Available online at www.sciencedirect.com
ScienceDirect
Electronic Notes in Theoretical Computer Science
ELSEVIER
Electronic Notes in Theoretical Computer Science 186 (2007) 27–42
www.elsevier.com/locate/entcs

Temporal Logics of Knowledge and their Applications in Security

Clare Dixon[a], Mari-Carmen Fernández Gago[b], Michael Fisher[a] and Wiebe van der Hoek[a]

[a] *Department of Computer Science The University of Liverpool, Liverpool L69 7ZF, UK*
{clare,michael,wiebe}@csc.liv.ac.uk

[b] *Departamento de Lenguajes y Ciencias de la Computación, University Of Malaga, 29071 Málaga, Spain*
mcgago@lcc.uma.es

Abstract

Temporal logics of knowledge are useful for reasoning about situations where the knowledge of an agent or component is important, and where change in this knowledge may occur over time. Here we investigate the application of temporal logics of knowledge to the specification and verification of security protocols. We show how typical assumptions relating to authentication protocols can be specified. We consider verification methods for these logics, in particular, focusing on proofs using clausal resolution. Finally we present experiences from using a resolution based theorem prover applied to security protocols specified in temporal logics of knowledge.

Keywords: Temporal logics of knowledge, security, verification, theorem proving, resolution

1 Introduction

As more and more information is transferred electronically, and as the sensitivity of this data increases, there is an increasing need to provide mechanisms for ensuring the *security* of information transfer between computers. In addition, with the advent of grid and ubiquitous computing, security and trust are key issues that are integral to the success of these technologies. This need for security has led to the development of *cryptographic protocols*, in which authentication of the individuals and messages concerned (typically through encryption) occurs. Given the importance of the area, it is vital that techniques be used which provide automatic analysis of such protocols prior to deployment.

Ever since the security of information transfer between computers became an issue, it has been clear that formal methods should have a key part to play in analysing protocol security. A wide range of tools for formal verification applicable to security have been developed [36]. Typically, these involve either full (or at

doi:10.1016/j.entcs.2006.11.043

least abstracted) state-space exploration, for example via model checking [33,38], or complex theorem-proving using higher-order tools [41,13]. While both these types of approach have made significant advances, neither is completely satisfactory: the first has limitations with infinite state spaces (even with abstraction techniques); the second suffers from the complexity of the human intervention required, and the lack of heuristics having a high probability of success. A more appealing solution lies in between these two, namely to use proof methods, but to restrict the logic to decidable or semi-decidable fragments. While such a restriction is not always possible, there are a wide range of cryptographic protocols where analysis of this style can be effective. In particular, many authentication techniques that can be captured in terms of the change in the participants' *knowledge* and *belief* fall into this category.

The application of logical tools to the analysis of security protocols was pioneered by Burrows, Abadi and Needham (BAN logics) [3]. These are purpose built belief logics especially designed for dealing with authentication issues. Whilst these logics were influential, they suffered from a number of drawbacks such as a lack of semantics, the inability to express time explicitly, the need to capture additional features, etc. This led to the development of extensions, or logics of a similar style aimed at a larger class of systems such as [21,50]. In this paper we describe how to use a combination of standard, well-known modal and temporal logics, rather than developing special purpose logics, to specify security protocols.

Temporal logic is an extension of classical logic that is widely used in the specification and verification of complex systems. In this logic, time is an added dimension and so logical properties can evolve over time; consequently this form of logic is very suitable for systems that change over time, such as dynamic or adaptive systems. Temporal logic has been used in many areas within Computer Science and Artificial Intelligence, including the specification and verification of reactive (e.g. distributed or concurrent) systems [34], the synthesis of programs from temporal specifications [42,31], algorithmic verification via model-checking [26,7], knowledge representation and reasoning [15,2,52], and temporal databases [6,5].

Modal logics involve reasoning over possible worlds. The worlds are connected by relations for each agent or process denoting possible alternative worlds. A formula φ is *possible*, denoted $\Diamond_i\varphi$ in a world s if and only if there is a world t such that (s,t) is in the i-relation and φ holds in t. A formula φ is *necessary*, denoted $\Box_i\varphi$ in a world s if and only if for all worlds t if (s,t) is in the i-relation then φ holds at t. Modal logics have been used to capture notions such as knowledge, belief, desire, intention (see for example [15,44]) etc. In particular, the modal logic S5 has been used to reason about knowledge [15]. As we are interested in knowledge, we use the operator K_i instead of $\Box_i$ and $\neg K_i\neg$ instead of $\Diamond_i$, where $K_i\varphi$ denotes that agent i knows φ.

Together combinations of logics have been used to specify complex systems such as distributed or multi-agent systems. When specifying such systems we may need to represent dynamic, informational and motivational aspects of systems. Dynamic aspects are typically modelled using temporal or dynamic logics, informational aspects

such as knowledge or belief using the modal logics S5 and KD45 and motivational aspects such as desires, intentions and wishes using the modal logics KD.

In this paper we will concentrate on a specific temporal logic of knowledge [15,37]. This is a temporal logic enriched by the addition of modal connectives for representing the knowledge of a group of processes, using the modal logic S5. These logics can be used to formalise statements such as: "if process p_1 *knows* that process p_2 has received message m_1, then p_1 should *eventually* send message m_2 to p_2".

The structure of this paper is as follows. In Section 2 we describe a particular authentication protocol, namely the Needham-Schroeder protocol. In Section 3 we provide a syntax and semantics for the temporal logic of knowledge used in this paper, $KL_{(n)}$, while in Section 4, we discuss how protocols such as the Needham-Schroeder protocol can be specified using this logic. In Section 5 we give an overview to a resolution based proof method for temporal logics of knowledge. In Section 6 we provide results having carried out proofs by hand and with an automatic theorem prover for a related logic. In Section 7 we mention related work and in Section 8 we provide concluding remarks.

2 The Needham-Schroeder protocol with public keys

The Needham-Schroeder protocol with public keys [40] 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 agents' public keys from a server. Note that omitting these steps is equivalent to assuming that each agent always knows all 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 using its corresponding private key 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.

A *man in the middle* attack on this protocol has been described by Lowe [33]. Agent A tries to run the protocol with a dishonest agent C. C pretends to be A and runs the protocol with B. Steps of the two protocol runs are interleaved and the outcome is that B thinks he has been running the protocol with A whereas he has actually been running it with C. Consequently C learns B's nonce.

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 [24]. 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 [24] 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 [19] set of future-time temporal connectives $\bigcirc$ (*next*), $\Diamond$ (*sometime*, or *eventually*), $\Box$ (*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 $i \in Ag$)

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

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

Definition 3.1 A *literal* is either p, or $\neg p$, where $p \in \mathcal{P}$.

Definition 3.2 A *modal literal* is either $K_i l$ or $\neg K_i l$ where l is a literal and $i \in Ag$.

3.2 Semantics

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

Definition 3.3 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 3.4 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 3.5 A *valuation* π, is a function $\pi : Points \times \mathcal{P} \to \{T, F\}$.

Definition 3.6 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; and
- 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 $\langle M, q \rangle$ (where M is a model and q is a point in $TL \times \mathbb{N}$), and formulae in WFF_K. The rules defining the satisfaction relation are given below. We omit the semantics of some of the classical operators as they are standard and the temporal operators $\mathcal{U}$ and $\mathcal{W}$ as they will not be used further in this paper.

$$\langle M, (t,u) \rangle \models \textbf{true}$$

$$\langle M, (t,u) \rangle \not\models \textbf{false}$$

$$\langle M, (t,u) \rangle \models p \quad \text{iff} \quad \pi((t,u), p) = T \qquad (\text{where } p \in \mathcal{P})$$

$$\langle M, (t,u) \rangle \models \neg A \quad \text{iff} \quad \langle M, (t,u) \rangle \not\models A$$

$$\langle M, (t,u) \rangle \models A \vee B \quad \text{iff} \quad \langle M, (t,u) \rangle \models A \text{ or } \langle M, (t,u) \rangle \models B$$

$$\langle M, (t,u) \rangle \models \bigcirc A \quad \text{iff} \quad \langle M, (t, u+1) \rangle \models A$$

$$\langle M, (t,u) \rangle \models \Box A \quad \text{iff} \quad \forall u' \in \mathbb{N}, \text{ if } (u \leq u') \text{ then } \langle M, (t,u') \rangle \models A$$

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

$$\langle M, (t,u) \rangle \models K_i A \quad \text{iff} \quad \forall t' \in TL.\ \forall u' \in \mathbb{N}.\ \text{if } ((t,u),(t',u')) \in R_i \text{ then } \langle M, (t',u') \rangle \models A$$

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 [14]).

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 Specification using Temporal Logics of Knowledge

In this section, we will give an overview on how to use $KL_{(n)}$ to specify the Needham-Schroeder protocol. In order to do this we use the following syntactic conventions. Let M, M_1 and M_2 be variables over messages, W be a variable over keys, N be a variable over nonces, V be a variable over values for keys and nonces, 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:

- $send(X, M, W)$ is satisfied if agent X sends message M encrypted by key W;
- $rcv(X, M, W)$ is satisfied if agent X receives message M encrypted by key W;
- $Msg(M)$ is satisfied if M is a message;
- $nonce(N)$ is satisfied if N is a nonce;
- $val\_pub\_key(X, V)$ is satisfied if the value of the public key of X is V;
- $val\_priv\_key(X, V)$ is satisfied if the value of the private key of X is V;
- $val\_nonce(N, V)$ is satisfied if the value of nonce N 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, keys and nonces this logic remains essentially propositional. The following are examples of typical 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;
- messages sent are not guaranteed to arrive at the required destination;
- if a message is received by an agent, then that message must have been previously sent by some agent;
- knowledge of messages persists, i.e. agents do not forget message contents;
- if a message is received, and the receiver knows the private key required, then the receiver will know the content of the message; and
- an intruder can intercept messages sent to others.

We allow a simple representation of public and private keys in terms of the unary functions '$pub\_key(X)$' and '$priv\_key(X)$'. All agents know the public keys of all other agents, but each agent's private key is only known by that agent.

Rather than give the complete set of axioms we provide some examples showing how to specify the above assumptions.

- *Initially, only agent A knows the value of its own nonce and only B knows the value of its own nonce.*

  $$K_A val\_nonce(N_A, a_n) \quad \neg K_A val\_nonce(N_B, a_n) \quad \neg K_A val\_nonce(N_B, b_n) \text{ etc} \ldots$$

  In the above a_n and b_n are particular values for nonces and there will be similar formulae relating to B's knowledge. Note that these statements are the initial conditions and must hold at the beginning of time.

- *Messages sent are not guaranteed to arrive.*
  There is no axiom of the form

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

- *If any message does arrive, then that message must have been previously sent by some agent.*

  $$\forall X, M, W.\ rcv(X, M, W) \ \Rightarrow\ \exists Y.\ \blacklozenge\, send(Y, M, W)$$

  Note that '$\blacklozenge$' is the operator *sometime in the past* and has the following semantics.

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

  Whilst the use of this past-time operator makes specifying this axiom easier (see also, for example, [32]), it is well known that for finite past such operators add no extra expressive power [19,32] and can be translated using just future-time operators.

- *Agents' knowledge of nonces and keys persists.*

  $$\forall X, N, V.\ K_X val\_nonce(N, V) \ \Rightarrow\ \bigcirc K_X val\_nonce(N, V)$$

  This is the axiom for nonces and there is a similar axiom for keys.

- *If a message is received containing a nonce, and the receiver knows the private key required, then the receiver will know the value of the nonce.*

  $$\begin{aligned} &\forall X, M, Y, V, N. \\ &(rcv(X, M, pub\_key(Y)) \wedge K_X val\_priv\_key(Y, V) \wedge \\ &\qquad Msg(M) \wedge contains(M, N) \wedge nonce(N)) \\ &\qquad\qquad \Rightarrow \exists V_1 K_X val\_nonce(N, V_1) \end{aligned}$$

  The actual axiom in [4,12] is more complex than this as it also incorporates the idea that the knowledge of some information must come about from either having known something in a previous moment, if there was one, or from the initial

conditions or from having been sent a message which the agent could decrypt containing that information.

- *An intruder can intercept messages sent to others.*
  The predicate *send* does not contain who the message is sent to and the predicate *rcv* doesn't contain information relating to who sent the message. Similarly, there is no axiom stating that receiving a message implies that the message must have been sent *to that agent* by a third party.

For full details of the specification see the axioms in [4,12].

5 Resolution for Temporal Logics of Knowledge

We give a brief overview of the resolution method for $KL_{(n)}$; for full details see [11,10]. To show that a $KL_{(n)}$formula, φ, is valid we negate it and translate into a normal form, SNK_K. Formulae in normal form are of the form

$$\Box^* \bigwedge_i T_i$$

where

$$\langle M,(t,u)\rangle \models \Box^* \bigwedge_i T_i \quad \text{iff} \quad \langle M,(t',u')\rangle \models \bigwedge_i T_i$$

for every point (t',u') reachable from (t,u) using either temporal or epistemic transitions of arbitrary length.

Each T_i is known as a *clause* and must be one of the following:

$$\begin{aligned}
\mathbf{start} &\Rightarrow \bigvee_{b=1}^{r} l_b && \text{(an } \textit{initial} \text{ clause)}\\
\bigwedge_{a=1}^{g} k_a &\Rightarrow \bigcirc \bigvee_{b=1}^{r} l_b && \text{(a } \textit{step} \text{ clause)}\\
\bigwedge_{a=1}^{g} k_a &\Rightarrow \Diamond l && \text{(a } \textit{sometime} \text{ clause)}\\
\mathbf{true} &\Rightarrow \bigvee_{b=1}^{r} m_{1_b} && \text{(a } K_1\text{–clause)}\\
\ldots &\Rightarrow \ldots\\
\mathbf{true} &\Rightarrow \bigvee_{b=1}^{r} m_{n_b} && \text{(a } K_n\text{–clause)}\\
\mathbf{true} &\Rightarrow \bigvee_{b=1}^{r} l_b && \text{(a } \textit{literal}\text{–clause)}
\end{aligned}$$

where k_a, l_b, and l are literals (propositions or their negations) and m_{j_b} are either literals or are of the form $K_j l$ or $\neg K_j l$ (modal literals).

The translation to SNF_K removes all temporal operators apart from $\bigcirc$ (in the next moment in time) and $\diamondsuit$ (sometime in the future) by: rewriting them using their fixpoint definitions; renaming complex subformulae by new propositional variables where the truth value of these new propositions is linked to the formulae they replaced at all moments in time; and by using standard equivalences. Then resolution rules are applied to clauses until either $\mathbf{start} \Rightarrow \mathbf{false}$ is derived, meaning 'φ is valid', or until no new clauses can be generated, meaning 'φ is not valid'. The method has been shown to be sound, complete and terminating [11,10]. Rather than giving full details, we next provide a brief overview of the key aspects of the approach. Resolution rules can be of one of four types: initial, step, modal or temporal.

The following step resolution rules, applied between two step clauses or between a step and a literal clause, are similar to classical style resolution rules.

$$\text{[SRES1]}\quad \frac{P \Rightarrow \bigcirc(F \vee l) \qquad Q \Rightarrow \bigcirc(G \vee \neg l)}{(P \wedge Q) \Rightarrow \bigcirc(F \vee G)} \qquad \text{[SRES2]}\quad \frac{P \Rightarrow \bigcirc(F \vee l) \qquad \mathbf{true} \Rightarrow (G \vee \neg l)}{P \Rightarrow (F \vee G)}$$

There are similar initial resolution rules for resolving two initial clauses or an initial and literal clause.

The following (also termed a step resolution rule) states 'if Q leads to a contradiction then $\neg Q$ must hold everywhere'.

$$\text{[SRES2]}\quad \frac{Q \Rightarrow \bigcirc \mathbf{false}}{\mathbf{true} \Rightarrow \neg Q}$$

There is also a complex temporal resolution rule that resolves sets of step clauses that together imply $P \Rightarrow \bigcirc \Box l$ with a sometime clause of the form $Q \Rightarrow \diamondsuit \neg l$. Explanation of this is beyond the scope of this paper and the interested reader is referred to [17].

Modal resolution rules are applied between two K_i clauses (for some i) or between a K_i clause and a literal clause. These relate to the axioms of the modal logic S5. For example, the following rule:

$$\text{[MRES3]}\quad \frac{\mathbf{true} \Rightarrow D \vee K_i l \qquad \mathbf{true} \Rightarrow D' \vee \neg l}{\mathbf{true} \Rightarrow D \vee D'}$$

relates to the axiom $K_i\varphi \Rightarrow \varphi$ as $K_i l$ and $\neg l$ cannot both hold together.

Theorem 5.1 Translation to SNF_K preserves satisfiability *[11,10]. A KL_n formula A is satisfiable if, and only if, $\tau_K[A]$ is satisfiable (where τ_K is the translation into SNF_K).*

Theorem 5.2 Termination*[11,10]. The resolution procedure terminates.*

Theorem 5.3 Soundness*[11,10]. 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.*

Theorem 5.4 Completeness*[11,10]. If a set of SNF_K clauses is unsatisfiable then it has a refutation by the given temporal resolution procedure.*

6 Verifying Properties of NSP

We next describe how we used resolution based proof methods for $KL_{(n)}$ to verify simple properties of NSP.

6.1 Resolution Proof in Temporal Logics of Knowledge

We have proved several (simple) properties of NSP by hand using clausal resolution for $KL_{(n)}$. Full details are given in [4,12]. The properties we have considered are given below.

- *Once B receives the nonce of A encoded by B's public key then B knows the value of that nonce.*
  This may be formalised as

$$\Box(rcv(B, m1, pub\_key(B)) \Rightarrow \bigcirc K_B val\_nonce(N_A, a_n)).$$

  Here m_1 represents the first message of NSP, i.e. A's identity and nonce encoded in B's public key and a_n is the actual value of A's nonce.
- *Once A receives its nonce back encoded via A's public key then A knows that B knows the value of that nonce*, i.e.

$$\Box(rcv(A, m_2, pub\_key(A)) \Rightarrow \bigcirc K_A K_B val\_nonce(N_A, a_n))$$

  Here m_2 represents the second message of NSP, i.e. a message containing both A's and B's nonces encoded in A's public key and, as previously, a_n is the actual value of A's nonce.
- *C will never know the value of A's nonce.*

$$\Box\neg K_C value\_nonce(N_A, a_n)$$

  Here C represents some other intruder agent and, as previously, a_n is the actual value of A's nonce. This can be proved as we have used axioms stating that A and B can only send messages containing their nonces in the public keys of A and B. Whilst this is a strong axiom, if we allowed A or B to send messages containing their nonces in C's public key it is obvious that C will be able to learn their contents.

6.2 Proof Using the Theorem Prover TeMP

Next we describe using a resolution theorem prover for monodic First-Order Temporal Logic (TeMP) to carry out these proofs automatically. First-Order (discrete linear time) Temporal Logic (FOTL) is an extension of classical first-order logic with operators that deal with a linear and discrete model of time. A formula, φ of FOTL is *monodic* if any subformula $\mathcal{T}\phi$ (where $\mathcal{T}$ is $\Box$, $\Diamond$, or $\bigcirc$) or any subformula $\phi_1 \mathcal{T} \phi_2$ (where $\mathcal{T}$ is $\mathcal{U}$ or $\mathcal{W}$) contains *at most* one free variable. Whilst full FOTL is incomplete, the monodic fragment has a finite axiomatisation [53] and may also be decidable if the pure first-order part is restricted to be a decidable fragment [25].

TeMP [27], is an implementation of a resolution-based calculus for monodic First-Order Temporal Logic over expanding domains [28]. TeMP implements the calculus described in [29], which is an implementable version of the monodic temporal resolution calculus of [8].

It is possible to translate formulae from temporal logics of knowledge into the monodic fragment of First-Order Temporal Logic (see for example [18,46] for translations from modal to first-order logics). Essentially, the truth of propositions in worlds are encoded by a unary predicate, accessibility relations between worlds are encoded by binary predicates, and restrictions on accessibility relations such as reflexivity, symmetry, transitivity (recall the modal accessibility relations are equivalence relations) are encoded as conditions on the binary relations. We provide such a translation in [16] and prove it is satisfiability preserving.

Using this approach, we have translated the specification of the NSP and the properties we wish to prove into FOTL and have run TeMP on the output. We have been able to prove the above properties provided we utilise the axiomatic translation [46] which is amenable to automatic theorem proving. In the axiomatic translation, the transitivity axiom is dealt with in a non-standard way to avoid termination problems which may occur with the standard translation. Full details of this approach, timings, the number of clauses generated, etc, can be found in [16].

7 Related Work

The application of logical tools to the analysis of security protocols was pioneered by Burrows, Abadi and Needham [3]. Known as BAN logic, this was an early approach to formalising the description and analysis of authentication protocols. BAN logic is a logic of belief, and it is the 'beliefs' of the computational components that are modified as the protocol progresses. Primitives in BAN allow statements such as A *believes* X, A *sees* M etc. The rules in these logics are such as "if an agent A sees M encrypted with a key K and he believes K is a good key for talking with another agent then A has seen M".

Whilst BAN was very influential it suffered from a number of drawbacks. Firstly BAN had no formal semantics. Although a modal logic semantics was given to a BAN-like logic in [1], this is complex. Also, when using BAN logics the protocol *idealisation* step, i.e. transforming the informal description of the protocol to one using BAN primitives, is far from easy. Further, BAN originally had no explicit

temporal component so could not express statements such as *if a message is received then it was sent at some moment in the past*, and suffered from difficulties in dealing with negated beliefs, i.e. the absence of information.

Since the introduction of BAN logic, many other logics in this style have been derived, some attempting to address the above drawbacks. These have either been extensions of BAN logic (e.g. GNY [21]) or logics developed in a similar style, but aimed at a larger class of systems (e.g. SvO [50]). What the majority of these techniques have in common is that their underlying basis can be seen as a multi-modal logic (typically *knowledge*, or *belief*). Indeed, the SvO logic is an S5 logic for cryptographic protocol analysis that unifies several earlier logics and provides a semantic basis for them. In this work, it is clear than some of the problems that have occurred require more *dynamic* notations for their solution. This has led to the addition of temporal aspects to these logics [49].

In [20] a temporal logic of knowledge is described also incorporating obligation and permission for reasoning about security policies. However the temporal part uses a simplified branching time logic.

Other approaches to the formal verification of security protocols involve, for example, the use of process algebras [47,45], tools for higher order logics [41,13] and model checking [33,38,45].

Whilst some of the verification tools mentioned above are general purpose tools that have been used for the formal verification of security protocols, there are other tools developed specifically for the analysis of security protocols. It is worth mentioning the NRL Protocol Analyzer [35] (where NRL stands for Naval Research Laboratory where this tool was developed). This analyzer has been successful in finding unpublished vulnerabilities.

Regarding verification for other types of protocol, standard epistemic logics [15,37] have been applied successfully to reason about communication protocols, for example the derivation of the alternating bit protocol in [22] or, more recently, the analysis of TCP [48]). 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.

Temporal logics of knowledge (and belief) have been widely studied see for example [15,24,30,37,51]. As well as the resolution based proof methods for these logics, tableau style algorithms also exist, see for example [54] for temporal logics of knowledge and belief, or [43,44] for the fusion of either linear or branching-time temporal logic with the modal logics KD45 for belief, and KD for desire and intention.

8 Concluding Remarks

We have given an overview of how to specify security protocols, in particular the Needham-Schroeder protocol, using temporal logics of knowledge. We have proved properties of this protocol both by hand and automatically using the theorem prover

TeMP via a translation to the monodic fragment of First-Order Temporal Logic. Using proof in temporal logics of knowledge for the verification of security protocols has a number of advantages. Firstly, these logics are well known and well studied, having formal semantics and a bank of work relating to axiomatisations, complexity results etc. Secondly, time is incorporated explicitly to deal with the ordering of events ensuring that these logics are able to describe dynamic aspects of protocols. Verification can then be achieved via proof within an appropriate logic. Thus we avoid having to develop special purpose logics and proof methods specifically for the security domain. Dealing with negated information presents us with no problems and negated properties can be easily specified. Further, Lowe's attack [33] suggests that it may be worthwhile to use both knowledge and belief in the specification, as agents can be misled. Technically this presents us with no problems, since we just use a temporal logic combined with suitable modal logics for knowledge (S5) and belief (KD45).

Whilst this approach has a number of advantages it does have some drawbacks. We carried out the specification by hand, encoding the assumptions, the protocol, and initial situation. This task is non-trivial and prone to error. Ideally we would like to produce a framework suggesting suitable axioms, given particular assumptions. This problem is similar to that of idealisation in BAN, i.e. moving from an informal protocol description to one using BAN primitives. On the other hand, our experience is that spelling out such a specification makes crystal clear what assumptions are needed to show a protocol guarantees certain desirable properties.

Another issue is the size of the specification. Apparently simple protocols such as the NSP require many axioms and initial conditions. The first-order aspects of some of these axioms makes a huge specification when written propositionally. Future work involves re-considering these axioms to see whether we can abstract away from some of the detail (such as the values of keys and nonces) while still being able to prove useful properties of the protocol.

Regarding our explicit representation of time, several axioms deal with *no forgetting* or *perfect recall* assumptions, i.e. if we know something now (eg the value of a nonce) then we will also know this in the next moment. Thus when, incorporating time explicitly we have to state axioms relating to not forgetting the value of nonces, keys etc. An alternative to this would be to use a temporal logic of knowledge with more complex interactions between knowledge and time. For example if we allow the following as an axiom

$$K_i \bigcirc p \Rightarrow \bigcirc K_i p,$$

meaning "if agent i knows that, in the next moment, p will be true then, in the next moment, agent i will know that p is true" then we can describe how knowledge evolves over time. Systems with the above axiom have been termed systems of *synchrony and perfect recall* [23]. Such logics, allowing non-trivial interactions between the modal and temporal components become, in general, theoretically more complex, sometimes undecidable [23]. Proof methods for these complex logics have been developed in [9,39].

References

[1] Abadi, M. and M. R. Tuttle, *A Semantics for a Logic of Authentication*, in: L. Logrippo, editor, *10th Annual ACM Symposium on Principles of Distributed Computing* (1991), pp. 201–216.

[2] Artale, A. and E. Franconi, *Temporal Description Logics*, in: *Handbook of Temporal Reasoning in Artificial Intelligence*, Foundations of Artificial Intelligence, Elsevier, 2005 pp. 375–388.

[3] Burrows, M., M. Abadi and R. Needham, *A Logic for Authentication*, Proceedings of the Royal Society of London **426(1871)** (1989), pp. 233–271.

[4] C. Dixon and M.-C. Fernández Gago and M. Fisher and W. van der Hoek, *Using Temporal Logics of Knowledge in the Formal Verification of Security Protocols*, in: *Proceedings of TIME 2004 the Eleventh International Symposium on Temporal Representation and Reasoning* (2004), pp. 148–151.

[5] Chomicki, J., *Efficient Checking of Temporal Integrity Constraints Using Bounded History Encoding*, ACM Transactions on Database Systems **20(2)** (1995), pp. 149–186.

[6] Chomicki, J. and D. Toman, *Temporal Logic in Information Systems*, in: *Logics for Databases and Information Systems*, 1998, pp. 31–70.

[7] Clarke, E., O. Grumberg and D. A. Peled, "Model Checking," MIT Press, 2000.

[8] Degtyarev, A., M. Fisher and B. Konev, *Monodic Temporal Resolution*, ACM Transactions on Computational Logic **7(1)** (2006).

[9] Dixon, C. and M. Fisher, *Clausal Resolution for Logics of Time and Knowledge with Synchrony and Perfect Recall*, in: H. Wansing and F. Wolter, editors, *Proceedings of the Third International Conference on Temporal Logic (ICTL)*, Leipzig, Germany, 2000, pp. 43–52.

[10] Dixon, C. 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* (2000), pp. 69–78.

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

[12] Dixon, C., M.-C. F. Gago, M. Fisher and W. van der Hoek, *Using Temporal Logics of Knowledge in the Formal Verification of Security Protocols*, Technical Report ULCS-03-022, University of Liverpool, Department of Computer Science (2003), `http://www.csc.liv.ac.uk/research/techreports`.

[13] Dutertre, B. and S. Schneider, *Using a PVS Embedding of CSP to Verify Authentication Protocols*, in: *Theorem Proving in Higher Order Logics: TPHOL's 97*, Lecture Notes in Computer Science **1275** (1997), pp. 121–136.

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

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

[16] Fernández Gago, M., U. Hustadt, C. Dixon, M. Fisher and B. Konev, *First-Order Verification in Practice*, Journal of Automated Reasoning **34(3)** (2005), pp. 295–321.

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

[18] Gabbay, D., A. Kurucz, F. Wolter and M. Zakharyaschev., "Many-Dimensional Modal Logics: Theory and Applications," Studies in Logic and the Foundations of Mathematics **148**, Elsevier, 2003.

[19] Gabbay, D., 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*, Las Vegas, Nevada, 1980, pp. 163–173.

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

[21] Gong, L., R. Needham and R. Yahalom, *Reasoning about Belief in Cryptographic Protocols*, in: *Proceedings of the IEE Computer Society Symposium on Research in Security and Privacy* (1990), pp. 234–248.

[22] Halpern, J. and L. Zuck, *A Little Knowledge Goes a Long Way: Simple Knowledge-Based Derivations and Correctness Proofs for a Family of Protocols*, in: *Proceedings of the 6th ACM Symposium on Principles of Distributed Computing*, 1987, pp. 268–280.

[23] Halpern, J. Y., R. van der Meyden and M. Vardi, *Complete Axiomatizations for Reasoning About Knowledge and Time*, SIAM Journal on Computing **33(3)** (2004), pp. 674–703.

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

[25] Hodkinson, I., F. Wolter and M. Zakharyaschev, *Decidable Fragments of First-Order Temporal Logics*, Annals of Pure Applied Logic **106(1-3)** (2000), pp. 85–134.

[26] Holzmann, G., "Design and Validation of Computer Protocols," Prentice-Hall, Englewood Cliffs, New Jersey, 1991.

[27] Hustadt, U., B. Konev, A. Riazanov and A. Voronkov, ***TeMP**: A Temporal Monodic Prover*, in: D. A. Basin and M. Rusinowitch, editors, *Proceedings of the Second International Joint Conference on Automated Reasoning (IJCAR)*, LNAI **3097** (2004), pp. 326–330.

[28] Konev, B., A. Degtyarev, C. Dixon, M. Fisher and U. Hustadt, *Towards the Implementation of First-Order Temporal Resolution: the Expanding Domain Case*, in: *Proceedings of TIME-ICTL* (2003), pp. 72–82.

[29] Konev, B., A. Degtyarev, C. Dixon, M. Fisher and U. Hustadt, *Mechanising First-Order Temporal Resolution*, Information and Computation **199(1-2)** (2005), pp. 55–86.

[30] Kraus, S. and D. Lehmann, *Knowledge, Belief and Time*, Theoretical Computer Science **58** (1988), pp. 155–174.

[31] Kupferman, O. and M. Vardi, *Synthesis with Incomplete Information*, in: *Advances in Temporal Logic*, Applied Logic Series **16** (2000), pp. 109–127, proceedings the Second International Conference on Temporal Logic (ICTL). ISBN 0-7923-6149-0.

[32] Lichtenstein, O., A. Pnueli and L. Zuck, *The Glory of the Past*, in: R. Parikh, editor, *Logics of Programs* (*Proc. Conf. Brooklyn USA 1985*), Lecture Notes in Computer Science **193**, Springer, Berlin, 1985 pp. 196–218.

[33] Lowe, G., *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*, Lecture Notes in Computer Science **1055** (1996), pp. 147–166.

[34] Manna, Z. and A. Pnueli, "The Temporal Logic of Reactive and Concurrent Systems: Specification," Springer, New York, 1992.

[35] Meadows, C., *The NRL Protocol Analyzer: An Overview.*, Logic Programming **26(2)** (1996), pp. 113–131.

[36] Meadows, C., *Formal Methods for Cryptographic Protocol Analysis: Emerging Issues and Trends*, IEEE Journal on Selected Areas in Communication **21(1)** (2003), pp. 44–54.

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

[38] Mitchell, J., M. Mitchell and U. Stern, *Automated Analysis of Cryptographic Protocols using Murϕ*, in: *Proceedings of the 1997 IEEE Symposium on Security and Privacy* (1997), pp. 141–151.

[39] Nalon, C., C. Dixon and M. Fisher, *Resolution for Synchrony and No Learning*, in: R. Schmidt, I. Pratt-Hartmann, M. Reynolds and H. Wansing, editors, *AiML-2004: Advances in Modal Logic*, Advances in Modal Logic Series **5** (2005), pp. 231–248.

[40] Needham, R. and M. Schroeder, *Using Encryption for Authentication in Large Networks of Computers*, Communications of the ACM **21** (1978), pp. 993–999.

[41] Paulson, L. C., *The Inductive Approach to Verifying Cryptographic Protocols*, Journal of Computer Security **6(1-2)** (1998), pp. 85–128.

[42] Pnueli, A. and R. Rosner, *On the Synthesis of a Reactive Module*, in: *Proceedings of the 16th ACM Symposium on the Principles of Programming Languages*, 1989, pp. 179–190.

[43] Rao, A. S., *Decision Procedures for Propositional Linear-Time Belief-Desire-Intention Logics*, in: M. Wooldridge, K. Fischer, P. Gmytrasiewicz, N. R. Jennings, J. P. Müller and M. Tambe, editors, *IJCAI-95 Workshop on Agent Theories, Architectures, and Languages*, Montréal, Canada, 1995, pp. 102–118.

[44] Rao, A. S. and M. P. Georgeff, *Decision procedures for BDI logics*, Journal of Logic and Computation **8(3)** (1998), pp. 293–342.

[45] Ryan, P., S. Schneider, M. Goldsmith, G. Lowe and B. Roscoe, "Modelling and Analysis of Security Protocols," Addison Wesley, 2001.

[46] Schmidt, R. A. and U. Hustadt, *A Principle for Incorporating Axioms into the First-Order Translation of Modal Formulae*, in: *Automated Deduction—CADE-19*, Lecture Notes in Artificial Intelligence **2741** (2003), pp. 412–426.

[47] Schneider, S., *Verifying Authentication Potocols with CSP*, in: *PCSFW: Proceedings of The 10th Computer Security Foundations Workshop* (1997), pp. 3–17.

[48] Stulp, F. and R. Verbrugge, *A Knowledge-Based Algorithm for the Internet Transmission Control Protocol (TCP)*, Bulletin of Economic Research **54(1)** (2002), pp. 69–94.

[49] Syverson, P., *Adding Time to a Logic of Authentication*, in: *Proceedings of the First ACM Conference on Computer and Communications Security* (1993), pp. 97–101.

[50] Syverson, P. and P. van Oorschot, *On Unifying some Cryptographic Protocols*, in: *Proceedings of the IEEE Computer Society Symposium on Research in Security and Privacy* (1994), pp. 14–28.

[51] van der Meyden, R., *Axioms for Knowledge and Time in Distributed Systems with Perfect Recall*, in: *Proceedings of the Ninth IEEE Symposium on Logic in Computer Science*, 1994, pp. 448–457.

[52] Wolter, F. and M. Zakharyaschev, *Temporalizing Description Logics*, in: *Frontiers of Combining Systems* (1999), pp. 379–402.

[53] Wolter, F. and M. Zakharyaschev, *Axiomatizing the Monodic Fragment of First-Order Temporal Logic*, Annals of Pure Applied Logic **118(1-2)** (2002), pp. 133–145.

[54] Wooldridge, M., C. Dixon and M. Fisher, *A Tableau-Based Proof Method for Temporal Logics of Knowledge and Belief*, Journal of Applied Non-Classical Logics **8(3)** (1998), pp. 225–258.