical Computer Science,* pages 996–1072. Elsevier Science Publishers B.V.: Amsterdam, The Netherlands, 1990.

[6] M. Fisher. A Resolution Method for Temporal Logic. In *Proceedings of the Twelfth International Joint Conference on Artificial Intelligence (IJCAI)*, pages 99–104, Sydney, Australia, August 1991. Morgan Kaufman.

[7] M. Fisher, C. Dixon, and M. Peim. Clausal Temporal Resolution. *ACM Transactions on Computational Logic*, 2(1):12–56, January 2001.

[8] M. Fisher and M. Wooldridge. On the Formal Specification and Verification of Multi-Agent Systems. *International Journal of Cooperative Information Systems*, 6(1), January 1997.

[9] J. Glasgow, G. MacEwen, and P.Panangaden. A Logic to Reason About Security. *ACM Transactions on Computer Systems*, 10(3):226–264, August 1992.

[10] L. Gong, R. Needham and R. Yahalom, Reasoning about Belief in Cryptographic Protocol Analysis, *Proc. IEEE Symp. on Research in Security and Privacy*, p. 234–248, 1990.

[11] D. Gabbay, A. Pnueli, S. Shelah, and J. Stavi. The Temporal Analysis of Fairness. In *Proceedings of the Seventh ACM Symposium on the Principles of Programming Languages*, pages 163–173, Las Vegas, Nevada, January 1980.

[12] R. Fagin, J. Y. Halpern, Y. Moses, and M. Y. Vardi. *Reasoning About Knowledge*. MIT Press, 1995.

[13] J. Y. Halpern. Using reasoning about knowledge to analyze distributed systems. *Annual Review of Computer Science*, 2, 1987.

[14] J.Y. Halpern and L.D. Zuck, A Little Knowledge Goes a Long Way: Simple Knowledge-Based Derivations and Correctness Proofs for a Family of Protocols, *Proc. 6th ACM Symp. on Principles of Distributed Computing*, 1987, p. 268–280.

[15] J. Y. Halpern and M. Y. Vardi. The Complexity of Reasoning about Knowledge and Time. I Lower Bounds. *Journal of Computer and System Sciences*, 38:195–237, 1989.

[16] Z. Manna and A. Pnueli. *The Temporal Logic of Reactive and Concurrent Systems: Specification*. Springer-Verlag, New York, 1992.

[17] G. Lowe. Breaking and Fixing the Needham-Schroeder Public-key Protocol Using csp and fdr. In T. Margaria and B. Steffen, editors, *Tools and Algorithms for the Construction and Analysis of Systems: second international workshop, TACAS '96*, volume 1055 of *Lecture Notes in Computer Science*, pages 147–166. Spinger, 1996.

[18] J.-J. C. Meyer and W. van der Hoek. *Epistemic Logic for Computer Science and Artificial Intelligence*, volume 41 of *Cambridge Tracts in Theoretical Computer Science*. Cambridge University Press, 1995.

[19] G. Mints. Gentzen-Type Systems and Resolution Rules, Part I: Propositional Logic. *Lecture Notes in Computer Science*, 417:198–231, 1990.

[20] R.M. Needham and M.D. Schroeder. Using Encryption for Authentication in Large Networks of Computers. *Communications of the ACM*, 21:993–999, 1978.

[21] D. A. Plaisted and S. A. Greenbaum. A Structure-Preserving Clause Form Translation. *Journal of Symbolic Computation*, 2(3):293–304, September 1986.

[22] F. Stulp and R. Verbrugge, A knowledge-based algorithm for the Internet protocol TCP, to appear in the *Bulletin of Economic Research*, 2001.
Also at `http://tcw2.ppsw.rug.nl/prepublications`

[23] P. Syverson. Adding Time to a Logic of Authentication. In *Proceedings of the First ACM Conference on Computer and Communications Security*, pages 97–101. ACM Press, 1993.