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, UK

{clare,mcarmen,michael,wiebe}@csc.liv.ac.uk
http://www.csc.liv.ac.uk/~{clare,mcarmen,michael,wiebe}

Overview

- With the growth of the Internet, wireless computing and ubiquitous computing there is an increasing need to interchange sensitive data.
- Cryptographic protocols are commonly used to distribute keys and authenticate agents and data over hostile networks.
- Verifying that particular properties do hold for these protocols (for example an intruder cannot gain sensitive information under certain assumptions) is critical.
- We investigate the specification and verification of security protocols using temporal logics of knowledge.

The Needham-Schroeder Protocol

Here is a simplified version of the Needham-Schroeder Protocol (NSP) consisting of three messages between two agents A and B.

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	$A \rightarrow B:$	$\{N_B\}_{pub\_key(B)}$

Messages of the form $\{X, Y\}_{pub\_key(Z)}$ represent messages containing both X and Y encrypted with Z's public key.

Elements of the form N_X are called *nonces*.

Typical Assumptions

Some of the assumptions made are:

- initially, only agent A knows the value of its own nonce N_A and only B knows the value of its own nonce N_B;
- if agent A or B sends messages containing A or B's nonce then they are encrypted with A or B'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;
- agents' knowledge of messages persists;
- if a message is received, and the receiver knows the private key required, then the receiver will know the content of the message.

BAN Logics/Related Work

- The application of logical tools to the analysis of security protocols was pioneered by Burrows, Abadi and Needham.
- 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".
- Other approaches have included the use of model checkers and the use of interactive theorem provers.

Some Limitations of BAN

- BAN had no formal semantics. A modal logic semantics was given to a BAN-like logic by Abadi and Tuttle.
- Using BAN logics the first step is protocol *idealisation*, i.e. transforming the informal description of the protocol to one using BAN primitives. How to do this is *not* easy.
- BAN originally had no explicit temporal component so can't express statements like *if a message is received then it was sent at some moment in the past*.
- Other extensions have added syntax to allow statements relating to the possession of keys etc.
- Its hard to deal with negated information, i.e. absence of information.

Using Temporal Logics of Knowledge

- We would like to specify and verify security protocols using temporal logics of knowledge ($KL_{(n)}$).
- We primarily use knowledge rather than belief. The main difference is for the former $K_i\varphi \Rightarrow \varphi$ is an axiom whereas $B_i\varphi \Rightarrow \varphi$ is not an axiom for the latter.
- We incorporate time explicitly to deal with the ordering of events.
- There are formal semantics for the logic.
- There is a bank of work on axiomatisations, complexity results etc.
- Verification can be carried out via proof within the logic.

Syntax of $KL_{(n)}$

The formulae of $KL_{(n)}$ (the fusion of PTL and S5) are constructed using:-

- a set $\mathcal{P}$ of proposition symbols and the constants **false** and **true**;
- the connectives $\neg$, $\vee$, $\wedge$, $\Rightarrow$, $\bigcirc$, $\Diamond$, $\Box$, $\mathcal{U}$, $\mathcal{W}$ and K_i (where $i \in Ag$).

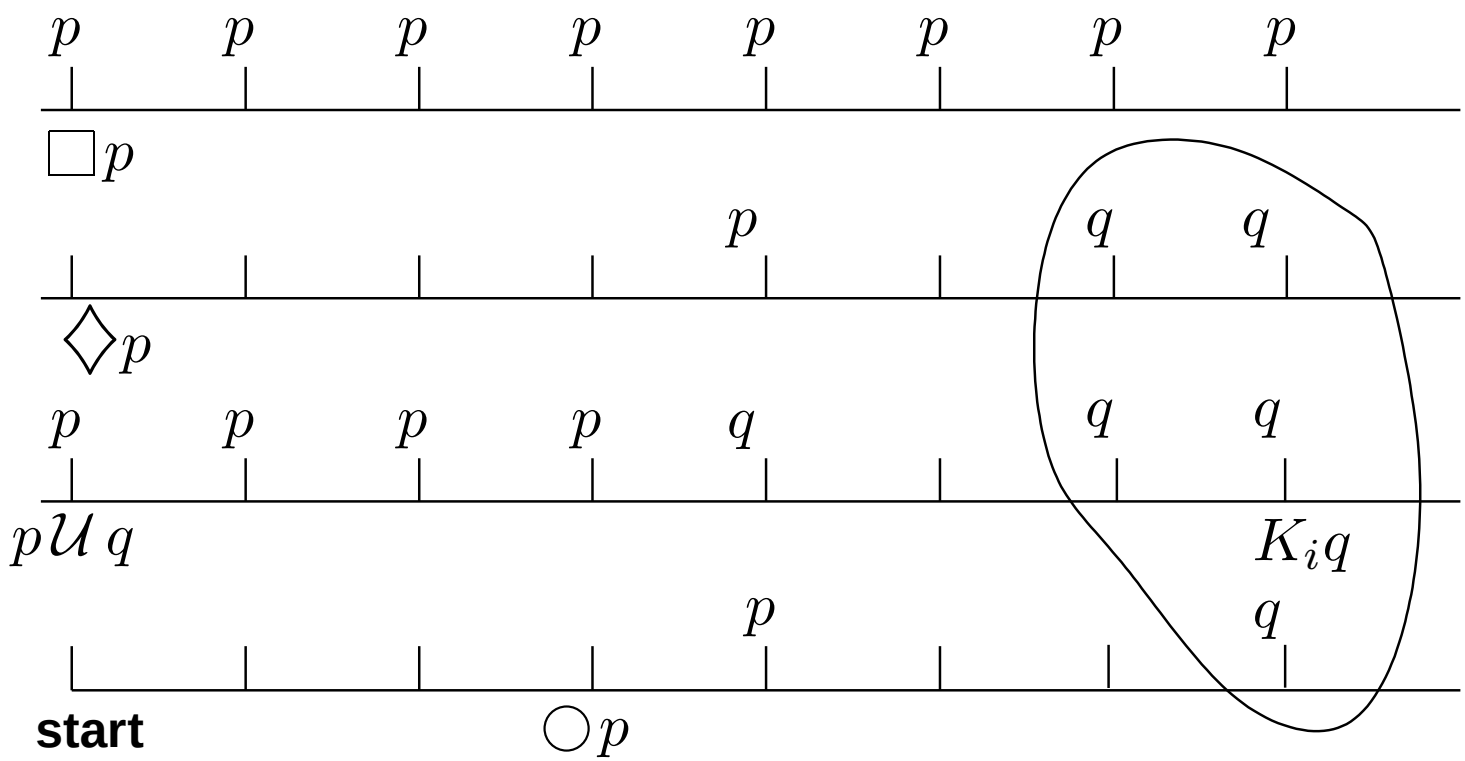
Well-formed formulae of $KL_{(n)}$ (WFF_K) are:-

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

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

Semantics of $KL_{(n)}$

A *model*, $\mathcal{M}$, for $KL_{(n)}$ is a structure $\mathcal{M} = \langle TL, R_1, R_2, \ldots, \pi \rangle$.



where

$$p \, \mathcal{W} \, q \equiv p \, \mathcal{U} \, q \vee \Box p$$

Specifying NSP in $KL_{(n)}$ (1)

To simplify the description, we allow quantification and equality over finite sets thus, the logic remains propositional. We identify the following predicates:

- $send(X, M, K)$ means agent X sends message M encrypted by key K;
- $rcv(X, M, K)$ means agent X receives message M encrypted by key K;
- $Msg(M)$ means M is a message;
- $val\_pub\_key(X, V)$ ($val\_priv\_key(X, V)$) means the value public (private) key of X is V;
- $val\_nonce(N_X, V)$ means the value of nonce N_X is V
- $contains(M_1, M_2)$ means M_2 is contained within M_1.

Specifying NSP in $KL_{(n)}$ (2)

We specify the protocol using axioms. For example the following states that *if an agent X knows the value of the private key of Y is V then in the next moment X knows this.*

$$\forall X, Y, V.\ K_X priv\_key(Y, V) \Rightarrow \bigcirc K_X priv\_key(Y, V)$$

Also *if an agent receives a message, then there was some agent that previously sent that message*.

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

Initial conditions are also specified such as *initially agent A knows the value of his private key is a_v* which can be specified as

$$K_A priv\_key(A, a_v)$$

Verifying Properties of the NSP in $KL_{(n)}$

- We have proved several properties of NSP, eg *once B receives the nonce of A encoded by B's public key then B knows the nonce of A*.

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

- Also *once A receives its nonce back encoded via A's public key then A knows that B knows the nonce of A*.

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

- Such properties have been proved by hand using clausal resolution for $KL_{(n)}$. Currently we are trying to use TeMP to do such proofs automatically.

Resolution for $KL_{(n)}$

Given any $KL_{(n)}$ formula ψ to be shown valid, negate giving $\neg\psi$ and perform the following steps.

1. Translate $\neg\psi$ into a clausal normal form.
2. Perform step, modal and temporal resolution until false has been derived or no new resolvents are generated.

$$\begin{array}{rcl} P & \Rightarrow & \bigcirc(R \vee l) \\ Q & \Rightarrow & \bigcirc(S \vee \neg l) \\ \hline (P \wedge Q) & \Rightarrow & \bigcirc(R \vee S) \end{array}$$

3. If false has been derived ψ is valid otherwise ψ is not valid.

The method is sound, complete and terminating.

Lowes Attack on the NSP

Lowe considers a situation where A runs the protocol with an enemy C, then C pretends he is A and starts a new run of the protocol with B.

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

Here C gets to know the nonces of both A and B.

Conclusions

- We have used $KL_{(n)}$ to specify the NSP.
- We have verified various properties of the specification using clausal resolution.
- We obtain the good properties of the logic for free.
- Given the (informal) description of the protocol it is non-trivial to specify it in $KL_{(n)}$.
- There are many axioms and the number of clauses generated is large.

Issues/Future Work

- To provide a framework which allows the translation of informally described protocols into $KL_{(n)}$. For example defining axioms relating to the environment and those relating to the particular protocol in question.
- Is $KL_{(n)}$ suitable for specifying other protocols found in the literature or do we need to represent some other aspect?
- Can we also use temporal resolution successfully to prove properties (or find flaws) in other protocols?
- How can we use the output from failed proofs to animate possible attacks?

References

[1] Dixon, C., Fernandez Gago, M.C., Fisher, M., and van der Hoek, W., Using Temporal Logics of Knowledge in the Formal Verification of Security Protocols in the Proceedings of TIME 2004, 1st-3rd July 2004, Tatihou, Normandie, France. IEEE.

[2] Dixon, C., Fernandez Gago, M.C., Fisher, M., and van der Hoek, W., Using Temporal Logics of Knowledge in the Formal Verification of Security Protocols, University of Liverpool Technical Report, ULCS-03-022, available at http://www.csc.liv.ac.uk/research/techreports/