C. Dixon, C. Fernandez-Gago, M. Fisher, and W. van der Hoek, "Using Temporal Logics of Knowledge in the Formal Verification of Security Protocols", *11th International Symposium on Temporal Representation and Reasoning (TIME04)*, pp. 148-151, 2004.
http://doi.org/http://doi.ieeecomputersociety.org/10.1109/TIME.2004.1314432
NICS Lab. Publications: https://www.nics.uma.es/publications

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
The University of Liverpool, Liverpool L69 7ZF, UK
{clare,mcarmen,michael,wiebe}@csc.liv.ac.uk

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 use temporal logics of knowledge to reason about security protocols. We show how to specify part of the Needham-Schroeder protocol using temporal logics of knowledge and prove various properties using a clausal 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. Consequently, cryptographic protocols are commonly used to distribute keys and authenticate agents and data over hostile networks. Although the protocols used often appear watertight, 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 [7] specific epistemic logics, collectively referred to as BAN logics, were proposed to deal with authentication issues. We propose an approach using a combination of temporal and epistemic logics.

By combining both 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. These logics have the advantages of a well-defined semantics, an existing body of theoretical work relating to, for example, axiomatisations and complexity, see for example [9], and sound and complete proof methods for example [2].

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 [12]. This protocol has been widely studied with particular problems uncovered via formal analysis, for example [11]. Our aim is to demonstrate the suitability of $KL_{(n)}$, with its resolution method, for security and authentication, rather than bringing new insights to the Needham Schroeder protocol.

Note that, due to lack of space, we will neither give a full description of the resolution calculus, nor full details of the Needham-Schroeder proofs. These details can be found in a companion technical report [4].

2. The Needham-Schroeder Protocol (NSP)

The Needham-Schroeder protocol (NSP) with public keys [12] intends to establish authentication between an agent A who initiates the protocol and an agent B who responds to A. The complete protocol consists of seven messages, but we here 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. The protocol can then be described as the 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	$A \rightarrow B:$	$\{N_B\}_{pub\_key(B)}$

Here $X \rightarrow Y$ denotes that agent X sends agent Y a message. 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 initially unknown to all other agents.

3. Temporal Logic of Knowledge

The logic, $KL_{(n)}$, a *temporal logic of knowledge* is the fusion of propositional linear-time temporal logic with multi-modal S5. The temporal component is interpreted over a discrete linear model of time with finite past and infinite future and the each modal relation is an equivalence class. This logic has been studied in detail [9] and is the most commonly used temporal logic of knowledge. We use the usual set of operators including $\bigcirc$ (*next*), $\diamondsuit$ (*sometime* or *eventually*), $\square$ (*always*), K_i for *knowledge* and allow an operator **start** to denote the initial moment in time. For details of the syntax and semantics of $KL_{(n)}$ see for example [2].

To prove properties of our specification we use a resolution calculus for $KL_{(n)}$. Due to lack of space we omit the details of the proof method but refer the interested reader to [2, 3].

4. Specifying the NSP in $KL_{(n)}$

In this section, we will use $KL_{(n)}$ to specify the NSP. 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 and we introduce an agent C to represent a potential intruder. We identify the following predicates:

- $send(X, Msg, Key)$ (respectively $rcv(X, Msg, Key)$) is satisfied if agent X sends (respectively receives) message Msg encrypted by Key;
- $Msg(M_1)$ is satisfied if M_1 is a message;
- $val\_pub\_key(X,V)$ (respectively $val\_priv\_key(X,V)$) is satisfied if the value of the public (respectively private) key of X is V
- $val\_nonce(N_X, V)$ is satisfied if the value of nonce N_X 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 finite sets of agents, messages and keys; thus, this logic remains essentially propositional.

Specifying Structural Assumptions We begin with various structural assumptions concerning keys and message contents. These are given in Figure 1.

1. $\forall X, Key, M_1.\ send(X, M_1, Key) \Rightarrow \neg contains(M_1, priv\_key(X))$
   — agents will not reveal their private key to others
2. $\forall X, V_1, V_2, V_3.$
   $[val\_pub\_key(X,V_1) \Leftrightarrow \square val\_pub\_key(X,V_1)] \wedge$
   $[val\_priv\_key(X,V_2) \Leftrightarrow \square val\_priv\_key(X,V_2)] \wedge$
   $[val\_nonce(X,V_3) \Leftrightarrow \square val\_nonce(X,V_3)]$
   — the public keys, private keys and nonces of all the agents remain the same during the protocol
3. $\forall X, Y, V.\ (val\_pub\_key(X,V) \wedge val\_pub\_key(Y,V)) \Rightarrow X = Y)$
   — no two agents have the same public keys
4. $\forall Key, M_1.(send(A, M_1, Key) \wedge [contains(M_1, N_A) \vee contains(M_1, N_B)]) \Rightarrow (Key = pub\_key(B))$
   — if agent A sends out messages containing N_A or N_B they must be encrypted with B's public key.
5. $\forall Key, M_2.\ (send(B, M_2, Key) \wedge [contains(M_2, N_A) \vee contains(M_2, N_B)]) \Rightarrow (Key = pub\_key(A))$
   — if agent B sends out messages containing N_A or N_B they must be encrypted with A's public key.

Figure 1. Specifying Structural Assumptions.

Specifying Scenario Assumptions In Figure 2 we instantiate message contents, keys and names for this particular scenario.

Specifying Basic Knowledge Axioms In Figure 3 we specify the attributes of an agent's knowledge.

Specifying Communication Axioms We now specify communication between agents, and how this affects the agent's knowledge. For convenience, we use past-time temporal operators, in particular "$\circledcirc$", meaning in the previous moment in time, and "$\blacklozenge$", meaning at some time in the past. These operators have the usual semantics (see for example [4]). This is shown in Figure 4.

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

Figure 4. Specifying Communication Axioms.

6. $\forall M_1$
$Msg(M_1) \Leftrightarrow ((M_1 = m_1) \vee (M_1 = m_2) \vee (M_1 = m_3))$
— in this particular scenario, we just use three messages, m_1,m_2 and m_3. Other (dummy) messages can be added to make this axiom more realistic.

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** $\Rightarrow$
$val\_priv\_key(A, a_v) \wedge val\_priv\_key(B, b_v)\wedge$
$val\_priv\_key(C, c_v) \wedge val\_pub\_key(A, a)\wedge$
$val\_pub\_key(B, b) \wedge val\_pub\_key(C, c))\wedge$
$val\_nonce(N_A, a_n) \wedge val\_nonce(N_B, b_n)\wedge$
$val\_nonce(N_C, c_n)$
— the initial values of public and private keys and nonces (we also have negated statements eg **start** $\Rightarrow \neg val\_priv\_key(A, b_v)$ etc).

Figure 2. Specifying Scenario Assumptions.

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. $\forall X, Y.\ (\exists V. K_X val\_priv\_key(Y, V) \Leftrightarrow (X = Y))$
— 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

Figure 3. Specifying Knowledge Axioms.

5. Verifying Properties of the Specification

Once we have the above axioms relating to the specific scenario, we can attempt to prove various statements. Proof is by clausal resolution. For more details see [4].

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

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

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.

$$\forall V.\ \square \neg K_C val\_nonce(N_A, V)$$

Confirmation of B**'s Knowledge** Once A receives m_2 (which, in turn, contains N_A) back, then it can infer that B knows the value of N_A, i.e.

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

6. Related Work and Conclusions

BAN logics such [1] and [7] analyse security protocols by reasoning about the beliefs of principals. In our approach we use a well studied non-classical logic, i.e. the temporal logic of knowledge to specify and verify protocols. We initially choose to reason about knowledge rather than belief as the epistemic logic of knowledge (S5) is stronger than that of belief (KD45) requiring the axiom $Kl \Rightarrow l$ i.e. if an agent knows l then l is true. We could also easily incorporate beliefs to capture situations when a principal believed items that may not be true.

Further, we use temporal operators to capture temporal information, relating to the order of events for example sending and receiving messages, that is not explicitly stated in BAN logics. In [14], an extension of BAN, time is incorporated in by allowing the past time temporal operators *always in the past* and its dual *sometime in the past* to impose some order on events. We allow a much richer temporal language. BAN logics implicitly assume that beliefs cannot decrease over time. Here we must explicitly state what knowledge persists.

In an approach similar to ours, in [6] a simple branching time logic allowing limited combinations of temporal operators is combined with modal logics of knowledge and permission/obligation to specify security protocols. However no proof method is provided for the resulting logic.

In [8] the authors use Lamport's Raw Temporal Logic of Actions (RTLA) [10] to specify and verify security protocols. In RTLA an action is a statement about pairs of states. Axioms, for example relating to sending and receiving messages, are written with respect to the relevant changes in state. The language allows connectives from classical logic as well as the temporal connectives $\square$ and $\diamondsuit$ and other constructs.

Another approach that does use theorem proving, though not particularly related to our approach, is presented in [5]. This paper shows how to introduce time into the Communicating Sequential Processes (CSP) protocol verification framework of [13]. Then CSP is embedded in the PVS (Prototype Verification System).

In this paper, we have shown how temporal logics of knowledge are useful for specifying complex aspects of security protocols. One of the advantages of using a standard combination of temporal and modal logics is that there is a clear semantics for this logic. This was a problem with the early BAN logics. Secondly there is an existing bank of work relating to axiomatisations, complexity, proof methods etc that can be applied. In combination with clausal resolution techniques we have developed, this allows us to carry out verification of properties of security protocols. While there has been work on verification of such protocols before, the clarity of the logic, together with the flexibility of the proof technique, makes this work important. In the future, we will consider adding first-order aspects to the logic, thus allowing the verification of infinite state protocols.

Acknowledgements This work was partially supported by EPSRC research grant GR/R45376/01.

References

[1] M. Burrows, M. Abadi, and R. Needham. A Logic for Authentication. In *Proceedings of the Royal Society of London*, volume 426, pages 233–271, 1989.

[2] 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.

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

[4] C. Dixon, M.-C. F. Gago, M. 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`.

[5] N. Evans and S. Schneider. Analysing time dependent security properties in CSP using PVS. In *ESORICS*, pages 222–237, 2000.

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

[7] L. Gong, R. Needham, and R. Yahalom. Reasoning about Belief in Cryptographic Protocols. In *Proceedings of the IEEE Computer Societey Symposium on Research in Security and Privacy*, pages 234–248. IEEE Computer Society Press, 1990.

[8] J. W. Gray and J. McLean. Using Temporal Logic to Specify and Verify Cryptographic Protocols. *Computer Security Foundations Workshop*, 12:108–116, 1995.

[9] 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.

[10] L. Lamport. The Temporal Logic of Actions. Technical Report Research Report 79, DEC Systems Research Center, Palo Alto, CA, 1991.

[11] 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.

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

[13] S. Schneider. Verifying authentication protocols with CSP. In *PCSFW: Proceedings of The 10th Computer Security Foundations Workshop*. IEEE Computer Society Press, 1997.

[14] 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.