

A Semantic Approach for Reasoning about Security Protocols (extended abstract)

Arjen Hommersom^{1,*}, John-Jules Meyer², Erik de Vink^{3,4}

¹ Nijmegen Institute of Information and Computing Sciences, University of Nijmegen

² Institute of Information and Computer Science, University of Utrecht

³ Department of Mathematics and Computer Science, Technische Universiteit Eindhoven

⁴ Leiden Institute of Advanced Computer Science, Leiden University

Abstract We present a model-theoretic approach for reasoning about security protocols, applying recent insights from dynamic epistemic logics. This enables us to describe exactly the subsequent epistemic states of the agents participating in the protocol, using Kripke models and transitions between these based on updates of the agents' beliefs associated with steps in the protocol. As a case study we will consider the SRA Three Pass protocol.

1 Introduction

In today's world of e-commerce and the Internet, the role of security protocols is getting increasingly important. The design of these security protocols is difficult and error-prone [Sch00, And01], which makes (automatic) verification of protocols of crucial importance. Since the late eighties, one line of research, amongst others, for reasoning about security protocols is based on the use of the so-called BAN logic, proposed by Burrows, Abadi and Needham in [BAN90]. This is an epistemic logic augmented by constructs that are relevant for reasoning about security, such as the property of having the disposal of a cryptographic key to be able to decode a message and therefore to know its contents. Although many useful results having been reported (e.g., [KN98, AHV01, SW02]), due to their complexity and their semantic underpinning the use of BAN logics to prove the correctness of security protocols has so far not been very successful (cf. [AT91, BM97, WK96, SC01]).

In this paper we will apply insights from dynamic epistemic logics as recently developed by Gerbrandy [Ger97, Ger99], Baltag [BMS99, Bal00], Van Ditmarsch [Dit00, Dit01], and Kooi [Koo03]. Moreover, contrary to the traditional BAN logic approach, our approach is semantic or model-theoretic. We use Kripke models to represent the epistemic state of the agents involved in a protocol, similarly to the S5 preserving approach of Van Ditmarsch to analyze certain kinds of games involving knowledge. From Baltag's action models we import the idea to describe belief updates of the agents by semantic operators transforming the Kripke models at hand by copying and deleting parts of these models, although we use traditional Kripke models rather than Baltag's action models. To this end we need also operators for unfolding

*Corresponding author: arjenh@cs.kun.nl

models, which is in its turn inspired by Gerbrandy’s work on possibilities. The difference being that in our approach only *partial* unfolding is called for. We furthermore propose a language to express belief updates in the context of security protocols as well as properties of these updates, and give a semantics of this language in terms of the models mentioned and the operators on them. Since our approach is model-theoretic, we believe that it may serve as a starting point for the automatic verification of (properties of) security protocols.

As a case study illustrating our approach we will consider the so-called SRA Three Pass protocol and prove a property of it. It is not our intention to prove that the protocol is completely secure (as it is not in full generality), but we will prove that if the agents participating in the protocol are honest, then an intruder watching the communication does not learn anything about the plain-text messages in a single run. Furthermore we show what the intruder is able to learn about the agents participating.

2 Preliminaries

In this section we briefly discuss some preliminaries and background regarding the updates we will handle and the epistemic model we will use. First, we define objective formulas and so-called o-seriality.

Definition 2.1. Fix a set \mathcal{P} of propositional variables. The class of objective formulas is the smallest class such that:

- all propositional variables or atoms $p \in \mathcal{P}$ are objective;
- if ϕ is objective, then $\neg\phi$ is objective;
- if ϕ_1 and ϕ_2 are objective, then $\phi_1 \wedge \phi_2$ is objective.

So, objective formulas do not involve beliefs. For our purposes it is important that every agent distinguishes a world with the same ‘objective’ information. This leads to the notion of an o-serial model.

Definition 2.2. A model $M = \langle S, \pi, R_1, \dots, R_m \rangle$ is *o-serial* iff for all i , $1 \leq i \leq m$, and $w \in S$, there exists $v \in S$ such that $(w, v) \in R_i$ and for all objective formulas ϕ it holds that $(M, w) \models \phi \Leftrightarrow (M, v) \models \phi$.

We use a, b, c , etc. as typical agents, taken from a class \mathcal{A} . We use the notation $\{x\}_{k_a}$ to denote a message x encrypted with the cryptographic key k_a of agent a . Furthermore, B is used as a doxastic modal operator. For example, $B_a\phi$ should be read as ‘ a believes ϕ ’. We interpret formulas on standard Kripke models $(M, s) = (\langle S, \pi, R_1, \dots, R_m \rangle, s)$, where $(M, s) \models B_i\phi$ iff $\forall t \in S: R_i(s, t) \rightarrow (M, t) \models \phi$.

We require the relations R_i to be o-serial, transitive and euclidean. This yields a class of models that we will call Kt45, a proper subset of the class of models of the well-known doxastic logic $KD45$. The lower case t refers to the axiom

$$B_i\phi \Rightarrow \phi, \tag{t}$$

for every objective ϕ . The system Kt45 is sound with respect to the class of o-serial, transitive and euclidian models [Hom03]. We will show that the operations we introduce preserve Kt45.

The point is that in worlds of Kt45 models, we cannot both have $B_i\phi$ and $B_i\neg\phi$, for an objective formula ϕ . This is reasonable from our assumption that agents are conscious about the protocol. Therefore, they will not infer objective contradictions. This objectivity is captured locally for each state. As a consequence, the operations that we introduce can restrict the set of states without destroying objective information.

For the analysis of security protocols below, we assume that we are omniscient about the values of the variables in different runs of a protocol. For example, the program variable p in a protocol run has the value $\llbracket p \rrbracket$. In the real world it is, obviously, always true that $p = \llbracket p \rrbracket$. However, it is cumbersome to keep track of what is the real world in the operations on Kripke structures that we employ below. Therefore, we assume that an interpretation $\llbracket \cdot \rrbracket$ is given, that provides the ‘real’ values of the program variables when needed. It might very well be the case that $p \neq \llbracket p \rrbracket$ in a certain state. From now on, we will abbreviate $p = \llbracket p \rrbracket$ to p on (thus transforming a program expression into a propositional variable). Similarly, $\neg p$ is an abbreviation of $p \neq \llbracket p \rrbracket$. For example, agent a that learns $B_b p \vee B_b \neg p$, learns that agent b has assigned a value to the program variable p .

The types of updates we consider are (i) public announcement of a variable, (ii) the private learning of a variable and (iii) the private learning about the knowledge of other agents.

The first type of update typically runs as follows: In an open network agent a sends a message to agent b . From a security perspective, it is customary [DY83] to assume that all agents in the network can read this message too. However, also in open networks private learning, the second type of update, can take place. For example, agent b receives a message $\{x\}_k$ from agent a . Here $\{x\}_k$ denotes a message with content x encrypted with the (symmetric) key k . If b possesses the key k , then b privately learns the message content x (assuming that the key k is shared among a and b). The final type of update, learning about knowledge of others, is probably the most interesting. It is realistic to assume that the steps in a protocol run are known to all agents. Therefore, observing that an agent receives a message will increase the knowledge of the other agents. For example, if agent a sends a message $\{x\}_k$ to agent b , then agent c learns that b has learned the information contained in the message $\{x\}_k$, but typically, c does not learn x if c does not possess the key k .

Stronger types of updates we do not consider here. For example, we will not update the beliefs of an honest agent such that it learns that an intruder has learned about others. In the present paper, we restrict ourselves to updating beliefs about objective formulas and beliefs about objective formulas.

3 Update constructions

In this section we describes various types of updates in detail. We will start by defining an update for propositions in subsection 3.1. In subsection 3.2 we will define a belief update for agents that learn something about the belief of others. We do this in two slightly different ways by varying in the functions that describe a side-effect for an agent.

3.1 Objective updates

The belief update of objective formulas we will use is based on [RHM02, BMS99]. The construction works as follows: We will make copies of the states of the model such that the *old* worlds in $old(S)$ correspond to the information in the original model and the *new* worlds in $new(S)$ correspond to the new information.

Definition 3.1. Let a world $(M, w) = (\langle S, \pi, R_1, \dots, R_m \rangle, w)$, a group of agents \mathcal{B} , and an objective formula ϕ be given. Then $\text{EXPAND}_{(\phi, \mathcal{B})}(M, w) = (\langle S', \pi', R'_1, \dots, R'_m \rangle, w')$, where

- $S' = \{\text{new}(s) \mid (M, s) \models \phi\} \cup \text{old}(S)$
- $w' = \text{new}(w)$
- for all $p \in \mathcal{P}$: $\pi'(\text{old}(u))(p) = \pi'(\text{new}(u))(p) = \pi(u, p)$
- for $1 \leq i \leq m$ the relation R'_i on S' is minimal such that:

$R'_i(\text{old}(u), \text{old}(v))$	\Leftrightarrow	$R_i(u, v)$
$R'_a(\text{new}(u), \text{new}(v))$	\Leftrightarrow	$R_a(u, v) \wedge (M, v) \models \phi$ if $a \in \mathcal{B}$
$R'_b(\text{new}(u), \text{old}(v))$	\Leftrightarrow	$R_b(u, v)$ if $b \notin \mathcal{B}$

The following example shows how this works on a concrete model.

Example 3.1. Consider the model (M, s) in Figure 1 where $\pi(s)(p) = \text{true}$ and $\pi(t)(p) = \text{false}$. The operation we execute is that b learns p . This results in the model (M, u) in Figure 2 where $\pi(u)(p) = \pi(v)(p) = \text{true}$ and $\pi(w)(p) = \text{false}$ and $\text{new}(s) = u, \text{old}(s) = v$ and $\text{old}(t) = w$. The world $\text{new}(t)$ is unreachable from the actual world and is therefore omitted from the figure. (In fact, in all figures in this paper, we will omit the unreachable worlds.)

We can see that the belief of agent a has not changed: it still considers its old worlds possible. The belief of agent b however, has changed. It now only considers the state u possible where p holds.

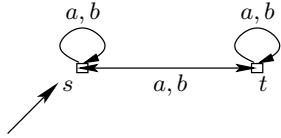


Figure 1: (M, s)

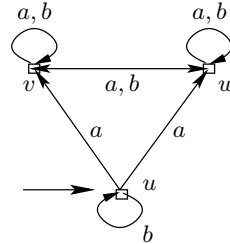


Figure 2: (M, u)

The update $\text{EXPAND}_{(\phi, \mathcal{B})}$ is based on a formula ϕ and a set of agents \mathcal{B} . Roorda *et al.* [RHM02] have given a characterization of the formulas that are altered by such an operation with a single learning agent. Here we extend the definition for multi-agent purposes.

Definition 3.2. An update function $(\cdot)[\phi, \mathcal{B}]$ is called proper if

- | | | |
|---|-------------------|---|
| $(M, w)[\phi, \mathcal{B}] \models p$ | \Leftrightarrow | $(M, w) \models p$ |
| $(M, w)[\phi, \mathcal{B}] \models \alpha \wedge \beta$ | \Leftrightarrow | $(M, w)[\phi, \mathcal{B}] \models \alpha$ and $(M, w)[\phi, \mathcal{B}] \models \beta$ |
| $(M, w)[\phi, \mathcal{B}] \models \neg\alpha$ | \Leftrightarrow | $(M, w)[\phi, \mathcal{B}] \not\models \alpha$ |
| $(M, w)[\phi, \mathcal{B}] \models B_a\alpha$ | \Leftrightarrow | $(M, w) \models B_a\alpha$ if $a \notin \mathcal{B}$ |
| $(M, w)[\phi, \mathcal{B}] \models B_b\alpha$ | \Leftrightarrow | $\forall u: ((B_b(w, u) \wedge (M, u) \models \phi) \Rightarrow (M, u)[\phi, \mathcal{B}] \models \alpha)$ if $b \in \mathcal{B}$ |

Following Roorda *et al.* we have that $\text{EXPAND}_{(\phi, \mathcal{B})}$ is proper. Moreover, $\text{EXPAND}_{(\phi, \mathcal{B})}$ is uniquely characterized by Definition 3.2 upto elementary equivalence, i.e. if $(\cdot)[\phi, \mathcal{B}]$ is a proper update function, then $(M, w)[\phi, \mathcal{B}]$ and $\text{EXPAND}_{(\phi, \mathcal{B})}(M, w)$ are elementary equivalent. We collect the following properties of $\text{EXPAND}_{(\phi, \mathcal{B})}$.

Lemma 3.1.

- For all ϕ it holds that $(M, w) \models \phi \Rightarrow \text{EXPAND}_{(\phi, \mathcal{B})}(M, w) \models B_{\mathcal{B}}\phi$.
- If (M, w) satisfies the Kt45 properties and ϕ is objective, then $\text{EXPAND}_{(\phi, \mathcal{B})}(M, w)$ satisfies the Kt45 properties as well.
- $\text{EXPAND}_{(\psi, \mathcal{C})}(\text{EXPAND}_{(\phi, \mathcal{B})}(M, w))$ and $\text{EXPAND}_{(\phi, \mathcal{B})}(\text{EXPAND}_{(\psi, \mathcal{C})}(M, w))$ are bisimilar.

Note that in Lemma 3.1 ϕ ranges over arbitrary formulas, including non-objective ones. However, for a non-objective formula $B_i\phi$, it can happen that, unintended, an agent increases the objective knowledge encapsulated by the formula ϕ . This is illustrated by the next example.

Example 3.2. Suppose we are interested in agent a learning the formula $B_b p \vee B_b \neg p$, but not p itself. Consider the Kripke model (M, s) in Figure 3, where $\pi(s)(p) = \pi(u)(p) = \text{true}$ and $\pi(t)(p) = \text{false}$. This models the state where b knows that p is true. Agent a does not know p or $\neg p$, and it does not know if b knows p .

If we apply the definition of the belief expansion function, it results in the model (M, v) from Figure 4, where $\pi(v)(p) = \pi(s)(p) = \pi(u)(p) = \text{true}$ and $\pi(t)(p) = \text{false}$. The reason that it turns out like this, is because the only state where $B_b p \vee B_b \neg p$ holds, is the state s . Thus, all the other states have no corresponding *new* states. Figure 4 illustrates that a has learned $B_b p \vee B_b \neg p$, but also that a has learned p itself.

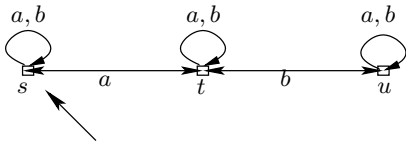


Figure 3: (M, s)

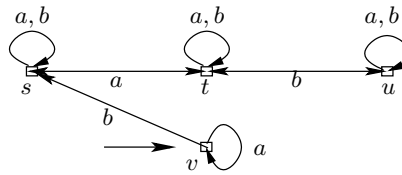


Figure 4: (M, v)

In the next subsection we will define a side-effect function such that a will learn about others, but does not learn any objective formulas itself.

3.2 Side-effects

The main reason that an update of $B_b p \vee B_b \neg p$ for agent a fails, is that it actually deletes the wrong arrows. The belief expansion function deletes arrows of a to gain the states that satisfy the updating formula. This is not what we intend. We want a to keep all the states it considers possible, but at the same time we would like to update all the possible states of a such that the formula $B_b p \vee B_b \neg p$ holds in these states. Moreover, we do not want to change the knowledge of other agents. In this section we define the functions that accomplish these requirements.

A technical obstacle is that states can be shared among agents. It is obvious that if we change a state with the intention to change the belief of one agent, then the belief of the other agents that consider this state possible, is changed as well. Therefore, the first thing to do, is separate the states of learning agents from the states of agents that do not learn. This procedure will be called *unfolding*. The functions $new_{\mathcal{B}}$ and $orig$ are generalizations of new and old from the previous section, but the function $orig$ is only defined on the point of the model (the actual world).

Definition 3.3. Given a model (M, w) with $M = \langle S, \pi, R_1, \dots, R_m \rangle$, a partitioning \mathcal{X} of \mathcal{A} , we define a function $UNFOLD_{\mathcal{X}}(M, w) = (\langle S', \pi', R'_1, \dots, R'_m \rangle, w')$, where

- $S' = (\bigcup_{\mathcal{B} \in \mathcal{X}} new_{\mathcal{B}}(S)) \cup orig(w)$
- $w' = orig(w)$
- $\pi'(new_{\mathcal{B}}(v))(p) = \pi(v)(p)$, $\pi'(w')(p) = \mathbf{true}$ for all $p \in \mathcal{P}$, $\mathcal{B} \in \mathcal{X}$
- the relation R'_i on S' is minimal such that

$$R'_i(new_{\mathcal{B}}(u), new_{\mathcal{C}}(v)) \Leftrightarrow R_i(u, v) \wedge (\mathcal{B} = \mathcal{C})$$

$$R'_i(orig(w), new_{\mathcal{B}}(u)) \Leftrightarrow R_i(w, u) \wedge i \in \mathcal{B}$$
 where \mathcal{B}, \mathcal{C} range over \mathcal{X} .

So for every group of agents \mathcal{B} there is copy of the original states (viz. $new_{\mathcal{B}}(s)$ for every $s \in S$). This function does indeed preserve our Kt45 properties and it models the same knowledge, which is captured in the following lemma.

Lemma 3.2.

- (a) If (M, w) is a Kt45 model and \mathcal{X} a partition, then $UNFOLD_{\mathcal{X}}(M, w)$ is a Kt45 model.
- (b) For every model (M, w) and partition \mathcal{X} , it holds that (M, w) and $UNFOLD_{\mathcal{X}}(M, w)$ are bisimilar.

Example 3.3 (unfold). Consider the Kripke model (M, s) in Figure 5 with $\pi(s)(p) = \pi(u)(p) = \mathbf{true}$, $\pi(t)(p) = \mathbf{false}$. So, b knows that p is true, while a does not. Furthermore, a does not know if b knows p . Now the operation we perform is $UNFOLD_{\{\{a\}, \{b\}\}}(M, s)$ which results in the model (M', s) in Figure 6.

So we have split the knowledge of a and b . The state s is the original state, the primed states model a 's knowledge and the double primed states model b 's knowledge. So the upper half of the model represents the knowledge of a , and the lower half represents the knowledge of b . Note that no states are shared, in particular because the point of the model is not reflexive.

Now we give some preparatory definitions. First we define the notion of a submodel.

Definition 3.4. A model $M = \langle S, \pi, R_1, \dots, R_m \rangle$ is a submodel of $M' = \langle S', \pi', R'_1, \dots, R'_m \rangle$, written as $M \sqsubseteq M'$, iff $S \subseteq S'$, $\pi(s)(p) = \pi'(s)(p)$ for all $s \in S$, $p \in \mathcal{P}$ and $R_i \subseteq R'_i$.

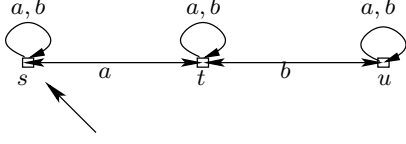


Figure 5: (M, s)

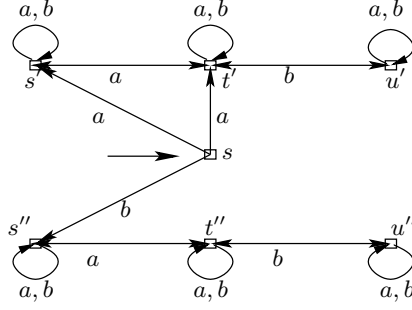


Figure 6: (M', s)

Next, we construct a submodel that represent the knowledge of an agent a .

Definition 3.5. Given a model $(M, w) = (\langle S, \pi, R_1, \dots, R_m \rangle, w) = \text{UNFOLD}_{\{\{a\}, \mathcal{B}\}}(M', w')$ such that $\{\{a\}, \mathcal{B}\}$ is a partition of \mathcal{A} , for some (M', w') , define the a -submodel $\text{SUB}_a(M) = \langle S', \pi', R'_1, \dots, R'_m \rangle$ where

- $s \in S' \Leftrightarrow \text{new}_a(s) \vee \text{orig}(s)$,
- for all $p \in \mathcal{P}$ it holds that $\pi'(\text{new}_a(s))(p) = \pi'(\text{orig}(s))(s) = \pi(s)(p)$,
- $R'_i(s, t) \Leftrightarrow R_i(s, t) \wedge s, t \in S'$.

Clearly an a -submodel is a submodel in the sense of Definition 3.4. The restmodel is the part of the model that is complements the submodel with respect to the accessibility relation.

Definition 3.6. Given a model $(M, w) = \langle S, \pi, R_1, \dots, R_m \rangle$ and a submodel $N = \langle S'', \pi'', R''_1, \dots, R''_m \rangle$ of M , define the restmodel $\text{REST}_N(M) = \langle S', \pi', R'_1, \dots, R'_m \rangle$ where

- $s \in S' \Leftrightarrow s \in S \wedge \exists (u, v) \in R'_i: u = s \vee v = s$
- $\pi'(s)(p) = \pi(s)(p)$ for all $p \in \mathcal{P}$
- $R'_i(s, t) \Leftrightarrow R_i(s, t) \wedge \neg R''_i(s, t)$

We can see the a -submodel and restmodel definitions in action by taking the model of example 3.3 and applying the above definitions. See Figure 7. This exactly corresponds to the idea of two submodels that represent knowledge of different agents.

Now we would like to update the belief of some agents. To this end, we want to replace the submodel that represents their belief by a new model. We will apply the following definition.

Definition 3.7. Given a model $N = \langle S, \pi, R_1, \dots, R_m \rangle$, a model M , a model N' such that $N \sqsubseteq N' \sqsubseteq M$ with $\text{REST}_{N'}(M) = \langle S', \pi', R'_1, \dots, R'_m \rangle$, we define $\text{REPLACE}_{N'}(N, M) = \langle S'', \pi'', R''_1, \dots, R''_m \rangle$ where

- $s \in S'' \Leftrightarrow s \in S \vee s \in S'$,
- for all $p \in \mathcal{P}$ it holds that $\pi''(s)(p) = \pi(s)(p)$ for $s \in S''$,

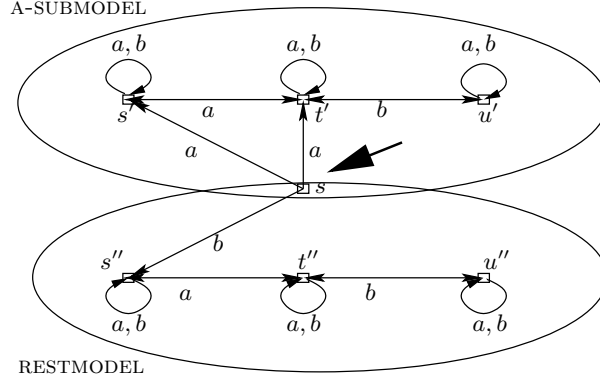


Figure 7: a-submodel outline

- $R_i''(s, t) \Leftrightarrow (s, t) \in R_i \vee (s, t) \in R_i'$.

The idea is that once the belief is completely separated, we can not only safely change the belief of a certain agent, but also preserve the Kt45 properties. The function $\text{ATOMSPLIT}_{(p,b)}$ removes the arrows of b between states that have a different valuation for p .

Definition 3.8. Given a model $M = \langle S, \pi, R_1, \dots, R_m \rangle$, we define a function $\text{ATOMSPLIT}_{(p,b)}(M) = \langle S', \pi', R'_1, \dots, R'_m \rangle$ as follows:

- $s \in S' \Leftrightarrow s \in S$,
- $\pi'(s)(p) = \pi(s)(p)$ for all $p \in \mathcal{P}$,
- $R_i'(s, t) \Leftrightarrow R_i(s, t)$ ($i \neq b$),
- $R_b'(s, t) \Leftrightarrow R_b(s, t) \wedge \pi(s)(p) = \pi(t)(p)$.

Finally we are in a position to define the actual side-effect function that ties these things together.

Definition 3.9. For a model (M', w') such that $(M', w') = \text{UNFOLD}_{\{\{a\}, \mathcal{A} \setminus \{a\}\}}(M, w)$ and $N = \text{SUB}_a(M')$ we define $\text{SIDE-EFFECT}_{(p,a,b)}(M, w) = (\text{REPLACE}_N(\text{ATOMSPLIT}_{p,b}(N), M'), w')$.

Example 3.4. We continue Example 3.3. To the a -submodel of M we now apply $\text{ATOMSPLIT}_{p,b}$ which results in the model (M'', s) in Figure 8. The arrow (t', u') has disappeared, since $\pi(t')(p) \neq \pi(u')(p)$. Therefore, u' is not reachable anymore, and can be dropped. Notice that a believes $B_b p \vee B_b \neg p$, while a has learned nothing about p itself.

We have the following results.

Lemma 3.3.

- If (M, w) is a Kt45-model, then $\text{SIDE-EFFECT}_{(p,a,b)}(M, w)$ is a Kt45-model as well.
- Given a model (M, w) , agents a, b, c, d , two propositions $p, q \in \mathcal{P}$, it holds that $\text{SIDE-EFFECT}_{(p,c,d)}(\text{SIDE-EFFECT}_{(q,a,b)}(M, w))$ and $\text{SIDE-EFFECT}_{(q,a,b)}(\text{SIDE-EFFECT}_{(p,c,d)}(M, w))$ are bisimilar.

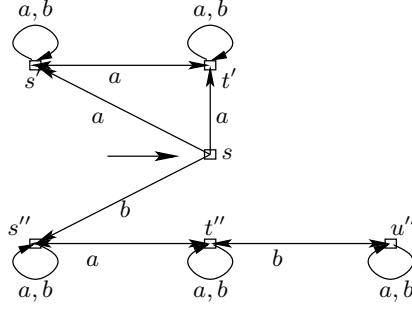


Figure 8: (M'', s)

(c) Given a model (M, w) , agents a, b, c , a propositions $p \in \mathcal{P}$ and an objective formula ϕ , it holds that $\text{SIDE-EFFECT}_{(p,c,d)}(\text{EXPAND}_{(\phi,a)}(M, w))$ and $\text{EXPAND}_{(\phi,a)}(\text{SIDE-EFFECT}_{(p,c,d)}(M, w))$ are bisimilar.

Next, we consider how the formulas are altered by the side-effect function. We will partially answer this by presenting a few interesting formulas that hold in the resulting model. We will look at groups of agents instead of a single agent:

1. the group of agents that learn about other agents, ranged over by a ;
2. the group of agents that is learned about, ranged over by b ;
3. other agents, ranged over by c .

The fact that the agents a are the only agent that learn at all, is clear. The other agents consider their old worlds possible; their belief has not changed. With this in mind, we present a few properties of the side-effect function.

Lemma 3.4. *Let a, b and c be three different agents. Given a model (M, w) and the model $(M', w') = \text{SIDE-EFFECT}_{(p,a,b)}(M, w)$, it holds that*

- (a) $(M', w') \models B_a(B_b p \vee B_b \neg p)$;
- (b) $(M', w') \models B_a \phi$ iff $(M, w) \models B_a \phi$, where ϕ objective;
- (c) $(M', w') \models B_a C_{abc}(B_b p \vee B_b \neg p)$;
- (d) $(M', w') \models B_i \phi$ iff $(M, w) \models B_i \phi$ for $i \neq a$.

In the above a is an agent that learns about another agent, viz. agent b ; agent b is the agent about whom is learned; agent c is an agent different from a and b . In part (a) of the above lemma, agent a obtains derived knowledge of an agent b . Part (b) states that no object knowledge is learned. Part (c) phrases that agent a considers the rest of the agents as smart itself. Finally, part (d) captures that other agents do not learn.

Property (c) is a reasonable assumption of a about the other agents. If one agent believes that another agent knows the value of p , then it is reasonable to assume that another agent will believe the same. On the other hand common knowledge might be too strong to assume. Next, we address the issue that an agent a shares its belief about an agent b with other agents. We represent this by linking a 's beliefs of those other agents back to the original (unmodified) states. We distinguish four different type of groups of agents.

1. the group A of agents that learn about other agents, ranged over by a ;
2. the group B of agents that is learned about, ranged over by b ;
3. the group C of agents of which agents a believe they have commonly learned about agents in group b with, ranged over by c ;
4. the group D of agents of which agents in a believe they have learned nothing about, ranged over by d .

For simplicity of presentation, we assume that there is exactly 1 agent present in each group, i.e. $\mathcal{A} = \{a, b, c, d\}$. We define the new side-effect operation 0-UNFOLD (where 0 refers to zero-knowledge). Note that the 0-UNFOLD operation depends on the particular partitioning of agents \mathcal{A} . Since the operation S0-IDE-EFFECT will depend on the unfold operation, the side-effect function is also taken with respect to some chosen partitioning of the agent set.

Definition 3.10. Given a model (M, w) such that $M = \langle S, \pi, R_1, \dots, R_m \rangle$ and the set of agents $\{a, b, c, d\}$, we define a function $0\text{-UNFOLD}(M, w) = (\langle S', \pi', R'_1, \dots, R'_m \rangle, w')$, where:

- $S' = \text{new}_a(S) \cup \text{new}_{bcd}(S) \cup \text{orig}(w)$
- $w' = \text{orig}(w)$
- $\pi'(\text{new}_B(v))(p) = \pi(v)(p)$ and $\pi'(\text{orig}(w))(p) = \pi(w)(p)$ for all $p \in \mathcal{P}$, $B \in \{\{a\}, \{bcd\}\}$
- R'_i on S' is minimal such that

$R'_d(\text{new}_a(u), \text{new}_{bcd}(v))$	\Leftrightarrow	$R_d(u, v)$
$R'_i(\text{new}_a(u), \text{new}_a(v))$	\Leftrightarrow	$R_i(u, v)$ ($i \neq d$)
$R'_i(\text{new}_{bcd}(u), \text{new}_{bcd}(v))$	\Leftrightarrow	$R_i(u, v)$
$R'_a(\text{orig}(w), \text{new}_a(v))$	\Leftrightarrow	$R_a(w, v)$
$R'_i(\text{orig}(w), \text{new}_{bcd}(v))$	\Leftrightarrow	$R_i(w, v) \wedge i = b, c, d$

So instead of completely separating the knowledge of the agent a with the other agents, we share the knowledge of a about d with the other agents. Since the other agents do not learn anything, a does not gain knowledge about d . We present a lemma similar to Lemma 3.2.

Lemma 3.5.

(a) If (M, w) is a *Kt45* model, then so is $0\text{-UNFOLD}(M, w)$.

(b) For every model (M, w) it holds that (M, w) and $0\text{-UNFOLD}(M, w)$ are bisimilar.

Due to the case distinction for $R'_a(\text{orig}(w), \text{new}_B(v))$ in Definition 3.10 above, it holds that the knowledge of a about b is not interconnected with the knowledge of other agents about b or of b itself. So we can ‘cut out’ the submodel containing the b arrows from the belief of a :

Definition 3.11. Given a model $(M', w') = \langle S', \pi', R'_1, \dots, R'_m \rangle$ such that $(M', w') = 0\text{-UNFOLD}(M, w)$, for some (M, w) , define $B\text{-SUB}(M') = \langle S'', \pi'', R''_1, \dots, R''_m \rangle$ where

- $S'' = \{ \text{new}_a(s) \mid s \in S \}$
- $\pi''(s)(p) \Leftrightarrow \pi(s)(p)$ for all $p \in \mathcal{P}$

- $R''_b(s, t) \Leftrightarrow R'_b(s, t)$ for $s, t \in S''$
- $R''_i = \emptyset$ ($i \neq b$).

The operation 0-SIDE-EFFECT is then given by $0\text{-SIDE-EFFECT}_p(M, w) = (\text{REPLACE}_N(\text{ATOMSPLIT}_{(p,b)}(N), M'), w')$ where $N = \text{B-SUB}(M')$.

Lemma 3.6. *If (M, w) is a $Kt45$ -model, then so is $0\text{-SIDE-EFFECT}_p(M, w)$.*

Similarly to Lemma 3.3 and Lemma 3.4 we have the following result.

Lemma 3.7. (a) *If (M, w) is a $Kt45$ -model, then so is $0\text{-SIDE-EFFECT}_p(M, w)$.*

(b) *Given a model (M, w) and the model $(M', w') = 0\text{-SIDE-EFFECT}_p(M, w)$, it holds that*

- (i) $(M', w') \models B_a(B_b p \vee B_b \neg p)$
- (ii) $(M', w') \models B_a \phi$ iff $(M, w) \models B_a \phi$ for ϕ objective
- (iii) $(M', w') \models B_a C_{abc}(B_b p \vee B_b \neg p)$
- (iv) $(M', w') \models B_i \phi$ iff $(M, w) \models B_i \phi$ for $i \neq a$
- (v) $(M', w') \models B_a B_d \phi$ iff $(M, w) \models B_a B_d \phi$

Parts (i) to (iv) are compare to the properties given in Lemma 3.4. Part (v) states that the knowledge of agent a about agent d has not changed, which exactly as desired.

Example 3.5 (side-effect function). Recall the model (M, s) from Example 3.3 (Figure 5). We now present this model with four agents $\{a, b, c, d\}$ in Figure 9 such that $\pi(s)(p) = \pi(u)(p) = \mathbf{true}$ and $\pi(t)(p) = \mathbf{false}$. Now apply $0\text{-SIDE-EFFECT}_p(M, w)$ and we gain the model (M', s) from Figure 10 such that $\pi(s)(p) = \pi(s')(p) = \pi(s'')(p) = \pi(u'')(p) = \mathbf{true}$, $\pi(t')(p) = \pi(t'')(p) = \mathbf{false}$. Note that in this model, a still knows exactly the same about d as it did before.

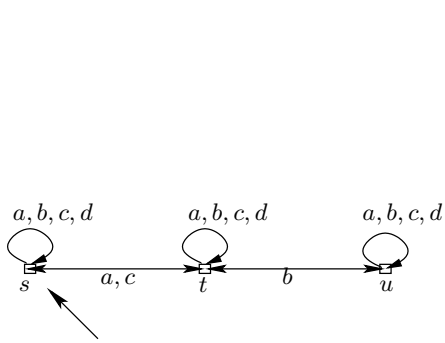


Figure 9: (M, s)

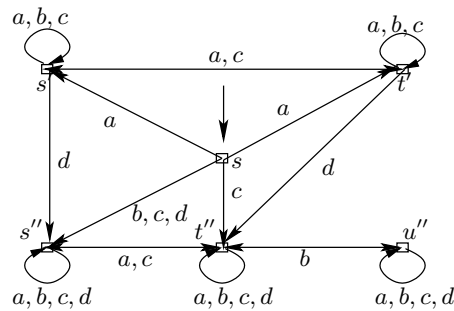


Figure 10: (M', s)

4 A logical language for security protocols

In this section we exploit the ideas of the previous section for a logical language to reason about security protocols. The EXPAND and SIDE-EFFECT operations are used for its semantics. It is illustrated for the case of the SRA Three Pass protocol how the various steps of the protocols change the initial knowledge of the agents involved and what Kripke-structure is finally obtained.

Definition 4.1. Fix a set of proposition \mathcal{P} , ranged over by p , and a set of agents \mathcal{A} of m elements, ranged over by a . The language \mathcal{L}_C is given by

$$\begin{aligned}\phi &::= p \mid \neg\phi \mid \phi_1 \wedge \phi_2 \mid B_i\phi \mid [\sigma]\phi \\ \sigma &::= \text{Priv}(i \rightarrow j, p) \mid \text{Pub}(i, p) \mid \sigma; \sigma'\end{aligned}$$

The σ symbol denotes a (possibly composed) communication action. The action $\text{Priv}(i \rightarrow j, p)$ is a private or peer-to-peer message p from i to j ; the action $\text{Pub}(i, p)$ means a public announcement or a broadcast by i about p . In the latter every agent on the network learns p , whereas in the former only j learns p . The bracket operator $[\sigma]\phi$ has the interpretation that after executing the communication action, ϕ holds.

The subscript in \mathcal{L}_C refers to a set of so-called transition rules \mathcal{C} . The transition rules capture the updates, i.e. the expansions and side-effects, necessary for the interpretation of the constructs $\text{Priv}(i \rightarrow j, p)$ and $\text{Pub}(i, p)$. The transitions rules enforce consistency among the propositions that hold. For example, if an agent believes that the value of a message m is $\llbracket m \rrbracket$ and possesses a key $k(a)$, then it must believe that the value of the encryption $E_{k(a)}(m)$ of m has a value that corresponds with $\llbracket m \rrbracket$.

A transition rule has either the form $B_i p \Rightarrow \beta$ or $H_i p \Rightarrow \beta$. $B_i p$ and $H_i p$ are conditions, expressing that p must be believed by agent i or that p has been delivered to agent i , respectively. The body β of a transition rule is a sequence of actions $\alpha_1; \dots; \alpha_n$. Actions come in two flavours $L_{\mathcal{B}} p$ and $S_{i,j} p$. Here, $L_{\mathcal{B}} p$ expresses that p is learned among the agents in the set \mathcal{B} and corresponds to belief expansion, whereas $S_{a,b} p$ expresses the side-effect that the agent a has learned that agent b now knows about p .

As an example, we will have the transition rule $B_b \{x\}_k \Rightarrow L_{ab} \{x\}_k$, when agents a and b share the key k and a sends b the message $\{x\}_k$. A typical application of an $H_i p \Rightarrow \beta$ transition rule is in a protocol that uses a challenge. In the situation described above, agent a sends the message x to agent b and agent b returns the message $\{x\}_k$. Since it is shared, a already can compute $\{x\}_k$ itself, so the delivery of $\{x\}_k$ does not teach a anything about this value. But, since it must come from b , it does learn that b has the shared key and authenticates b toward a . The transition rule in this case is $H_a \{x\}_k \Rightarrow S_{a,b} k$, stating that agent a , on observing $\{x\}_k$, learns that agent b knows the right value of the key k .

The semantics for the language \mathcal{L}_C , provided in the next definition, follows the set-up of [BMS99, Dit00]

Definition 4.2. Let \mathcal{C} be a set of transition rules. For $\sigma \in \mathcal{L}_C$ the relation $\llbracket \sigma \rrbracket$ on models

for \mathcal{A} over \mathcal{P} is given by

$$\begin{aligned}
(M, w) \llbracket \text{Priv}(i \rightarrow j, p) \rrbracket (M', w') &\Leftrightarrow (M, w) \models B_i p \Rightarrow (\text{EXPAND}_{p,j}(M, w) \triangleright_p (M', w')) \\
(M, w) \llbracket \text{Pub}(i, p) \rrbracket (M', w') &\Leftrightarrow (M, w) \models B_i p \Rightarrow (\text{EXPAND}_{p,\mathcal{A}}(M, w) \triangleright_p (M', w')) \\
(M, w) \llbracket \sigma; \sigma' \rrbracket (M', w') &\Leftrightarrow (M, w) \llbracket \sigma \rrbracket (M'', w'') \llbracket \sigma' \rrbracket (M', w') \text{ for some model } (M'', w'') \\
(M, w) \triangleright_p (M', w') &\Leftrightarrow \text{if } (x \Rightarrow \beta) \in \text{Sel}(M, w, p) \\
&\quad \text{then } (M, w) \langle \beta \rangle (M'', w'') \triangleright_p (M', w') \text{ for some } (M'', w'') \\
&\quad \text{else } (M, w) = (M', w') \\
&\quad \text{end} \\
(M, w) \langle \rangle (M', w') &\Leftrightarrow (M, w) = (M', w') \\
(M, w) \langle L_{\mathcal{B}} p; \beta \rangle (M', w') &\Leftrightarrow \text{EXPAND}_{(p,\mathcal{B})}(M, w) \langle \beta \rangle (M', w') \\
(M, w) \langle S_{i,j} p; \beta \rangle (M', w') &\Leftrightarrow \text{SIDE-EFFECT}_{(p,i,j)}(M, w) \langle \beta \rangle (M', w') \\
(M, w) \models p &\Leftrightarrow \pi(w)(p) = \text{true} \\
(M, w) \models \neg \phi &\Leftrightarrow (M, w) \not\models \phi \\
(M, w) \models \phi \wedge \psi &\Leftrightarrow (M, w) \models \phi \text{ and } (M, w) \models \psi \\
(M, w) \models B_i \phi &\Leftrightarrow (M, v) \models \phi \text{ for all } v \text{ such that } w R_i v \\
(M, w) \models [\sigma] \phi &\Leftrightarrow (M', w') \models \phi \text{ if } (M, w) \llbracket \sigma \rrbracket (M', w')
\end{aligned}$$

where

$$\begin{aligned}
\text{Sel}(M, w, p) &= \{ B_i p \Rightarrow \beta \in \mathcal{C} \mid (M, w) \langle \beta \rangle (M', w'), (M', w') \models \phi \not\Leftarrow (M, w) \models \phi, (M, w) \models B_i p \} \\
&\cup \{ H_i p \Rightarrow \beta \in \mathcal{C} \mid (M, w) \langle \beta \rangle (M', w'), (M', w') \models \phi \Leftarrow (M, w) \models \phi \}
\end{aligned}$$

The operation \triangleright_p applies a number of transition rules after the execution of a communication action. For the transition rules of type $B_i p \Rightarrow \beta$ it is checked if the precondition $B_i p$ holds.

The selection $\text{Sel}(M, w)$ is organized in such a way that no transition rule is applied over and over again. The recursive definition of \triangleright therefore stops if no fresh transition rule can be applied. In the definition of Sel it is checked if the belief of the agents changes under the transition rules, preventing an infinite chain of rewrites for \triangleright_p . Note that because of the results of the previous section, the order of applying these transition rules does not matter.

5 The SRA Three Pass protocol

In this section we discuss how the machinery developed above works out for a concrete example. Preparatory for this, in order to keep the models within reasonable size, we employ two helpful tricks. The first one is the disregarding of propositions not known to any agent. Thus, if a proposition is not part of the model, then the interpretation is that no agent has any knowledge about it. What we have to specify is how we add that proposition into the model. We accomplish this by making two copies of the original states. One of them we assign ‘positive’ and the other ‘negative’. In the positive states, the proposition will be **true**, and in the negative states, the proposition will be **false**.

Definition 5.1. Given a model $(M, w) = \langle S, \pi, R_1, \dots, R_m \rangle$ we define the function ADDATOM_p such that $(M', w') = \text{ADDATOM}_p(M, w) = \langle S', \pi', R'_1, \dots, R'_m \rangle$ where

- $S' = \text{pos}(S) \cup \text{neg}(S)$

- $\pi'(pos(s))(q) = \text{if } p = q \text{ then true else } \pi(s)(q)$
- $\pi'(neg(s))(q) = \text{if } p = q \text{ then false else } \pi(s)(q)$
- $R'_i(\alpha(s), \beta(t)) \Leftrightarrow R_i(s, t), \alpha, \beta = pos, neg$
- $w' = pos(w)$

We have the following property.

Lemma 5.1. *Given a model (M, w) and $(M', w') = \text{ADDTOM}_p(M, w)$ it holds that i) $(M', w') \models p$, ii) $(M, w) \models \phi \Leftrightarrow (M', w') \models \phi$ for $p \notin \phi^*$ with ϕ^* the closure under subformulas of ϕ and iii) for all $i \in \mathcal{A}$: $(M', w') \not\models B_i p$.*

The second trick helps to prevent the useless applying of rules which keeps the model in a reasonable size.

Lemma 5.2. *Given a model (M, w) , the model (M', w') such that $(M, w) \langle L_i p; S_{j_i} p \rangle x [Pub(i, p)] (M', w')$ (for some x, i, j) and the model (M'', w'') such that $(M, w) \langle L_{\mathcal{A}} p \rangle (M'', w'')$ are bisimilar.*

That is to say, if an agent i learns p and then all other agents learn about i that it has learned p , followed by the action where everyone learns p (commonly), then it is equivalent to say that they have just learned p commonly.

Shamir, Rivest and Adelman have suggested the three-pass protocol [CJ97] for the transmission a message under minimal assumptions for commutative encryption. It is known to be insecure and various attacks have been suggested. However, it serves an illustrative purpose here. The protocol has the following steps:

1. $a \rightarrow b : \{x\}_{k_a}$
2. $b \rightarrow a : \{\{x\}_{k_a}\}_{k_b}$
3. $a \rightarrow b : \{x\}_{k_b}$

Both agent a and b have their own encryption key, k_a and k_b , respectively. The encryption key can be a symmetric key or a the private key of a private key-public key-pair for that matter. Agent a wants to send message x to agent b through an insecure channel and therefore wants to send x encrypted to b . It does this by sending x encrypted with its own key. Next, b will encrypt this message with b 's key and sends this back. Since the encryption is commutative, a can now decrypt this message and sends this to b . Finally, b can decrypt the message it has just received and learn the value of x .

We consider three agents $\{a, b, i\}$ where a and b are honest agents that will run this protocol and i is the intruder that looks at the messages that are being transmitted through the network. So we're interested, if i does not actively attack the protocol, what i can learn during the run of this protocol.

Next, we define the transition rules. The first transition rule models the fact that agents can encrypt with their own key: $B_j m_{\mathcal{K}} \Rightarrow L_j m_{\mathcal{K}+j}; S_{ij} m_{\mathcal{K}+j}$ and, for $j \in \mathcal{K}$, $B_j m_{\mathcal{K}} \Rightarrow L_j m_{\mathcal{K}-j}; S_{ij} m_{\mathcal{K}-j}$, for all $j \in \mathcal{A}$, where \mathcal{K} is some set of agents, $+$ a function that adds an agent and $-$ a function that deletes one. Here, $m_{\mathcal{K}}$ represent a message successively encrypted with the keys of the agents in \mathcal{K} . In the modeling, we limit ourselves by defining the list

of useful propositions. The propositions we want to consider here are $\mathcal{P} = \{m, m_a, m_b, m_{ab}\}$ where m_a abbreviates $\{x\}_{k_a} = \llbracket \{x\}_{k_a} \rrbracket$ and m_{ab} abbreviates $\{\{x\}_{k_a}\}_{k_b} = \llbracket \{\{x\}_{k_a}\}_{k_b} \rrbracket$.

We assume that

- encryption is commutative;
- every agent has its own key not possessed by any other agent;
- during the run of the protocol, the value of the keys do not change.

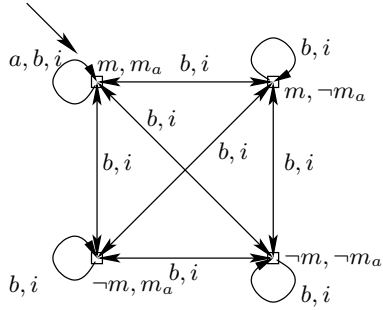


Figure 11: Starting point

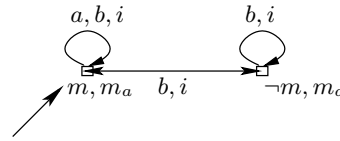


Figure 12: after $Pub(a, m_a)$

The next assumption we must make is about the knowledge state of the agents *before* the run of the protocol. We will assume that a is the only agent that knows m and m_a . Furthermore, we will assume that the other agents know this about a . The corresponding Kripke structure is in Figure 11.

The first step is executed. That is, m_a is propagated on the network, so all agents will learn this value. Thus, we execute the action $Pub(a, m_a)$. If we discard the states that become unreachable, this results in the model of Figure 12. Note that this model models $B_b m_a$. This triggers one of the transition rules, that is, it triggers $B_b m_a \Rightarrow L_b m_{ab}; S_{ib} m_{ab}$ since the antecedent holds in the point now.

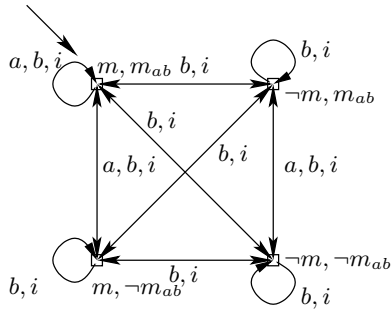


Figure 13: added m_{ab}

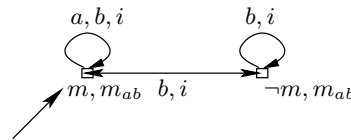


Figure 14: after $Pub(b, m_{ab})$

We have not modelled m_{ab} yet, so this is the first step. We will not repeat m_a in the figure since this holds in any state of the model. The function $ADDATOM_{m_{ab}}$ results in the model of Figure 13. Observe that in the next step of the protocol $L_{\mathcal{A}} m_{ab}$ ($Pub(b, m_{ab})$) is executed, since the message is being transmitted to all agents on the network. So with Lemma 5.2 it is justified to skip the steps that are required by the transition rules. Thus, we get the model in

Figure 14. This results in the triggering of the transition rule: $B_a m_{ab} \Rightarrow L_a m_b; S_{ia} m_b$. This is in fact a completely similar case as in the previous step of the protocol. Again we dismiss the m_{ab} proposition since every agent has learned this.

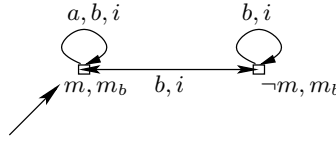


Figure 15: after $Pub(a, m_b)$

We introduce m_b and execute $Pub(a, m_b)$ because we can again skip the actions in the transition rules (Lemma 5.2). So we end up with the model in Figure 15. The last transition rule that is triggered is $B_b m_b \Rightarrow L_b m; S_{ib} m$. Again, we discard the proposition that holds in every state: m_b . We now only focus on the most interesting proposition m . First b learns m which results in the model in Figure 16.

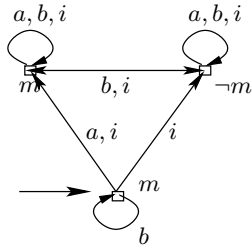


Figure 16: after $L_b m$

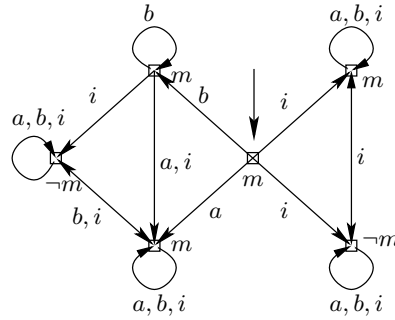


Figure 17: after $S_{ib} m$

The second action for the transition rule is that i learns that $B_b p \vee B_b \neg p$. If we execute this on the model we get the model (M', w') which can be seen in Figure 17. Recall that we have $(M', w') = [Pub(a, m_a); Pub(b, m_{ab}); Pub(a, m_b)](M, w)$. It holds that: $(M', w') \models \neg B_i m$, $(M', w') \models B_b m$ and $(M', w') \models B_i((B_b m \vee B_b \neg m) \wedge B_a(B_b m \vee B_b \neg m))$.

6 Conclusion

Inspired by recent work on dynamic epistemic logics, we have proposed a logical language for describing (properties of) runs of security protocols. The language contains constructs for the two basic types of epistemic actions that happen during such runs. The semantics of the language is based on traditional Kripke models representing the epistemic state of the agents involved in the protocol at hand. Changes in the epistemic state of the agent system as a result of the execution of a protocol are described by means of transition rules that precisely indicate what belief updates happen under certain preconditions. These belief updates give rise to modifications of the models representing the agents' epistemic state in a way that is precisely given by semantic operations on these models. We have illustrated our approach for a well-known security protocols such as the SRA Three Pass protocol. We also have analyzed

the Needham-Schröder public key protocol and Andrew RPC, see [Hom03]. It should be noted, that we focus here on a single protocol runs with passive intruder. A further research goal is to extend our approach to deal with a setting of multiple protocols/multiple runs and active intruder.

The semantic updates we used operate on traditional Kripke models as opposed to updates in the approaches of Gerbrandy and Baltag. We believe that this will make it less troublesome to integrate these updates into existing model checkers, which hopefully will lead to better and new tools for verifying properties of security protocols.

Although future research will have to justify this, we are confident that our method can be employed for a broad class of verification problems concerning security protocols because of the flexibility of our approach using transition rules for epistemic updates.

References

- [AHV01] N. Agray, W. van der Hoek, and E.P. de Vink. On BAN logics for industrial security protocols. In B. Dunin-Keplicz and E. Nawarecki, editors, *From Theory to Practice in Multi-Agent Systems*, pages 29–38. LNAI 2296, 2001.
- [And01] R.J. Anderson. *Security Engineering: A Guide to Building Dependable Distributed Systems*. Wiley, 2001.
- [AT91] M. Abadi and M. Tuttle. A semantics for a logic of authentication. In *Proc. PODC'91*, pages 201–216. ACM, 1991.
- [Bal00] A. Baltag. A logic for suspicious players: epistemic actions and belief-updates in games. Technical Report SEN-0044, CWI, 2000.
- [BAN90] M. Burrows, M. Abadi, and R.M. Needham. A logic of authentication. *ACM Transactions on Computer Systems*, 8:16–36, 1990.
- [BM97] A. Bleeker and L. Meertens. A semantics for BAN logic. In *Proceedings DIMACS Workshop on Design and Formal Verification of Protocols*. DIMACS, Rutgers University, <http://dimacs.rutgers.edu/Workshops/Security>, 1997.
- [BMS99] A. Baltag, L.S. Moss, and S. Solecki. The logic of public announcements, common knowledge and private suspicions. *Annals of Pure and Applied Logic*, 1999.
- [CJ97] J.A. Clark and J.L. Jacob. A survey of authentication protocols 1.0. Technical report, University of York, 1997.
- [Dit00] H.P. van Ditmarsch. *Knowledge games*. PhD thesis, Universiteit van Amsterdam, 2000.
- [Dit01] H.P. van Ditmarsch. The semantics of concurrent knowledge actions. In M. Pauly and G. Sandu, editors, *Proc. ESSLLI Workshop on Logic and Games*, Helsinki, 2001.
- [DY83] D. Dolev and A.C. Yao. On the security of public-key protocols. *IEEE Transaction on Information Theory*, 29:198–208, 1983.

- [Ger97] J. Gerbrandy. Dynamic epistemic logic. Technical Report LP-97-04, ILLC, 1997.
- [Ger99] J. Gerbrandy. *Bisimulations on Planet Kripke*. PhD thesis, University of Amsterdam, 1999.
- [Hom03] A.J. Hommersom. Reasoning about security. Master's thesis, Universiteit Utrecht, 2003.
- [KN98] V. Kessler and H. Neumann. A sound logic for analyzing electronic commerce protocols. In J.-J. Quisquater, Y. Deswarte, C. Meadows, and D. Gollman, editors, *Proc. ESORICS'98*, pages 345–360. LNCS 1485, 1998.
- [Koo03] B. Kooi. *Knowledge, Chance, and Change*. PhD thesis, Rijksuniversiteit Groningen, 2003.
- [RHM02] J.-W. Roorda, W. van der Hoek, and J.-J.Ch. Meyer. Iterated belief change in multi-agent systems. *Proceedings of the first International Joint Conference on Autonomous Agents and Multi-Agent Systems: Part 2*, 2002.
- [SC01] P. Syverson and I. Cervesato. The logic of authentication protocols. In R. Foccardi and R. Gorrieri, editors, *Foundations of Security Analysis and Design: Tutorial Lectures*. LNCS 2171, 2001.
- [Sch00] B. Schneier. *Secrets and Lies: Digital Security in a Networked World*. Wiley, 2000.
- [SW02] S.G. Stubblebine and R.N. Wright. An authentication logic with formal semantics supporting synchronization, revocation and recency. *IEEE Transactions on Software Engineering*, 28:256–285, 2002.
- [WK96] G. Wedel and V. Kessler. Formal semantics for authentication logics. In E. Bertino, H. Kurth, G. Martello, and E. Montolivo, editors, *Proc. ESORICS'96*, pages 219–241. LNCS 1146, 1996.