Belief Change and Cryptographic Protocol Verification

Aaron Hunter[1], James Delgrande[2]

[1] MacDonald, Dettwiler and Associates Ltd.
Richmond, B.C., Canada
hunter@cs.sfu.ca
[2] School of Computing Science,
Simon Fraser University
Burnaby, BC, Canada
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 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. We illustrate that this kind of reasoning is particularly important when protocol participants have incorrect beliefs.

Keywords. belief change, belief evolution, cryptographic protocol verification

1 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, & Needham1989] and continuing with several related logics [Adadi & Tuttle1991,Agray, van der Hoek, & de Vink2002,Bleeker & Meerter
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.

In this paper, we 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

Dagstuhl Seminar Proceedings 07351
Formal Models of Belief Change in Rational Agents
http://drops.dagstuhl.de/opus/volltexte/2007/1206

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.

2 Preliminaries

2.1 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 & Cervesato2001]. 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 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 & Thayer2000], 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.

2.2 Logics of Belief: BAN Logic

As noted above, the first logical approach to protocol verification was the so-called BAN logic. See [Burrows, Abadi, & Needham1990] for a formal introduction to the logic. We 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 ***controls*** X: P has jurisdiction on X. P is an authority on X and should be trusted on this matter.
- ***fresh***(X): Formula X has not been sent in any message at any time before the current protocol.
- $P \stackrel{K}{\longleftrightarrow} Q$: K is a good key for communication between P and Q
- ***PK***(P, k): K is a public key of P. K^{-1} is the corresponding secret key.
- $P \stackrel{X}{\leftrightharpoons} Q$: X is a secret (e.g. a password) known only to P and Q.
- $\{X\}_K$: X is encrypted under key K.

The intended semantics of these constructions is given through a collection of rules of inference. Space doesn't allow a full listing of the rules; however a consideration of two such rules gives a good indication of the overall approach. For example, the following rule concerns shared keys:

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

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 said X (and implicitly sent the message).

Similarly, for public keys there is the rule:

$$\frac{P \textit{ believes } PK(Q,K) \qquad P \textit{ received } \{X\}_{K^{-1}}}{P \textit{ believes } Q \textit{ said } X}.$$

Thus if P believes that K is Q's public key and P receives a message encoded with this key, then P concludes that Q said X.

¿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 Vink2002]. However, several different semantics have been proposed [Adadi & Tuttle1991,Bleeker & Meertens1997,Syverson & Cervesa and new protocol logics have been introduced based on the standard semantics for epistemic logic [Syverson & van Oorschot1996]. 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.

3 Non-Monotonic Belief Change

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 protocol 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.

3.1 Belief Update and Belief Revision

We will represent action effects by transition systems, as defined in [Gelfond & Lifschitz1998]; note however that our definition of a transition system is a restriction of that in [Gelfond & Lifschitz1998]. Define an *action signature* to be a pair $\langle \mathbf{A}, \mathbf{F} \rangle$ where $\mathbf{A}, \mathbf{F}$ are non-empty sets of symbols. We call $\mathbf{A}$ the set of *action symbols*, and $\mathbf{F}$ the set of *fluent symbols*.

Formally, the fluent symbols in $\mathbf{F}$ are propositional variables. The action symbols in $\mathbf{A}$ denote the actions that an agent may perform. We assume that the effects of actions are given by a *transition system*.

Definition 1. *A transition system T for an action signature $\sigma = \langle \mathbf{A}, \mathbf{F} \rangle$ is a pair $\langle S, R \rangle$ where*

1. $S \subseteq 2^{\mathbf{F}}$
2. $R \subseteq S \times \mathbf{A} \times S$.

The set S is called the set of *states* and R is the *transition relation*. For $F \in \mathbf{F}$, if $F \in s \in S$, then then we say that F is true in s; otherwise F is false in s. If $(s, A, s') \in R$, then we think of the state s' as a possible resulting state that could occur if the action A is executed in state s.

Transition systems can be visualized as directed graphs, where each node is labeled with a state and each edge is labeled by an action symbol. 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'.

An agent's *belief state* κ is represented by a set of states, informally consisting of those 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 as follows.[1]

Definition 2. *The update operator $\diamond$ is the function $\diamond : 2^S \times \mathbf{A} \rightarrow 2^S$ given by:*

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

Hence, the semantics of belief update is based on projecting an 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*.[2] In the interest of space, we assume that the reader is familiar with the AGM approach to belief revision [Alchourrón, G ardenfors, & Makinson1985]. AGM revision is usually 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.

We define a specific belief revision operator that we call *topological belief revision* [Hunter & Delgrande2007]. It is based on the intuition that a plausibility ordering for an agent's beliefs will be based on how "easy" it is to move from one state to another, and that this in turn will be based on the number of actions required to move from one state to another. Given a transition system $T = \langle S, R \rangle$, for $s, s' \in S$, define $d(s, s')$ to be the length of the shortest path between s and s' in S (and a suitably large number if there is no such path).

[1] More accurately, our operator is a *belief progression* operator; it has been argued elsewhere that update is a special case of belief progression [Lang2006].

[2] This is slightly simplistic, but suffices here. See [Friedman & Halpern1999,Lang2006]

Definition 3. *Let $T = \langle S, R\rangle$ be a* transition system *and let d be the topological distance function defined above. The* topological revision function $* : 2^S \times 2^S \rightarrow 2^S$ *is defined as follows*

$$\kappa * \alpha \;=\; \{w \in \alpha \mid \exists v_1 \in \kappa \textit{ such that for all } v_2 \in \alpha, v_3 \in \kappa$$
$$\textit{we have } d(w, v_1) \leq d(v_2, v_3)\}.$$

We obtain the following result.

Theorem 1. *The operator given in Definition 3 satisfies the AGM belief revision postulates.*

3.2 Belief Evolution

Protocol verification involves a combination of belief update and belief revision. When an agent sends a message, 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.[3] Thus, 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 & Delgrande2005]. 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 fixed update and revision operators. 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.

We need to introduce some notation. In particular, let $s^{-1}(A)$ denote the set of all states s' such that $(s', A, s) \in R$. We call $s^{-1}(A)$ the *pre-image* of s with respect to A. The next definition generalizes this idea to give the pre-image of a set of states with respect to a sequence of actions. In the definition, given any sequence of actions $\bar{A} = \langle A_1, \ldots, A_n\rangle$, we write $s \rightsquigarrow_{\bar{A}} s'$ to indicate that there is a path from s to s' that follows the edges labeled by the actions $A_1, \ldots, A_n$.

[3] There is a subtlety here: if an agent receives a message, then arguably this is the result of some earlier *send* action. However, the same message may come from different *send* actions, or action sequences. Topological revision reflects this fact.

Definition 4. *Let T be a transition system, let $\bar{A} = \langle A_1, \ldots, A_n \rangle$ and let α be an observation. Define $\alpha^{-1}(\bar{A}) = \{s \mid s \rightsquigarrow_{\bar{A}} s' \text{ for some } s' \in \alpha\}$.*

Hence, if the actual world is an element of α following the action sequence $\bar{A}$, then the initial state of the world must be in $\alpha^{-1}(\bar{A})$.

We have the following definition for *belief evolution*, $\circ$. In the definition, if $i \leq n$ then we let $\bar{A}_i$ denote the subsequence of actions $\langle A_1, \ldots, A_i \rangle$.

Definition 5. *Let κ be a belief state, let $\bar{A}$ be a sequence of actions of length n and let $\bar{\alpha}$ be a sequence of observations of length n. Assume that $\bar{A}$ and $\bar{\alpha}$ are mutually* consistent *in that each observation α_i is possible, given that the actions $(\bar{A}_j)_{j \leq i}$ have been executed. Define*

$$\kappa \circ \langle \bar{A}, \bar{\alpha} \rangle = \langle \kappa_0, \ldots, \kappa_n \rangle$$

where

1. $\kappa_0 = \kappa * \bigcap_i \alpha_i^{-1}(\bar{A}_i)$
2. *for* $i \geq 1$, $\kappa_i = \kappa_{i-1} \diamond A_1 \diamond \cdots \diamond A_i$.

We refer the reader to [Hunter & Delgrande2005] for the details.

4 Belief Change in Protocol Verification

We can think of cryptographic protocols as constraints on the sequences of messages that are exchanged in a *message passing system*. In this section, we introduce message passing systems, and we give specific revision and update operators that are suitable for reasoning about the belief change that occurs when messages are sent and received. We then formalize cryptographic protocol verification in terms of belief change operators. 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.

4.1 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[4] asserting that a participant has a certain key or a certain encrypted message. Such statements are the only assertions that are included 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 reflects 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 $\langle S, R\rangle$. The set of states S consists of all $s \in 2^{\mathbf{F}}$ such that $HasKey(P, \lambda) \in s$ for every P. Intuitively, the relation R will 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, R 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 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.

[4] That is, to be clear, we use a pseudo-first-order notation to represent propositional atoms.

Example 1. 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: $HasKey(x, y)$ and $HasMessage(x, N, y)$ for $x \in \{P, Q, I_Q\}$ and $y \in \{K, \lambda\}$, giving a total of 12 atomic formulas.

The initial state in the Challenge-Response protocol is the state s given by the following set of atoms

$$\{HasKey(P, \lambda), HasKey(Q, \lambda), HasKey(I_Q, \lambda),$$
$$HasKey(P, K), HasKey(Q, K),$$
$$HasMessage(P, N, \lambda), HasMessage(P, N, K)\}.$$

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

4.2 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 a message 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 a transition system. As well, arguably a topological revision operator is suitable as a specific revision operator to employ here.

Consequently, for a transition system $\langle S, R\rangle$ giving the effects of actions for a message passing system, we let $\diamond$ be the update operator given in Definition 2 and $*$ be the corresponding topological revision operator of Definition 3. 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.

4.3 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.

In BAN logic there are four steps in a protocol analysis [Syverson & van Oorschot1996]:

1. Idealize the protocol.
2. Give assumptions about the initial state
3. Annotate the protocol. So express the protocol using BAN assertions.
4. Use the logic to derive the beliefs held by the protocol participants.

By contrast, for the proposed approach we have the following steps:

1. Idealize the protocol. In this case, express the protocol by means of *send* actions defined by a transition system.
2. Give assumptions about the initial state
3. Use belief evolution with respect to the transition system to derive the beliefs of a protocol participant.

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
⋮	
2n-1.	A_n
2n.	α_n

In protocol verification, we typically assume that the principal 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 subsequence of a larger sequence of actions $ACT = \langle ACT_1, \ldots, ACT_m\rangle$ and $\alpha_1, \ldots, \alpha_n$ is a subsequence of a larger sequence of observations $OBS = \langle OBS_1, \ldots, OBS_m\rangle$ does it follow that
>
> $$\kappa \circ \langle ACT, OBS\rangle \models \phi?$$

¿From the perspective of a single agent, this is equivalent to asking if the action-observation sequence

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

guarantees that the agent will believe ϕ.

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 2. 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 M. 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 $X \in \mathbf{P}, Y \in \mathbf{M}$, if $s \models HasMessage(X, M, K)$ then $s \models HasMessage(X, M, \lambda)$
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. The receipt of the message N indicates that some agent involved in the protocol has the message N. Since N contains random information that cannot be guessed, P assumes that the only way an agent can have message N is by decrypting the message $\{N\}_K$ during this

protocol run. Since Q and P are the only agents that have the key K, we can identify the receipt of N 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

$$\begin{aligned} A_1 &= SendMessage(P, N, K) \\ \alpha_1 &= HasMessage(P, N, K) \\ A_2 &= SendMessage(P, N, \lambda) \\ \alpha_2 &= HasMessage(P, N, \lambda). \end{aligned}$$

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 a model of $HasMessage(Q, N, \lambda)$, so the protocol fails to establish the goal.

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.

To demonstrate that a protocol fails to establish an epistemic goal, one needs to prove that there is a corresponding sequence of updates and revisions where an agent will not believe that the goal is true. To show that a protocol does establish a goal, one needs to show that the goal is believed following *all* sequences of updates and revisions satisfying constraints given by the protocol. While there can be infinitely many update/revision sequences for a given (finite) transition system, nonetheless it is easily shown that an update/revision sequence that corresponds to a cycle in the transition system is equivalent to one that incorporates no cycles. Hence to determine whether a protocol establishes a goal, one needs check all *paths* in the transition system, of which there are only finitely many.

5 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 it is more appropriate to model epistemic change in protocol verification 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 state change. Although we have focused on authentication protocols, we could easily apply the same methods to more complex protocol goals, such as non-repudiation and anonymity.

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 & Tuttle1991] 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, van der Hoek, & de Vink2002] 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, G ardenfors, & Makinson1985] Alchourrón, C.; Gärdenfors, 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 & Meertens1997] 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, Abadi, & Needham1989] Burrows, M.; Abadi, M.; and Needham, R. 1989. A logic of authentication. Technical Report 39, Digital Systems Research Center.

[Burrows, Abadi, & Needham1990] Burrows, M.; Abadi, M.; and Needham, R. 1990. A logic of authentication. *ACM Transactions on Computer Systems* 8(1):18–36.

[Darwiche & Pearl1997] Darwiche, A., and Pearl, J. 1997. On the logic of iterated belief revision. *Artificial Intelligence* 89(1-2):1–29.

[Gelfond & Lifschitz1998] Gelfond, M., and Lifschitz, V. 1998. Action languages. *Linköping Electronic Articles in Computer and Information Science* 3(16):1–16.

[Guttman & Thayer2000] Guttman, J., and Thayer, J. 2000. Authentication tests. In *Proceedings 2000 IEEE Symposium on Security and Privacy*.

[Friedman & Halpern1999] Friedman, N., and Halpern, J. 1999. Belief Revision: A Critique. In *Journal of Logic, Language and Information*. 8(4):401–420.

[Hunter & Delgrande2005] Hunter, A., and Delgrande, J. 2005. Iterated belief change: A transition system approach. In *Proceedings of IJCAI05*, 460–465.

[Hunter & Delgrande2007] Hunter, A., and Delgrande, J. 2007. An Action Description Language for Iterated Belief Change. In *Proceedings of IJCAI07*, 2498-2503.

[Lang2006] Lang, J. 2006. About time, revision, and update. In *Proceedings of NMR2006*.

[Syverson & Cervesato2001] 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 & van Oorschot1996] Syverson, P., and van Oorschot, P. 1996. A unified cryptographic protocol logic. Technical Report 5540-227, Naval Research Lab.