This paper was selected by a process of
anonymous peer reviewing for presentation at

COMMONSENSE 2007

8th International Symposium on Logical Formalizations of Commonsense Reasoning

Part of the AAAI Spring Symposium Series, March 26-28 2007,
Stanford University, California

Further information, including follow-up notes for some of the
selected papers, can be found at:

www.ucl.ac.uk/commonsense07

An Explicit Model of Belief Change for Cryptographic Protocol Verification

Aaron Hunter and **James P. Delgrande**
School of Computing Science
Simon Fraser University
Burnaby, BC, Canada
{amhunter, jim}@cs.sfu.ca

Abstract

Cryptographic protocols are structured sequences of messages that are used for exchanging information in a hostile environment. Many protocols have epistemic goals: a successful run of the protocol is intended to cause a participant to hold certain beliefs. As such, epistemic logics have been employed for the verification of cryptographic protocols. Although this approach to verification is explicitly concerned with changing beliefs, formal belief change operators have not been incorporated in previous work. In this preliminary paper, we introduce a new approach to protocol verification by combining a monotonic logic with a non-monotonic belief change operator. In this context, a protocol participant is able to retract beliefs in response to new information and a protocol participant is able to postulate the most plausible event explaining new information. Hence, protocol participants may draw conclusions from received messages in the same manner conclusions are drawn in formalizations of commonsense reasoning. We illustrate that this kind of reasoning is particularly important when protocol participants have incorrect beliefs.

Introduction

Logics of belief have been useful in the design and analysis of cryptographic protocols, starting with the pioneering work on BAN logic (Burrows, Abadi, & Needham 1989) and continuing with several related logics (Adadi & Tuttle 1991; Agray, van der Hoek, & de Vink 2002; Bleeker & Meertens 1997; Syverson & van Oorschot 1996; Syverson & Cervesato 2001). The basic idea behind all of the BAN-like logics is to encode protocols in a logic of belief, then prove that the agents involved must have certain beliefs after a protocol is properly executed. Hence, protocol verification is fundamentally concerned with modelling changing beliefs. However, BAN-like logics do not involve any explicit formal model of belief change; the manner in which beliefs change is simply captured by some ad hoc axioms. In this paper, we argue that a more appropriate approach to protocol verification can be defined by describing beliefs in a static logical framework together with a formal belief change operator.

This is an exploratory paper intended to illustrate how formal approaches to belief change may be useful in reasoning about cryptographic protocols. The paper makes two main contributions to existing research. First, we extend the application of formal belief change techniques to a new class of problems. Cryptographic protocol verification provides a large class of belief change problems which are not only of theoretical interest, but also have great practical significance. The second contribution is the introduction of a specific model of belief change that is particularly suitable for the the verification of cryptographic protocols. Broadly speaking, researchers in belief change and researchers in protocol verification are both interested in the same kinds of problems. Our aim is to make this salient, and to illustrate how work in each area can benefit the other.

We proceed as follows. First, we introduce some preliminary background on the logical approach to cryptographic protocol verification. Next, we argue that BAN-like logics can not capture the kind of non-monotonic belief change that occurs in an authentication protocol, and we introduce some more appropriate belief change operators. Finally, we present a simple approach to protocol verification in terms of formal belief change operators. We conclude with a general discussion about belief change in the context of protocol verification.

Preliminaries

Authentication Protocols

For the purposes of this paper, we focus on the verification of authentication protocols. An authentication protocol is used to ensure that all parties in a communication session know with whom they are communicating. Authentication protocols may have additional goals as well, such as establishing a shared key for communication (Syverson & Cervesato 2001). In this section, we briefly introduce some standard notation for describing authentication protocols.

If P and Q denote agents in a communication session, then

$$P \rightarrow Q : X$$

means that agent P sends the message X to agent Q. A symmetric key that is shared by agents P and Q will be denoted by K_{pq}. If the message X is encrypted with the key K, then we write $\{X\}_K$. We let N_p denote a nonce generated by the agent P. A nonce is simply a random number that is generated by an agent during a communication session.

The following simple protocol is executed by agent P in order to determine if agent Q is alive in the communication session.

The Challenge-Response Protocol
1. $P \rightarrow Q : \{N_p\}_{K_{pq}}$
2. $Q \rightarrow P : N_p$

In this protocol, P generates a random number and encrypts it with the shared key K_{pq} before sending it to Q. Informally, if this message is intercepted by an intruder, it will not be possible for the intruder to determine the value N_p. So if P receives the message N_p, then it is natural to conclude that Q must have decrypted the original message and therefore Q is alive on the network. In the vocabulary of (Guttman & Thayer 2000), this protocol involves a single *outgoing authentication test*.

The standard model for cryptographic protocol analysis assumes that an intruder can read every message that is sent, and the intruder may choose to prevent messages from reaching the desired recipient. Moreover, when a message is received, it is assumed that the sender is always unknown. The only way that P can be certain Q sent a particular message is if the message contains information that is only available to Q. It is assumed that cryptography is strong in that encrypted messages can not be unencrypted during a protocol run without the proper key. Under these assumptions, there is an attack on the Challenge-Response protocol.

An Attack on the Challenge-Response Protocol
1. $P \rightarrow I_Q : \{N_p\}_{K_{pq}}$
   $1'$. $I_Q \rightarrow P : \{N_p\}_{K_{pq}}$
   $2'$. $P \rightarrow I_Q : N_p$
2. $I_Q \rightarrow P : N_p$

In this attack, I_Q intercepts the original message and then initiates a new protocol run by sending it back to P. After P receives the message encrypted with K_{pq}, then P follows the protocol and returns the decrypted nonce. At the last step, I_Q sends the same decrypted nonce to P. Note that, at the conclusion of the protocol, Q has not sent any messages. Hence P has no assurance that Q is actually alive on the network, which was the stated goal of the protocol.

Logics of Belief

As noted above, the first logical approach to protocol verification was the so-called BAN logic. We refer the reader to (Burrows, Abadi, & Needham 1990) for a formal introduction to the logic. We very briefly sketch the basic idea.

Sentences of BAN logic are generated through constructions of the following form.

- P ***believes*** X : P thinks that X is true
- P ***received*** X : a message containing X was received by P
- P ***said*** X : a message containing X was sent by P previously
- $P \xleftrightarrow{K} Q$: K is a good key for communication between P and Q

This list is not exhaustive, but simply illustrates the flavour of BAN constructions. The semantics of these constructions is given through a collection of rules of inference. For example, the following rule of inference is defined for BAN logic:

$$\frac{P \textit{ believes } P \xleftrightarrow{K} Q \qquad P \textit{ received } \{X\}_K}{P \textit{ believes } Q \textit{ said } X}.$$

This rule is called Message Meaning, and it attempts to capture the manner in which beliefs change during protocol execution. The rule states that, if P receives a message encrypted in a key that P shares with Q, then P should conclude that Q sent the message.

From a logical point of view, one well-known problem with BAN is the fact that there is no agreed upon semantics(Agray, van der Hoek, & de Vink 2002). However, several different semantics have been proposed (Adadi & Tuttle 1991; Bleeker & Meertens 1997; Syverson & Cervesato 2001), and new protocol logics have been introduced based on the standard semantics for epistemic logic (Syverson & van Oorschot 1996). We remark that such logics typically address belief change by introducing some ad hoc axioms or rules of inference.

It is worth noting that BAN logic is not able to establish the insecurity of the Challenge-Response Protocol. In BAN logic, it is simply assumed that an agent can recognize the messages that they have sent. Under this assumption, the given attack can not occur. Not only is this assumption ad hoc, but it is often unjustified in real applications.

Non-Monotonic Belief Change

Belief Update and Belief Revision

In practice, a protocol like the Challenge-Response Protocol will not be run in isolation. The agent P will receive information from many sources, each with differing degrees of reliability. As such, it is possible that P will have beliefs that are incorrect. Therefore, P needs to be able to retract beliefs and modify the current belief state in response to new information. The monotonic models of belief change in BAN-like logics are not suitable for this kind of reasoning. Instead, we suggest that we need to treat belief change in cryptographic protocols in terms of non-monotonic *belief change operators*. Informally, the goal of an intruder is to convince a protocal participant to hold certain beliefs. As such, we can view protocol verification as a problem in commonsense reasoning. A protocol participant is likely to draw many (possibly non-monotonic) conclusions when a message is received. We need a more realistic model of belief change to capture the reasoning of a protocol participant.

We will represent action effects by transition systems, as defined in (Gelfond & Lifschitz 1998). We introduce some standard notation. Assume an underlying set of atomic propositional formulas $\mathbf{F}$ and a set $\mathbf{A}$ of *action names*. A *state* is an interpretation over $\mathbf{F}$, and a *transition system* is a directed graph where each node is labeled by a state and each edge is labeled by a set of action symbols. Informally, an edge from s to s' labelled with A is interpreted to mean that executing the action A in the state s results in the state

s'. A *belief state* is a set of states, informally the set of states that an agent considers to be possible.

The belief change that occurs when an agent receives new information about some change in the world is called *belief update*. In belief update, we ask the following question: if an agent initially has belief state κ, then what should the new belief state be following the action A? We define a conditional belief update operator $\diamond$ to represent the belief change due to an action. More accurately, our operator is a *belief progression* operator; it has been illustrated elsewhere that update is a special case of belief progression (Lang 2006). If κ is a belief state and A is an action, then define

$$\kappa \diamond A = \{s' \mid (s, A, s') \in E \text{ and } s \models \kappa\}.$$

Hence, the semantics of belief update is based on projecting the initial belief set to accomodate the changed information.

The belief change that occurs when an agent receives new information about a static world is called *belief revision*. In the interest of space, we assume that the reader is familiar with the AGM approach to belief revision (Alchourrón, G ardenfors, & Makinson 1985). AGM revision is normally stated in terms of operators on sets of formulas, but it can easily be reformulated in terms of sets of states. In this context, the beliefs of an agent are represented by a set of states and the new information acquired is also represented by a set of states. We refer to a set of states representing new information as an *observation*. Let $2^{\mathbf{F}}$ denote the set of all states over $\mathbf{F}$. We say that a function $* : (2^{\mathbf{F}} \times \mathbf{F}) \rightarrow 2^{\mathbf{F}}$ is an AGM revision operator if it satisfies the AGM postulates suitably reformulated in terms of states. Such a function maps a belief state and an observation to a new belief state.

Belief Evolution

Protocol verification involves a combination of belief update and belief revision. When an agent sends a message, then the state of the world changes in a predictable manner. As such, sending a message causes an agent to perform belief update. On the other hand, received messages need not be the result of a change in the state of the world. An agent that receives a message may simply be receiving new information that must be incorporated in the current belief state. Hence, receiving messages may cause an agent to perform belief revision. Therefore, in order to reason about the iterated sequences of messages exchanged in a cryptographic protocol, one needs to reason about alternating sequences of revisions and updates.

There are plausible examples where it is clear that sequences of revisions and updates can not be interpreted by simply applying the operators iteratively; *belief evolution* operators have been proposed to combine an update operator and a revision operator (Hunter & Delgrande 2005). The problem is that many AGM belief revision operators are *Markovian*, in the sense that the new belief set is completely determined by the current belief set and the formula for revision. However, even simple protocols like the Challenge-Response Protocol require an agent to consider the history of messages sent in order to interpret incoming messages. As such, we need to use a non-Markovian belief change operator suitable for reasoning about iterated belief change. In this section, we briefly present a simplified version of belief evolution.

A belief evolution operator is defined with respect to a fixed update operator and a fixed revision operator. As such, assume that $*$ is an AGM revision operator (defined on sets of states), and let $\diamond$ be an update operator. The basic intuition behind belief evolution operators is that an agent should trace back new observations to conditions on the initial belief state. Let $\bar{A}$ denote a finite sequence of actions, and let α denote an observation. Define $\alpha^{-1}(\bar{A})$ to be the set of states w such that the sequence of actions $\bar{A}$ leads to a state in α. Define $\circ$ as follows:

$$\kappa \circ \langle \bar{A}, \alpha \rangle = \kappa * \alpha^{-1}(\bar{A}) \diamond \bar{A}.$$

Hence, belief evolution operators essentially revise the initial belief state before applying the effects of actions.

In the general case, belief evolution operators need to be defined for any alternating sequence of actions and observations. This can be done by successively passing each observation to a condition on the initial beliefs, then revising the initial beliefs by this sequence of observations. This process introduces the problem of *iterated revision*, which is known to be a difficult problem (see (Darwiche & Pearl 1997) for one representative approach). For the present purposes, we simply state that belief evolution operators generally take a sequence $\bar{A} = A_1, \ldots, A_n$ of actions and a sequence $\bar{\alpha} = \alpha_1, \ldots, \alpha_n$ of observations as arguments. Roughly speaking, the result of belief evolution corresponds to the belief change that should occur due to an alternating sequence of actions and observations:

$$\kappa \circ \langle \bar{A}, \bar{\alpha} \rangle \approx \kappa \cdot A_1 \cdot \alpha_1 \cdot \cdots \cdot A_n \cdot \alpha_n$$

We refer the reader to (Hunter & Delgrande 2005) for the details.

Belief Change in Protocol Verification

We can think of cryptographic protocols as message passing systems. In this section, we give specific revision and update operators that are suitable for reasoning about message passing systems. We remark that our intention is not to define a sophisticated protocol logic that could serve as an alternative to existing logics; instead, our goal is simply to illustrate how a formal approach to belief change can inform logical approaches to protocol verification.

Message Passing Systems

In order to reason about cryptographic protocols, we first need to introduce a propositional language for describing message passing systems. We assume a finite set $\mathbf{M}$ of *messages*, a finite set $\mathbf{K}$ of *keys*, and a finite set $\mathbf{P}$ of *participants*. Moreover, we assume that the set of keys contains a distinguished null key λ which will be used to represent unencrypted messages.

The set $\mathbf{F}$ of propositional symbols describing the state of the world is the following set:

$$\{HasKey(P, K) \mid P \in \mathbf{P}, K \in \mathbf{K}\}$$
$$\cup \{HasMessage(P, M, K) \mid P \in \mathbf{P}, M \in \mathbf{M}, K \in \mathbf{K}\}$$

The set $\mathbf{F}$ consists of all possible propositional statements asserting that a participant has a certain key or a certain encrypted message. Such statements are the only assertions that are possible in our message passing framework. The set $\mathbf{A}$ of actions is the following:

$$\{SendMessage(P, M, K) \mid P \in \mathbf{P}, M \in \mathbf{M}, K \in \mathbf{K}\}$$

Informally, $SendMessage(P, M, K)$ represents the action where agent P sends the message $\{M\}_K$. Note that no recipient is specified; this is intended to reflect the fact that every sent message can be intercepted. We can think of sent messages in terms of a white board system. Every time a message is sent, it is simply placed on a public posting board, and it can be viewed or erased by any participant. Hence, the effect of the action $SendMessage(P, M, K)$ is that it causes some agent other than P to have the message $\{M\}_K$.

Formally, the effects of the actions in $\mathbf{A}$ are given by a transition system (V, E). The set of vertices V consists of all propositional interpretations of $\mathbf{F}$ where $HasKey(P, \lambda)$ is true for each P. Intuitively, the edges in E should describe which messages are transferred between agents. For example, the act of sending a message M should be represented by including edges from each state s to every state s' that differs from s in that some other agent now has the message M. Formally, the set of edges E consists of all triples

$$(s, SendMessage(P_0, M_0, K_0), s')$$

where $s \models HasMessage(P_0, M_0, K_0)$ and s' satisfies the following conditions.

1. For all (P, K),
$$s' \models HasKey(P, K) \iff s \models HasKey(P, K)$$
2. There is some $Q_0 \neq P_0$ such that
$$s' \models HasMessage(Q_0, M_0, K_0)$$
and, if $s' \models HasKey(Q_0, K_0)$, then
$$s' \models HasMessage(Q_0, M_0, \lambda)$$
3. For all $(P, M, K) \notin \{(Q_0, M_0, K_0), (Q_0, M_0, \lambda)\}$,
$$s' \models HasMessage(P, M, K) \iff s \models HasMessage(P, M, K)$$

As stated previously, the action $SendMessage(P, M, K)$ causes some agent other than P to have the message $\{M\}_K$. In the third condition, we are actually making the simplifying assumption that exactly one other agent receives the message. In the whiteboard analogy, this is tantamount to assuming that agents must erase the whiteboard immediately after reading the contents. Note that, if the agent receiving $\{M\}_K$ happens to have the key K, then that agent also receives M. We remark that the sending agent need not have the key K; this allows messages to be redirected without being understood.

We illustrate with an example.

Example In the Challenge-Response protocol, we have the following messages, keys and partipants:

- $\mathbf{M} = \{N\}$
- $\mathbf{K} = \{K, \lambda\}$
- $\mathbf{P} = \{P, Q, I_Q\}$

We have omitted the subscripts on N_p and K_{pq}, since there is only one nonce and one non-null key. The atomic formulas in this domain include the following: $HasKey(P, K)$, $HasKey(P, \lambda)$, $HasMessage(P, N, K)$, and $HasMessage(P, N, \lambda)$. In each case P can be replaced with either Q or I_Q, giving a total of 12 atomic formulas.

The initial state in the Challenge-Response protocol is the state s satisfying exactly the following atomic formulas: $HasKey(P, \lambda)$, $HasKey(Q, \lambda)$, $HasKey(I_Q, \lambda)$, $HasKey(P, K)$, $HasKey(Q, K)$, $HasMessage(P, N, \lambda)$, $HasMessage(P, N, K)$. Hence, s represents the state where P is the only agent with the message N, and P, Q are the only agents that have K.

Define s_1 to be the interpretation satisfying the following conditions.

- $s_1 \models HasMessage(Q, N, K)$
- $s_1 \models HasMessage(Q, N, \lambda)$
- For all other atomic formulas ϕ,
$$s_1 \models \phi \iff s \models \phi$$

Similary, define s_2 to be the interpretation satisfying the following conditions.

- $s_2 \models HasMessage(I_Q, N, K)$
- For all other atomic formulas ϕ,
$$s_2 \models \phi \iff s \models \phi$$

Hence, s_1 represents the state that will result if Q receives the message $\{N\}_K$ and s_2 represents the state that will results if I_Q receives the message $\{N\}_K$. It is easy to verify that, $(s, SendMessage(P, N, K), s') \in E$ if and only if s' is one of s_1 or s_2.

Belief Change in Message Passing Systems

In the previous section, we presented a transition system framework for reasoning about the effects of actions in a message passing system. We also illustrated that we can describe the messages sent in a cryptographic protocol using our framework. However, as noted previously, many cryptographic protocols have epistemic goals. As such, before we can actually prove the correctness of a protocol, we need to address belief change in message passing systems. In a message passing system, sending messages causes an agent to update their beliefs, whereas receiving messages causes an agent to revise their beliefs. Hence, the belief change following a sequence of sent and received messages can be captured by a belief evolution operator. In this section, we illustrate how to define a belief evolution operator in our framework.

In order to define a belief evolution operator for message passing systems, we need a belief update operator and a belief revision operator. We already have a belief update operator defined with respect to the transition system giving

action effects. In order to proceed, we need to illustrate how to define a belief revision operator.

In general, AGM revision operators rely on an underlying similarity relation on states. In particular, given any fixed state s, we need to define a total pre-order $\preceq_s$ over all states. The minimal elements of $\preceq_s$ are intuitively the states that are the "most similar" to s.

In a message passing system, we can define the degree of similarity between two states s_1 and s_2 to be the number of actions that would need to be performed in order to get from s_1 to s_2. More specificially, let T be a transition system and let κ, α be sets of states. For technical reasons, we assume that every state in α is reachable from κ by a finite path in Φ. Define $\kappa * \alpha$ to be the set of elements of α that can be reached by a minimum length Φ-path. It is easy to prove that $*$ defines an AGM revision operator; we refer to this as the *topological revision operator* defined by T.

Let (V, E) be a transition system giving the effects of actions for a message passing system. Let $\diamond$ be the update operator defined by (V, E) and let $*$ be the corresponding topological revision operator. Let $\circ$ be the belief evolution operator defined with respect to $\diamond$ and $*$. In the next section, we illustrate that this operator provides a model of belief change that is useful for reasoning about protocol verification.

Verifying Authentication Protocols

A complete treatment of cryptographic protocols requires multiple agents with nested beliefs. In BAN logic, for example, the goals of a protocol typically involve beliefs about the beliefs of protocol participants. Dealing with nested beliefs is beyond the scope of this paper; reasoning about the revision of nested beliefs is a difficult problem on its own. However, it is possible to give a simple treatment of authentication tests in terms of belief evolution operators on the propositional beliefs of a single-agent.

As indicated in the preceding section, the set of actions that we consider consists of all possible message-sending actions. Messages received, on the other hand, are treated as observations. From the perspective of a single agent, cryptographic protocols generally have the following form, where each A_i is an action and each α_i is an observation.

Generic Protocol
1. A_1
2. α_1

$\vdots$

2n-1.A_n
2n. α_n

In protocol verification, we typically assume that the principle agent has some initial belief state κ, and we are interested in proving that some property holds after every protocol run. Suppose that the goal of a given protocol can be expressed as a propositional formula ϕ. Suppose that an agent executes the sequence of actions $A_1, \ldots, A_n$, interspersed with the observations $\alpha_1, \ldots, \alpha_n$. So the actions and observations of the agent together form the following sequence:

$$A_1, \alpha_1, \ldots, A_n \alpha_n.$$

The basic problem of protocol verification consists in answering the following question. If $A_1, \ldots, A_n$ is a subseqence of a larger sequence $ACT_1, \ldots, ACT_p$, does it necessarily follow that

$$\kappa \circ \langle ACT, OBS \rangle \models \phi?$$

Note that this approach to protocol verification does not require any ad hoc rules describing belief change, instead we have framed the problem as a simple application of belief evolution. We illustrate how this procedure can be applied in the case of the Challenge-Response protocol.

Example The Challenge-Response protocol consists of a single outgoing authentication test. The intuition behind an outgoing authentication test is that the interpretation of a received message is dependent upon the messages that have been sent previously. In particular, if an agent P receives a message M, then P should believe that the actual history of the world is one in which it is possible to receive the message X. In the challenge response protocol, when P receives the response N_p, it is not reasonable to conclude that Q decrypted $\{N_p\}_{K_{pq}}$. The strongest conclusion that P should draw is that either Q decrypted $\{N_p\}_{K_{pq}}$ or else P decrypted it unknowingly.

Let the initial belief state κ be the set of states s satisfying the following conditions.

1. for all $X \in \mathbf{P}$, $s \models HasKey(X, \lambda)$
2. for all $X \in \mathbf{P}$, $Y \in \mathbf{M}$, $s \models HasMessage(X, M, K) \rightarrow HasMessage(X, M, \lambda)$
3. $s \models HasKey(P, K)$
4. $s \models HasKey(Q, K)$
5. $s \models HasMessage(P, N, \lambda)$

Now suppose that the agent P initiates a run of the protocol by sending the message $\{N\}_K$ and the run eventually terminates when P receives the message N. Formally, the receipt of N is identified with the observation α defined as follows:

$$\alpha = HasMessage(Q, N, \lambda) \vee HasMessage(P, N, \lambda).$$

So α consists of all states where either P or Q has received the first message. The goal of the protocol is to establish that Q received the first message; hence, the goal of the protocol is to guarantee that P correctly believes $HasMessage(Q, N, \lambda)$ in the final state.

According to our approach, proving that the protocol is correct amounts to proving that, for any alternating sequence of the form

$$A_1, \alpha_1, \ldots, A_n, \alpha_n$$

containing the subsequence

$$SendMessage(P, N, K), HasMessage(P, N, \lambda),$$

it follows that $HasMessage(Q, N, \lambda)$ is true in

$$\kappa \circ \langle \langle A_1, \ldots, A_n \rangle, \langle \alpha_1, \ldots, \alpha_n \rangle \rangle$$

.

The attack on the Challenge-Response protocol is given by the sequence

$$A_1 = SendMessage(P, N, K)$$
$$\alpha_1 = HasMessage(P, N, K)$$
$$A_2 = SendMessage(P, N, \lambda)$$
$$\alpha_2 = HasMessage(P, N, \lambda).$$

Regardless of the underlying revision operator, the final belief state following this sequence will contain the state s_0 that satisfies only the following atomic formulas:

$$HasMessage(P, N, K), HasMessage(P, N, \lambda).$$

Clearly s_0 is not an element of $|HasMessage(Q, N, \lambda)|$, so the protocol fails to establish the goal.

Note that the problem with the Challenge-Response Protocol is that a single agent can play both the P role and the Q role in interweaved runs of the protocol. Our analysis makes this fact clear, because we have explicitly indicated that the receipt of $\{N\}_K$ causes P to believe that either P or Q has the message N. If P only uses the protocol to check for the aliveness of another agent, then the given attack is no longer a problem.

Discussion

The fundamental insight underlying BAN-like logics is that cryptographic protocols have epistemic goals. As such, the logical approach to protocol verification employs epistemic logics to represent the beliefs of each agent following a run of a given protocol. However, the logics use monotonic rules of inference to reason about changing beliefs. It is well-known that belief change is often a non-monotonic process in which beliefs need to be retracted in response to new information. As such, we have proposed that a more appropriate model of epistemic change in which protocol verification can be defined by introducing non-monotonic belief change operators.

In this paper, we have used belief evolution operators to reason about a specific protocol. The details of this particular approach are not important for our overall suggestion. The basic problem that arises in reasoning about protocols is that agents may have incorrect beliefs, and we need to be able to resolve such beliefs without lapsing into inconsistency. AGM belief revision operators provide a simple tool for handling this kind of problem, but we can not simply use AGM operators because we need to incorporate some notion of change.

Although we have focussed on authentication protocols, we could easily apply the same methods to more complex protocol goals, such as non-repudiation and anonymity. However, using belief evolution operators as the underlying approach is somewhat limited in that agents can not explicitly reason about failed or exogenous actions. For example, we might be interested in agents that are able to make inferences of the form: if α is believed at time 2, then it is believed that a message was intercepted at time 1. A more flexible formalism that is suitable for such problems is presented in (Hunter & Delgrande 2006).

At the most basic level, our goal in this paper is simply to connect two communities. The logical approach to protocol verification is explicitly concerned with belief change, yet standard approaches have not been informed by formal work on belief change. Similarly, there is a long history of studying formal properties of belief change, but there are relatively few practical applications. Connecting existing work in protocol verification with existing work in belief change is beneficial for both communities.

References

Adadi, M., and Tuttle, M. 1991. A semantics for a logic of authentication. In *Proceedings of the 10th ACM Symposium on Principles of Distributed Computing*, 201–216. ACM Press.

Agray, N.; van der Hoek, W.; and de Vink, E. 2002. On BAN logics for industrial security protocols. In Dunin-Keplicz, B., and Nawarecki, E., eds., *Proceedings of CEEMAS 2001*, 29–36.

Alchourrón, C.; G ardenfors, P.; and Makinson, D. 1985. On the logic of theory change: Partial meet functions for contraction and revision. *Journal of Symbolic Logic* 50(2):510–530.

Bleeker, A., and Meertens, L. 1997. A semantics for BAN logic. In *Proceedings of DIMACS Workshop on Design and Formal Verification of Security Protocols*.

Burrows, M.; Abadi, M.; and Needham, R. 1989. A logic of authentication. Technical Report 39, Digital Systems Research Center.

Burrows, M.; Abadi, M.; and Needham, R. 1990. A logic of authentication. *ACM Transactions on Computer Systems* 8(1):18–36.

Darwiche, A., and Pearl, J. 1997. On the logic of iterated belief revision. *Artificial Intelligence* 89(1-2):1–29.

Gelfond, M., and Lifschitz, V. 1998. Action languages. *Linköping Electronic Articles in Computer and Information Science* 3(16):1–16.

Guttman, J., and Thayer, J. 2000. Authentication tests. In *Proceedings 2000 IEEE Symposium on Security and Privacy*.

Hunter, A., and Delgrande, J. 2005. Iterated belief change: A transition system approach. In *Proceedings of IJCAI05*, 460–465.

Hunter, A., and Delgrande, J. 2006. Belief change in the context of fallible actions and observations. In *Proceedings of AAAI06*.

Lang, J. 2006. About time, revision, and update. In *Proceedings of NMR2006*.

Syverson, P., and Cervesato, I. 2001. The logic of authentication protocols. In Focardi, R., and Gorrieri, R., eds., *Foundations of Security Analysis and Design*, volume 2171 of *Lecture Notes in Computer Science*. Springer-Verlag. 63–136.

Syverson, P., and van Oorschot, P. 1996. A unified cryptographic protocol logic. Technical Report 5540-227, Naval Research Lab.