Toward Reasoning about Security Protocols: A Semantic Approach

Arjen Hommersom[1]

*Nijmegen Institute of Information and Computing Sciences
Radboud University, Nijmegen, the Netherlands*

John-Jules Meyer

*Institute of Information and Computer Science
Utrecht University, the Netherlands*

Erik de Vink

*Department of Mathematics and Computer Science
Technische Universiteit Eindhoven, the Netherlands*

*Leiden Institute of Advanced Computer Science
Leiden University, the Netherlands*

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 them 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.

Keywords: modal logic, belief revision, semantic updates, security protocols, dynamic epistemic logic

[1] Corresponding author: arjenh@cs.ru.nl

1   Introduction

In today's world of e-commerce and the Internet, the role of security protocols is getting increasingly important. The design of security protocols is difficult and error-prone [20,2], which makes (automated) 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 [5]. 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., [16,1,21]), due to their complexity and their semantic underpinning the use of BAN logics to prove the correctness of security protocols has so been of limited success (cf. [3,6,22,19]).

In this paper we will apply insights from dynamic epistemic logics as recently developed by Gerbrandy [13,14], Baltag and Moss [8,4,7], Van Ditmarsch [10,11], and Kooi [17]. 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 and Moss' 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 the more involved action models. To this end we need also operators for unfolding models, which is in 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 tool-supported verification 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 observing 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 the notion of an objective formula and the concept of o-seriality.

Definition 2.1 Fix a set $\mathcal{P}$ of propositional variables, also referred to as atoms. The class of objective formulas is the smallest class such that

- all propositional variables $p \in \mathcal{P}$ 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 Kripke model $M = \langle S, \pi, R_1, ..., R_m \rangle$ 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 $(M, s) = (\langle S, \pi, R_1, ..., R_m \rangle, s)$, where $(M, s) \models B_i\phi$ iff $\forall t \in S \colon R_i(s, t) \rightarrow (M, t) \models \phi$. The state s is referred to as the actual world.

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

(t) $$B_i\phi \Rightarrow \phi,$$

for every objective ϕ. The system Kt45 is sound with respect to the class of o-serial, transitive and euclidian models [15]. 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 $[\![p]\!]$. In the real world it is, obviously, always true that $p = [\![p]\!]$. 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 $[\![\cdot]\!]$ is given, that provides the 'real' values of the program variables when needed. It might very well be the case that $p \neq [\![p]\!]$ in a certain state. From now on, we will abbreviate $p = [\![p]\!]$ to p. (Thus transforming a program expression into a propositional variable.) Similarly, $\neg p$ is an abbreviation of $p \neq [\![p]\!]$. 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 to assume the Dolev-Yao framework [12], in which 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 [18,8]. 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 $(M, w) = (\langle S, \pi, R_1, ..., R_m \rangle, w)$, a group of agents $\mathcal{B}$, and an objective formula ϕ be given such that $(M, w) \models \phi$. Then

$$\text{UPDATE}_{(\phi, \mathcal{B})}(M, w) = (\langle S', \pi', R'_1, ..., R'_m \rangle, w')$$

where

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

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

Note that ϕ holds in each state in the *new* part. The following example shows how this works on a concrete model.

Example 3.2 Consider the model (M, s) in Figure 1a where $\pi(s)(p) = \texttt{true}$ and $\pi(t)(p) = \texttt{false}$. The operation we execute is that b learns p, i.e. $\text{UPDATE}_{(p,b)}$. This results in the model (M, u) in Figure 1b where $new(s) = u$, $old(s) = v$ and $old(t) = w$ and $\pi(u)(p) = \pi(v)(p) = \texttt{true}$ and $\pi(w)(p) = \texttt{false}$. The world $new(t)$ 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, hence b beliefs p.

The update $\text{UPDATE}_{(\phi, \mathcal{B})}$ is based on a formula ϕ and a set of agents $\mathcal{B}$. Roorda *et al.* [18] 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.
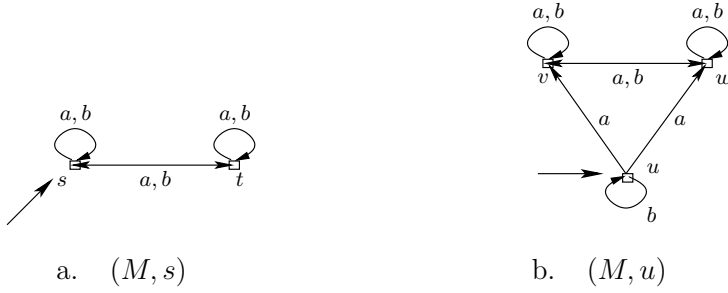
a. (M, s) b. (M, u)

Fig. 1.

Definition 3.3 An update function $(\cdot)[\phi, \mathcal{B}]$ is called proper if

$$(M, w)[\phi, \mathcal{B}] \models p \qquad \Leftrightarrow \quad (M, w) \models p$$

$$(M, w)[\phi, \mathcal{B}] \models \alpha \wedge \beta \quad \Leftrightarrow \quad (M, w)[\phi, \mathcal{B}] \models \alpha \text{ and } (M, w)[\phi, \mathcal{B}] \models \beta$$

$$(M, w)[\phi, \mathcal{B}] \models \neg\alpha \qquad \Leftrightarrow \quad (M, w)[\phi, \mathcal{B}] \not\models \alpha$$

$$(M, w)[\phi, \mathcal{B}] \models B_a\alpha \quad \Leftrightarrow \quad (M, w) \models B_a\alpha \text{ if } a \notin \mathcal{B}$$

$$(M, w)[\phi, \mathcal{B}] \models B_b\alpha \quad \Leftrightarrow \quad B_b(w, u) \wedge (M, u) \models \phi) \text{ implies}$$

$$(M, u)[\phi, \mathcal{B}] \models \alpha) \text{ if } b \in \mathcal{B}$$

Following Roorda *et al.* we have that UPDATE$_{(\phi, \mathcal{B})}$ is proper. Moreover, UPDATE$_{(\phi, \mathcal{B})}$ is uniquely characterized by Definition 3.3 upto elementary equivalence, i.e. if $(\cdot)[\phi, \mathcal{B}]$ is a proper update function, then $(M, w)[\phi, \mathcal{B}]$ and UPDATE$_{(\phi, \mathcal{B})}$ are elementary equivalent.

We collect the following properties of UPDATE$_{(\phi, \mathcal{B})}$.

Lemma 3.4

(a) *For all formula ϕ it holds that $(M, w) \models \phi \Rightarrow$ UPDATE$_{(\phi, \mathcal{B})}(M, w) \models B_\mathcal{B}\phi$.*

(b) *If a model (M, w) satisfies the Kt45 properties and ϕ is objective, then the model UPDATE$_{(\phi, \mathcal{B})}(M, w)$ satisfies the Kt45 properties as well.*

(c) *The models*

$$\text{UPDATE}_{(\psi, \mathcal{C})}\big(\text{UPDATE}_{(\phi, \mathcal{B})}(M, w)\big) \text{ and } \text{UPDATE}_{(\phi, \mathcal{B})}\big(\text{UPDATE}_{(\psi, \mathcal{C})}(M, w)\big)$$

are bisimilar.

Note that in Lemma 3.4a ϕ ranges over arbitrary formulas, including non-objective ones. However, for a non-objective formula, $B_i\phi$ say, it can happen that, unintended, an agent increases the objective knowledge encapsulated by the formula ϕ. This is illustrated by the next example.
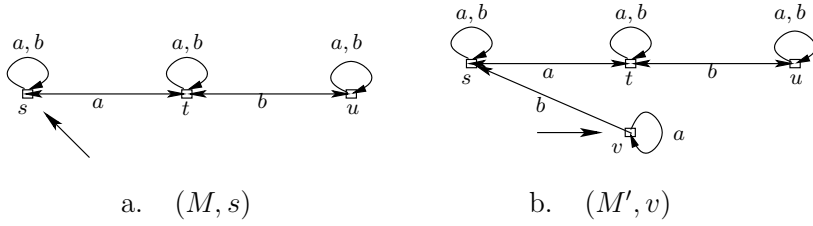
Fig. 2.

Example 3.5 Suppose we are interested in agent a learning the formula $B_b p \lor B_b \neg p$, but not p itself. Consider the Kripke model (M, s) in Figure 2a, where $\pi(s)(p) = \pi(u)(p) = \mathtt{true}$ and $\pi(t)(p) = \mathtt{false}$. This models that 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 update operation, it results in the model (M, v) from Figure 2b, where $\pi(v)(p) = \pi(s)(p) = \pi(u)(p) = \mathtt{true}$ and $\pi(t)(p) = \mathtt{false}$. The reason that it turns out like this, is because the only state where $B_b p \lor B_b \neg p$ holds, is the state s. Thus, all the other states have no corresponding *new* states that are reachable from the new point v. Figure 2b illustrates that a has learned $B_b p \lor B_b \neg p$, but also that a has learned p itself.

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

3.2   Side-effects

The main reason that in Example 3.5 the update of $B_b p \lor B_b \neg p$ for agent a fails, is that it actually deletes the wrong arrows. The update operation 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 $B_b p \lor B_b \neg p$ 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 to 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

at the point of the model, i.e. the actual world.

Definition 3.6 Given a model (M, w) with $M = \langle S, \pi, R_1, ..., R_m \rangle$, a partitioning $\mathcal{X}$ of the set of agents $\mathcal{A}$, we define the operation $\textsc{unfold}_{\mathcal{X}}(M, w) = (\langle S', \pi', R_1', ..., R_m' \rangle, w')$, where

- $S' = \left(\bigcup_{\mathcal{B} \in \mathcal{X}} new_{\mathcal{B}}(S) \right) \cup orig(w)$
- $w' = orig(w)$
- $\pi'(new_{\mathcal{B}}(v))(p) = \pi(v)(p)$, $\pi'(w')(p) = \texttt{true}$ for all $p \in \mathcal{P}, \mathcal{B} \in \mathcal{X}$
- the relation R_i' on S', $1 \le i \le m$, is minimal such that

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

$$R_i'(orig(w), new_{\mathcal{B}}(u)) \quad \Leftrightarrow \quad R_i(w, u) \wedge i \in \mathcal{B}$$

  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 operation does indeed preserve our Kt45 properties and it models the same knowledge, which is captured by the following lemma.
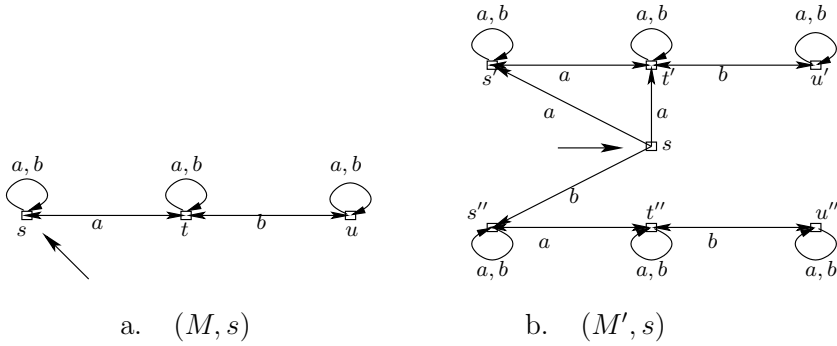
Fig. 3.

Lemma 3.7

(a) *If (M, w) is a Kt45 model and $\mathcal{X}$ a partition of ma, then* UNFOLD$_\mathcal{X}(M, w)$ *is a Kt45 model as well.*

(b) *For every model (M, w) and partition $\mathcal{X}$ of $\mathcal{A}$, it holds that the models (M, w) and* UNFOLD$_\mathcal{X}(M, w)$ *are bisimilar.*

Example 3.8 Consider the Kripke model (M, s) in Figure 3.a with $\pi(s)(p) = \pi(u)(p) = \mathtt{true}, \pi(t)(p) = \mathtt{false}$. 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 3b (with valuation as expected).

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.

Next, we give some preparatory definitions leading to the notion of a side-effect operation in Defintion 3.14. First we define the notion of a submodel.

Definition 3.9 A model $M = \langle S, \pi, R_1, ..., R_m \rangle$ is a submodel of $M' = \langle S', \pi', R'_1, ..., R'_m \rangle$, 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 $R_i \subseteq R'_i$.

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

Definition 3.10 Given a model $(M, w) = (\langle S, \pi, R_1, ..., R_m \rangle, w)$ and a partition $\{\{a\}, \mathcal{B}\}$ of $\mathcal{A}$ such that $(M, w) = $ UNFOLD$_{\{\{a\}, \mathcal{B}\}}(M', w')$ such that $\{\{a\}, \mathcal{B}\}$ for some model (M', w'), the a-submodel SUB$_a(M)$ of M for a is given by $(M', w') = \langle S', \pi', R'_1, ..., R'_m \rangle$ where
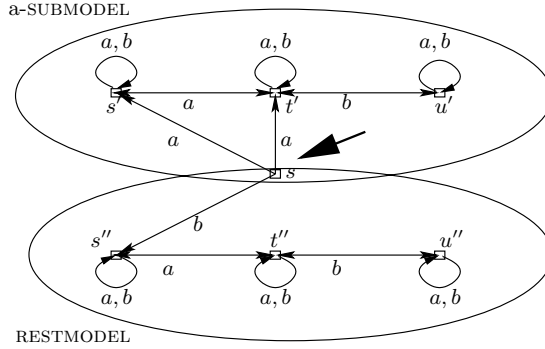
Fig. 4. a-submodel outline

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

Clearly an *a*-submodel is a submodel in the sense of Definition 3.9. The restmodel is the part of the model that is complements the submodel with respect to the accessibility relation.

Definition 3.11 Given a model $(M,w) = \langle S, \pi, R_1, ..., R_m \rangle$ and a submodel $N = \langle S'', \pi'', R''_1, ..., R''_m \rangle$ of M, the restmodel $\text{REST}_N(M)$ of M with respect to N is given by $\text{REST}_N(M) = \langle S', \pi', R'_1, ..., R'_m \rangle$ 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 3.8 and applying the above definitions. See Figure 4. 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. For this, we introduce the replace operation.

Definition 3.12 Given a model $N = \langle S, \pi, R_1, ..., R_m \rangle$, a model M, a submodel N' such that $N \sqsubseteq N' \sqsubseteq M$ with $\text{REST}_{N'}(M) = \langle S', \pi', R'_1, ..., R'_m \rangle$, we define $\text{REPLACE}_{N'}(N, M) = \langle S'', \pi'', R''_1, ..., R''_m \rangle$ where

- $s \in S'' \Leftrightarrow s \in S \vee s \in S'$
- $\pi''(s)(p) = \pi(s)(p)$ for $s \in S''$ for all $p \in \mathcal{P}$
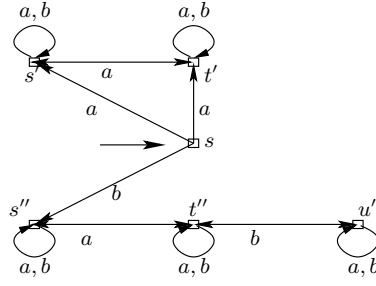
Fig. 5. (M'', s)

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

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. In particular, we change locally the knowledge of an agent a, e.g. regarding uncertainty of another agent b about a propositin p. The operation ATOMSPLIT$_{(p,b)}$ below removes the arrows of b between states that have a different valuation for p.

Definition 3.13 Given a model $M = \langle S, \pi, R_1, ..., R_m \rangle$, we define a function ATOMSPLIT$_{(p,b)}(M) = \langle S', \pi', R_1', ..., R_m' \rangle$ as follows:

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

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

Definition 3.14 For a model (M, w) put $(M', w') = $ UNFOLD$_{\{\{a\}, \mathcal{A} \backslash \{a\}\}}(M, w)$ and $N = $ SUB$_a(M')$. The model SIDE-EFFECT$_{(p,a,b)}(M, w)$, the side-effect for a learning b knowing p, is given by

$$\text{SIDE-EFFECT}_{(p,a,b)}(M, w) = (\text{REPLACE}_N(\text{ATOMSPLIT}_{p,b}(N), M'), w').$$

Example 3.15 We continue Example 3.8. To the a-submodel of M we now apply ATOMSPLIT$_{p,b}$ which results in the model (M'', s) in Figure 5. The arrow (t', u') has disappeared, since $\pi(t')(p) \neq \pi(u')(p)$. Therefore, u' is not reachable anymore from the actual world, and can be dropped. Notice that a believes $B_b p \lor B_b \neg p$, while a has learned nothing about p itself.

We have the following results.

Lemma 3.16

(a) If (M, w) is a Kt45-model, then SIDE-EFFECT$_{(p,a,b)}(M, w)$ *is a Kt45-model as well.*

(b) Given a model (M, w), agents a, b, c, d, two propositions $p, q \in \mathcal{P}$, it holds that the models

$$\text{SIDE-EFFECT}_{(p,c,d)}\big(\text{SIDE-EFFECT}_{(q,a,b)}(M, w)\big)$$

and

$$\text{SIDE-EFFECT}_{(q,a,b)}\big(\text{SIDE-EFFECT}_{(p,c,d)}(M, w)\big)$$

are bisimilar.

(c) Given a model (M, w), agents a, b, c, a proposition $p \in \mathcal{P}$ and an objective formula ϕ, it holds that the models

$$\text{SIDE-EFFECT}_{(p,c,d)}\big(\text{UPDATE}_{(\phi,a)}(M, w)\big)$$

and

$$\text{UPDATE}_{(\phi,a)}\big(\text{SIDE-EFFECT}_{(p,c,d)}(M, w)\big)$$

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:

 (i) the group of agents that learn about other agents, ranged over by a;

 (ii) the group of agents that is learned about, ranged over by b;

(iii) other agents, ranged over by c.

The fact that the agents of type a are the only agents 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 3.17 *Let a, b and c be three different agents. Given a model (M, w) and the model $(M', w') = $ 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 $(M, w) \models B_a \phi$ for 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 $i \neq a$.

Part (a) of the above lemma states that 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 in the context of open communication networks (as is the setting of Sections 4 and 5). 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.

We show how to refine the approach above by distinguishing between agents that are considered to have equal learning facilities and thos that do not. I.e., we split the former group of other agents in those that commonly learn and thos that will remain ignorant. We represent this by linking a's beliefs of those other agents back to the original (unmodified) states. We now distinguish four different type of groups of agents.

(i) the group A of agents that learn about other agents, ranged over by a;

(ii) the group B of agents that is learned about, ranged over by b;

(iii) the group C of agents of which agents a believe they have commonly learned about agents in group b with, ranged over by c;

(iv) 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 partinioning 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 3.18 Given a model (M, w) such that $M = \langle S, \pi, R_a, ..., R_d \rangle$, we define a function 0-UNFOLD$(M, w) = (\langle S', \pi', R'_a, ..., R'_d \rangle, w')$, where

- $S' = new_a(S) \cup new_{bcd}(S) \cup orig(w)$

- $w' = orig(w)$

- $\pi'(new_{\mathcal{B}}(v))(p) = \pi(v)(p)$ and $\pi'(orig(w))(p) = \pi(w)(p)$ for all $p \in \mathcal{P}$, $\mathcal{B} \in \{\{a\}, \{bcd\}\}$

- R'_i on S', $i \in \{a, b, c, d\}$, is minimal such that

$$R'_d(new_a(u), new_{bcd}(v)) \quad \Leftrightarrow \quad R_d(u,v)$$

$$R'_i(new_a(u), new_a(v)) \quad \Leftrightarrow \quad R_i(u,v) \ (i \neq d)$$

$$R'_i(new_{bcd}(u), new_{bcd}(v)) \quad \Leftrightarrow \quad R_i(u,v)$$

$$R'_a(orig(w), new_a(v)) \quad \Leftrightarrow \quad R_a(w,v)$$

$$R'_i(orig(w), new_{bcd}(v)) \quad \Leftrightarrow \quad R_i(w,v) \wedge i = b,c,d$$

So instead of completely separating the knowledge of te 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 3.7.

Lemma 3.19

(a) If (M, w) is a Kt45 model, then so is 0-UNFOLD(M, w).

(b) For every model (M, w) it holds that the models (M, w) and 0-UNFOLD(M, w) are bisimilar.

Due to the case distinction for $R'_a(orig(w), new_{\mathcal{B}}(v))$ in Definition 3.18 above, it holds that the knowledge of a about b is not related to the knowledge of other agents about b or of b itself. So, as before, we can 'cut out' the submodel containing the b arrows from the belief of a:

Definition 3.20 Given a model $(M', w') = \langle S', \pi', R'_a, ..., R'_d \rangle$ such that $(M', w') =$ 0-UNFOLD(M, w), for some (M, w), define B-SUB$(M') = \langle S'', \pi'', R''_a, ..., R''_d \rangle$ where

- $S'' = \{ new_a(s) \mid s \in S \}$
- $\pi''(s)(p) \Leftrightarrow \pi(s)(p)$ 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$_p$ is then given by

$$\text{0-SIDE-EFFECT}_p(M, w) = (\text{REPLACE}_N(\text{ATOMSPLIT}_{(p,b)}(N), M'), w')$$

where $N = \text{B-SUB}(M')$.

Again, the operation respect the Kt45 -properties.

Lemma 3.21 *If (M, w) is a Kt45-model, then so is the model* 0-SIDE-EFFECT$_p(M, w)$.

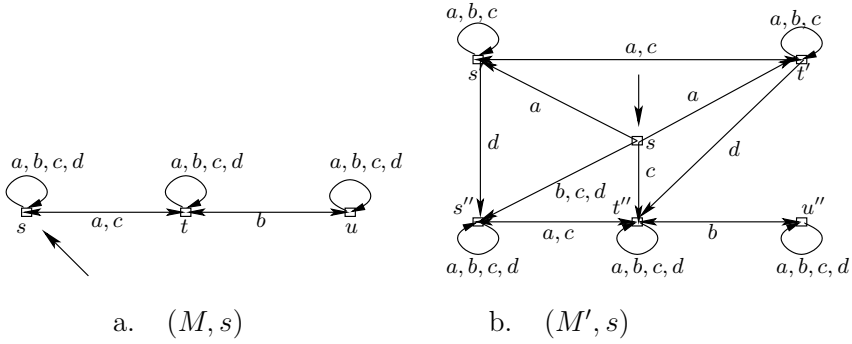a. (M, s) b. (M', s)

Fig. 6.

Similarly to Lemma 3.16 and Lemma 3.17 we have the following result.

Lemma 3.22

(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) $(M', w') \models B_a(B_b p \lor B_b \neg p)$

(ii) $(M', w') \models B_a \phi$ iff $(M, w) \models B_a \phi$ for ϕ objective

(iii) $(M', w') \models B_a C_{abc}(B_b p \lor B_b \neg p)$

(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) compare to the properties given in Lemma 3.17. Part (v) states that the knowledge of agent a about agent d has not changed, exactly as desired.

Example 3.23 Recall the model (M, s) from Example 3.8 (Figure 3.a). We now present this model with four agents $\{a, b, c, d\}$ in Figure 6.a such that $\pi(s)(p) = \pi(u)(p) = \texttt{true}$ and $\pi(t)(p) = \texttt{false}$. Now, we apply $0\text{-SIDE-EFFECT}_p(M, w)$ and we obtain the model (M', s) from Figure 6.b such that $\pi(s)(p) = \pi(s')(p) = \pi(s'')(p) = \pi(u'')(p) = \texttt{true}$, $\pi(t')(p) = \pi(t'')(p) = \texttt{false}$. Note that in the latter model, in agreement with clause (v) of Lemma 3.22b a still knows exactly the same about d as it did before.

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. The usage of the language is illustrated for the

case of the SRA Three Pass protocol. We describe 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

$$\phi ::= p \mid \neg\phi \mid \phi_1 \wedge \phi_2 \mid B_i\phi \mid [\sigma]\phi$$

$$\sigma ::= Priv(i \rightarrow j, p) \mid Pub(i, p) \mid \sigma; \sigma'$$

The σ symbol denotes a (possibly composed) communication action. The action $Priv(i \rightarrow j, p)$ is a private or peer-to-peer message p from i to j; the action $Pub(i, p)$ 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 updates and side-effects, necessary for the interpretation of the constructs $Priv(i \rightarrow j, p)$ and $Pub(i, p)$. The transitions rules enforce consistency among the propositions that hold. For example, if an agent believes that the value of a message m is $[\![m]\!]$ 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 $[\![m]\!]$.

A transition rule has either the form $B_ip \Rightarrow \beta$ or $H_ip \Rightarrow \beta$. B_ip and H_ip 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 $\alpha_1; \ldots; \alpha_n$. Actions come in two flavours, viz. $L_{\mathcal{B}}p$ and $S_{i,j}p$. Here, an action $L_{\mathcal{B}}p$ expresses that p is learned among the agents in the set $\mathcal{B}$ and corresponds to belief update, whereas an action $S_{a,b}p$ 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 $H_ip \Rightarrow \beta$ 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, the agent a 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 [8,10]

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

- $(M, w) \llbracket Priv(i \rightarrow j, p) \rrbracket (M', w')$

    $\Leftrightarrow (M, w) \models B_i p \Rightarrow (\text{EXPAND}_{p,j}(M, w) \rhd_p (M', w'))$

  $(M, w) \llbracket Pub(i, p) \rrbracket (M', w')$

    $\Leftrightarrow (M, w) \models B_i p \Rightarrow (\text{EXPAND}_{p,\mathcal{A}}(M, w) \rhd_p (M', w'))$

  $(M, w) \llbracket \sigma; \sigma' \rrbracket (M', w')$

    $\Leftrightarrow (M, w) \llbracket \sigma \rrbracket (M'', w'') \llbracket \sigma' \rrbracket (M', w')$ for some model (M'', w'')

- $(M, w) \rhd_p (M', w') \Leftrightarrow$

    if $(x \Rightarrow \beta) \in Mod(M, w, p)$

    then $(M, w) \langle \beta \rangle (M'', w'') \rhd_p (M', w')$ for some (M'', w'')

    else $(M, w) = (M', w')$

- $(M, w) \langle \rangle (M', w') \Leftrightarrow (M, w) = (M', w')$

  $(M, w) \langle L_\mathcal{B} p; \beta \rangle (M', w') \Leftrightarrow \text{EXPAND}_{(p,\mathcal{B})}(M, w) \langle \beta \rangle (M', w')$

  $(M, w) \langle S_{i,j} p; \beta \rangle (M', w') \Leftrightarrow \text{SIDE-EFFECT}_{(p,i,j)}(M, w) \langle \beta \rangle (M', w')$

- $(M, w) \models p \Leftrightarrow \pi(w)(p) = \texttt{true}$

  $(M, w) \models \neg \phi \Leftrightarrow (M, w) \not\models \phi$

  $(M, w) \models \phi \wedge \psi \Leftrightarrow (M, w) \models \phi$ and $(M, w) \models \psi$

  $(M, w) \models B_i \phi \Leftrightarrow (M, v) \models \phi$ for all v such that $w R_i v$

  $(M, w) \models [\sigma] \phi \Leftrightarrow (M', w') \models \phi$ if $(M, w) \llbracket \sigma \rrbracket (M', w')$

- $Mod(M, w, p) =$

$$\{ B_i p \Rightarrow \beta \in \mathcal{C} \mid (M, w) \langle \beta \rangle (M', w'),$$
$$(M', w') \models \phi \nRightarrow (M, w) \models \phi, \ (M, w) \models B_i p \} \cup$$
$$\{ H_i p \Rightarrow \beta \in \mathcal{C} \mid (M, w) \langle \beta \rangle (M', w'), \ (M', w') \models \phi \nRightarrow (M, w) \models \phi \}$$

The operation $\rhd_p$ 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 $Mod(M, w)$ is organized in such a way that no transition rule is applied over and over again. The recursive definition of $\rhd$ therefore is well-defined, since it stops if no fresh transition rule can be applied. In the definition of Mod it is checked if the belief of the agents changes under the transition rules, preventing an infinite chain of rewrites for $\rhd_p$. 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 than 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 $(M, w) = \langle S, \pi, R_1, ..., R_m \rangle$ we define the function ADDATOM$_p$ such that $(M', w') = $ ADDATOM$_p(M, w) = \langle S', \pi', R'_1, ..., R'_m \rangle$ where

- $S' = pos(S) \cup neg(S)$
- $\pi'(pos(s))(q) = $ if $p = q$ then `true` else $\pi(s)(q)$
- $\pi'(neg(s))(q) = $ if $p = q$ 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 5.2 *Given a model (M, w) and $(M', w') = \text{ADDATOM}_p(M, w)$ it holds that*

- $(M', w') \models p$
- $(M, w) \models \phi \Leftrightarrow (M', w') \models \phi$ *for $p \notin \phi^*$ with ϕ^* the closure under subformulas of ϕ*
- *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 5.3 *Given a model (M, w), the model (M', w') such that*

$$(M, w)\langle L_i p; S_{ji} p \rangle x[Pub(i, p)](M', w')$$

(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 [9] 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:

(i) $a \rightarrow b : \{x\}_{k_a}$
(ii) $b \rightarrow a : \{\{x\}_{k_a}\}_{k_b}$
(iii) $a \rightarrow b : \{x\}_{k_b}$

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 $\{x\}_{k_a} = [\![\{x\}_{k_a}]\!]$ and m_{ab} abbreviates $\{\{x\}_{k_a}\}_{k_b} = [\![\{\{x\}_{k_a}\}_{k_b}]\!]$.

We assume that

- encryption is commutative, i.e. $\{\{x\}_{k_a}\}_{k_b} = \{\{x\}_{k_b}\}_{k_a}$ (in the actual world);
- every agent has its own key not possessed by any other agent;
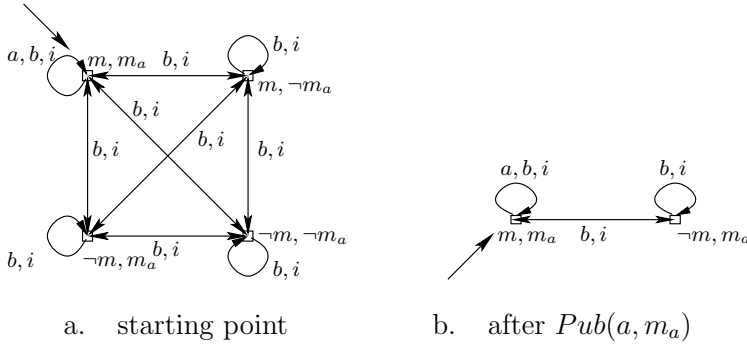- during the run of the protocol, the value of the keys do not change.

a.   starting point   b.   after $Pub(a, m_a)$

Fig. 7.

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 7.a.

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 7.b 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.

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 8.a. Observe that in the next step of the protocol $L_{\mathcal{A}} m_{ab}$ ($Pub(b, m_{ab})$) is executed, since the message is

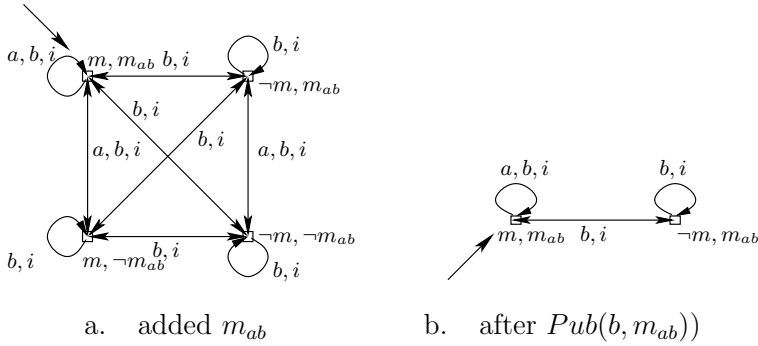a.   added m_{ab} b.   after $Pub(b, m_{ab}))$

Fig. 8.

being transmitted to all agents on the network. So with Lemma 5.3 it is justified to skip the steps that are required by the transition rules. Thus, we get the model in Figure 8.b. 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.
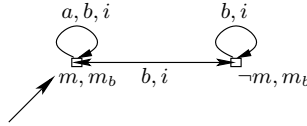
Fig. 9. after $Pub(a, m_b)$

We introduce m_b and execute $Pub(a, m_b)$ because we can again skip the actions in the transition rules (Lemma 5.3). So we end up with the model in Figure 9. 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 10.a.

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 10.b. 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 $(M', w') \models B_i((B_b m \vee B_b \neg m) \wedge B_a(B_b m \vee B_b \neg m))$.
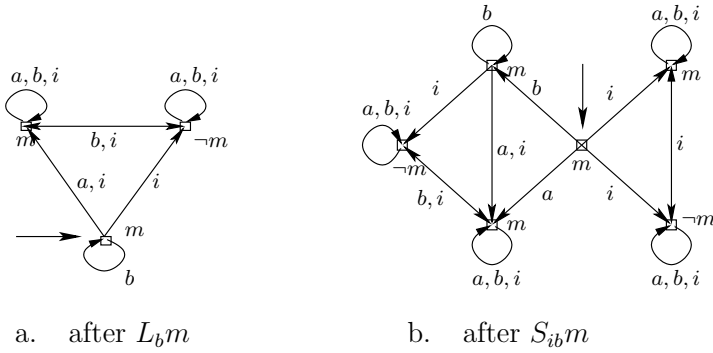
a.   after $L_b m$ b.   after $S_{ib} m$

Fig. 10.

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 [15]. It should be noted, that we focus here on 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 the verification of 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

[1] 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.

[2] R.J. Anderson. *Security Engineering: A Guide to Building Dependable Distributed Systems.* Wiley, 2001.

[3] M. Abadi and M. Tuttle. A semantics for a logic of authentication. In *Proc. PODC'91*, pages 201–216. ACM, 1991.

[4] A. Baltag. A logic for suspicous players: Epistemic actions and belief-updates in games. *Bulletin of Economic Research*, 54:1–46, 2002.

[5] M. Burrows, M. Abadi, and R.M. Needham. A logic of authentication. *ACM Transactions on Computer Systems*, 8:16–36, 1990.

[6] A. Bleeker and L. Meertens. A semantics for BAN logic. In *DIMACS Workshop on Design and Formal Verification of Protocols*. Rutgers University, `http://dimacs.rutgers.edu/Workshops/Security`, 1997.

[7] A. Baltag and L.S. Moss. Logics for epistemic programs. *Synthese: Knowledge, Rationality and Action*, 139:165–224, 2004.

[8] A. Baltag, L.S. Moss, and S. Solecki. The logic of public announcements, common knowledge and private suspicions. In Itzhak Gilboa, editor, *Proc. TARK'98*, pages 43–56, 1998.

[9] J.A. Clark and J.L Jacob. A survey of authentication protocols 1.0. Technical report, University of York, 1997.

[10] H.P. van Ditmarsch. *Knowledge games.* PhD thesis, ILLC Dissertation Series 2000-06. University of Groningen, 2000.

[11] 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.

[12] D. Dolev and A.C. Yao. On the security of public-key protocols. *IEEE Transaction on Information Theory*, 29:198–208, 1983.

[13] J. Gerbrandy. Dynamic epistemic logic. Technical Report LP–97–04, ILLC, 1997.

[14] J. Gerbrandy. *Bisimulations on Planet Kripke.* PhD thesis, ILLC Dissertation Series 1999-01. University of Amsterdam, 1999.

[15] A.J. Hommersom. Reasoning about security. Master's thesis, Universiteit Utrecht, 2003.

[16] 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.

[17] B. Kooi. *Knowledge, Chance, and Change.* PhD thesis, ILLC Dissertation Series 2003-01. University of Groningen, 2003.

[18] 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.

[19] P.F. Syverson and I. Cervesato. The logic of authentication protocols. In R. Focardi and R. Gorrieri, editors, *Foundations of Security Analysis and Design, Tutorial Lectures*, pages 63–136. LNCS 2171, 2000.

[20] B. Schneier. *Secrets and Lies: Digital Security in a Networked World.* Wiley, 2000.

[21] 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.

[22] 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.