Dynamic epistemic verification of security protocols: framework and case study

Francien Dechesne, Yanjing Wang

ABSTRACT. We propose a dynamic epistemic framework for the verification of security protocols. First, we introduce a dynamic epistemic logic equipped with iteration and cryptographic supplements in which we can formalize and check (epistemic) requirements of security protocols. On top of this, we give a general guide how to go from a protocol specification to its representation in our framework. We demonstrate this by checking requirements of a simplified version of a protocol for confidential message comparison.

Keywords: Dynamic epistemic logic, security, protocol verification

1 Introduction

Security protocols are designed to make sure that only the right agents get to *know* the right things. It seems natural to reason about such protocols using logics of knowledge and belief, since the required properties can be naturally expressed in such languages. One of the first efforts in formal verification of security protocols indeed was a logic containing a *predicate* for belief: BAN-logic [BAN89] is a logical language with an adaptable set of derivation rules, that aim to capture the reasoning steps one can safely make in the context of a given protocol. However, formalizing protocols in BAN remains ad hoc by the lack of a mechanism to transform a (high level) specification into a BAN-formalization. Additionally, the lack of a proper semantics for the original language made the value of correctness proofs in BAN-logic unclear. In fact, other verification methods found attacks on protocols that were proven correct before using BAN, most notably the man-in-the-middle attack to the Needham-Schroeder authentication protocol [Low96].

Operational aspects of protocols in general are successfully captured in various process algebraic approaches to the verification problem, generally enforced by tool support (model checking). However, as pointed out in [HS04], the formalization of some security properties (like anonymity and

privacy) is very subtle and error-prone. There is a general feeling and some evidence that epistemic logics could lead to more elegant and intuitive formalizations, at least for a certain subclass of protocols and their properties. Many epistemic logics for protocol analysis have been proposed recently. On one hand, [CD05a] gives Kripke-style semantics for an adaptation of BAN-logic, with completeness results. On the other hand, several *temporal* epistemic logics were proposed, e.g. [HO05, DGFvdH04, RS05, JH06] for analyzing security protocols, most of which follow the general idea of adding indistinguishability relations between states on the basis of local states of the agents. Another development is *dynamic* epistemic logic [BM04], that incorporates the epistemic effects of actions in the logic. This logic was used to analyze the famous protocol for anonymous broadcast, the Dining Cryptographers, in [EO05]. But this is not a typical protocol in the sense that it prescribes one specific scenario, with no intruder and no interleaving.

An initial attempt to formalize more typical security protocols in a dynamic epistemic framework was presented in [HMV04]. In this paper we focus on how the dynamic epistemic framework of [BM04] can be used to reason about all possible behaviors of security protocols, incorporating behavior of an intruder. For this purpose, we take a higher abstraction level with respect to the cryptographic reasoning involved. We summarize the highlights of our framework:

- We enrich dynamic epistemic logic with refined propositions to talk about the cryptographic elements, and an iteration operator to express and check typical requirements such as "after any run of the protocol property ϕ holds".
- Like [RS05, JH06], we distinguish between two types of knowledge: declarative knowledge as in "I *know* the secret" (where a secret is considered to be a piece of information, like a password) and propositional knowledge as in "I *know* she knows the secret". Thus we avoid part of the logical omniscience problem ([CD05b]) and stay at a fairly good abstraction level.
- We give a general way of modeling a protocol from its specification to a single action model, that also includes the intruder model. The iteration of this action model generates all the possible runs and epistemic indistinguishabilities.
- Our modeling is modular in the sense that we can separate the intruder model from the protocol model. Thus we may verify the protocol against different intruder models without altering the protocol model.
- The product update with the action model, models the higher level evolution of knowledge by the way agents observe the actions (rather than by which information they send or receive).

- We analyze a simplified version of the protocol for confidential secret comparison, as proposed in [Tee07]. The protocol is intended to let an agent find out whether another agent shares his secret (which can be regarded as a higher order knowledge statement), without revealing it if they don't share it.

The rest of the paper is structured as follows. We first define the logic, and then demonstrate through an example how it can be used to formalize security protocols. We end by mentioning some of the many interesting open questions regarding this direction of research.

2 A DEL-language with iteration for protocol analysis

We consider a dynamic epistemic language in the style of [DK06], extended with iterated action executions, and tailored to express facts in a security protocol setting, by a refinement of the structure of the basic propositions in the language: we construct basic propositions from a predicate '*has*' (to express that an agent *has* a certain message) and another predicate 'mk' (to *mark* that a certain protocol action that has been performed).

Parameters of this language are a set of agents I, a set $J \subseteq I$ of the agents whose epistemic states we want to discuss, a set of actions Act and a set of atomic message terms with some cryptographic functions. For the protocol to be discussed in the next section, we only need a singleton set of *hash functions* H, a finite set of atomic secret terms $\mathcal{S}$, and a special 'Nil'-term N. It is straightforward to add other cryptographic elements, like encryption (keys), nonces etc. when necessary for the analysis of a given protocol.

Definition 2.1. **(M and Φ^I_{sec})** Let M be the set of *message terms* m defined by

$$m ::= \quad i \quad | \quad s \quad | \quad h \quad | \quad N \quad | \quad h(m) \quad | \quad (m, m')$$

with $i \in I$, $s \in \mathcal{S}$, and $h \in H$. The set Φ^I_{sec} of basic propositions p is defined by

$$p ::= \quad has_i m \quad | \quad \overline{has}_i m \quad | \quad mk(\alpha)$$

with $i \in I$, $m \in M$ and $\alpha \in Act$. Let Φ_{IS}, $\Phi_{\overline{IS}}$ and Φ_{AM} be the subsets of Φ^I_{sec} containing only the propositions of the first, second and third type respectively. $\lhd$

The difference between $has_i m$ and $\overline{has}_i m$ is that the first is intended to mean that agent i has the message m itself, while the second means that i could *generate* m from the messages he possesses. This will become clearer when we define the semantics.

Definition 2.2. (Language $\mathcal{L}^{I,J}_{sec}$) The formulas of $\mathcal{L}^{I,J}_{sec}$ are built from the set Φ^I_{sec} as follows:

$$\top \quad | \quad p \quad | \quad \phi \wedge \psi \quad | \quad \neg\phi \quad | \quad K_j\phi \quad | \quad C_G\phi \quad | \quad [\mathcal{A},\alpha]\phi \quad | \quad [\mathcal{A}]^*\phi$$

with $p \in \Phi^I_{sec}$, $j \in J$, $G \subseteq J$ and α an action in the action model $\mathcal{A}$. $\lhd$

The intended meaning of the formulas is as usual in dynamic epistemic logic: $K_j\phi$ should be read as "agent j knows ϕ", $C_G\phi$ as "the agents in G commonly know ϕ", $[\mathcal{A},\alpha]\phi$ as "if the action α can be executed, then after this action ϕ holds". We include iteration: $[\mathcal{A}]^*\phi$ is to be read as "after every possible finite sequence of actions in $\mathcal{A}$, ϕ holds".

As usual, we define $\bot$, $\phi \vee \psi$, $\phi \to \psi$, $\langle K_i\rangle\phi$ and $\langle C_G\rangle\phi$ as the abbreviations of $\neg\top$, $\neg(\neg\phi \wedge \neg\psi)$, $\neg\phi \vee \psi$, $\neg K_i\neg\phi$ and $\neg C_G\neg\phi$ respectively. We use $[\mathcal{A}]\phi$ as the abbreviation of $\bigwedge_{\alpha\in\mathcal{A}}[\mathcal{A},\alpha]\phi$. We denote $\underbrace{[\mathcal{A}]\cdots[\mathcal{A}]}_{n}\phi$ as $[\mathcal{A}]^n\phi$ where $n \geq 1$. We define $\langle\mathcal{A},\alpha\rangle\phi$, $\langle\mathcal{A}\rangle\phi$ and $\langle\mathcal{A}\rangle^*\phi$ as the abbreviations of $\neg[\mathcal{A},\alpha]\neg\phi$, $\neg[\mathcal{A}]\neg\phi$ and $\neg[\mathcal{A}]^*\neg\phi$ respectively.

In order to interpret our $\mathcal{L}^{I,J}_{sec}$-formulas, we let the finer structure of the basic propositions correspond with a finer structure in the Kripke models (replacing the traditional valuation) by adding 'Information Sets' (IS) and 'Action Markers' (AM).

Definition 2.3. ($\mathcal{L}^{I,J}_{sec}$-models) A model for $\mathcal{L}^{I,J}_{sec}$ is a tuple:

$$\mathfrak{M} = (W, \{R_j\}_{i\in J}, IS, AM)$$

where W is a non-empty set of possible worlds; R_j are binary equivalence relations on W for all $j \in J$, and finally: $IS : W \times I \to \mathcal{P}(M)$ and $AM : W \to \mathcal{P}(Act)$. $\lhd$

For $w \in W, i \in I$, the set $IS(w,i)$ is intended to be the set of messages that agent i possesses in world w, either by some initial distribution of information, or by receiving them. For $\alpha \in Act$, the set $mk(w)$ should be read as the set of actions which have been marked.

The functions IS and AM determine the *valuation* of our basic propositions. Proposition $has_i m$ is satisfied in world w, if and only if $m \in IS(w,i)$. But based on the messages an agent possesses, he may also *derive* new messages. This goes along the following message derivation rules (cf. the `synth` and `analz`-rules of [Pau97]):

Definition 2.4. (Cryptographic Reasoning) Given a set $M' \subseteq M$, $\overline{M'}$ is the closure of M' under the following cryptographic derivation rules:

$$\frac{m \quad m'}{(m,m')} \qquad \frac{(m,m')}{m} \qquad \frac{(m,m')}{m'} \qquad \frac{m \quad h}{h(m)}$$

$\lhd$

For example, you can compute $h(s)$ if you have the secret term s and the hash function h, but you can not derive the message s from the messages $h(s)$ and h only, since s is not in the closure of $\{h, h(s)\}$.

Summarizing, the semantics of our basic propositions is as follows:[1]

Definition 2.5. (Satisfaction of basic propositions) Let $\mathfrak{M}$ be a model for $\mathcal{L}^{I,J}_{sec}$. Then:

$$\boxed{\begin{array}{lll} \mathfrak{M}, w \vDash has_i(m) & \Longleftrightarrow & m \in IS(w,i) \\ \mathfrak{M}, w \vDash \overline{has}_i(m) & \Longleftrightarrow & m \in \overline{IS(w,i)} \\ \mathfrak{M}, w \vDash mk(\alpha) & \Longleftrightarrow & \alpha \in AM(w) \end{array}}$$

$\lhd$

We will now define the action models. The actions in an action model will have pre- and postconditions, corresponding with the facts that should hold for an action to be possible, and the effects of executing that action in terms of changing (finitely many) basic facts.

For the postconditions, we adapt the terminology in [DK06]. If Φ is the set of basic propositions for a DEL-language $\mathcal{L}$, then we call a function $\Phi \to \mathcal{L}$ *a substitution in* $\mathcal{L}$ if it maps all but a finite number of basic propositions to themselves. We denote the set of all substitutions for Φ in $\mathcal{L}$ by $SUB(\mathcal{L})$.

In our setting, it suffices to consider restricted classes of pre- and postconditions for actions. The preconditions will all be in $\mathcal{L}|\Phi^I_{sec}$: the propositional (i.e. modality free) part of the language $\mathcal{L}^{I,J}_{sec}$. We divide the postconditions in two 'types': changes in the information sets (Pos_{IS}), and changes in the action markers (Pos_{AM}).

Definition 2.6. (Action model) An action model $\mathcal{A}$ for $\mathcal{L}^{I,J}_{sec}$ is a tuple: $\mathcal{A} = (Act, \{\sim_j\}_{j\in J}, Pre, Pos_{IS}, Pos_{AM})$ where Act is a finite non-empty set of actions, and $\sim_j$ is an equivalence relation on Act for each $j \in J$. $Pre : Act \to \mathcal{L}|\Phi^I_{sec}$ assigns to each action a propositional precondition. $Pos_{IS} : Act \to SUB(\mathcal{L}^{I,J}_{sec})$ assigns to each $\alpha \in Act$ a substitution for $\mathcal{L}$ with the property that $Pos_{IS}(\alpha)(p) \in \Phi_{IS} \cup \{\top, \bot\}$ for all $p \in \Phi_{IS}$ and $Pos_{IS}(\alpha)(p) = p$ otherwise. $Pos_{AM} : Act \to SUB(\mathcal{L}^{I,J}_{sec})$ assigns to each $\alpha \in Act$ a substitution for $\mathcal{L}$ with the property that $Pos_{AM}(\alpha)(p) \in \Phi_{AM} \cup \{\top, \bot\}$ for all $p \in \Phi_{AM}$ and $Pos_{AM}(\alpha)(p) = p$ otherwise. $\lhd$

In the following, we write '$m \in i$' for the substitution mapping $has_i m$ to $\top$ while leaving the other basic propositions in place. The effect will be

[1] Note that by taking messages as terms rather than propositions, we can now safely express "i knows that j 'knows' the password, while i does not 'know' the password herself": we can do so by the $\mathcal{L}^{I,J}_{sec}$-formula $K_i has_j p \wedge \neg has_i p$. A formula like $K_i K_j p \wedge \neg K_i p$ would be inconsistent in the standard reading of the K-operator.

the addition of m to the information set of i after the action. We write 'α^+' for the substitution mapping $mk(\alpha)$ to $\top$ while leaving the other basic propositions in place, and similarly 'α^-' for the substitution mapping $mk(\alpha)$ to $\bot$. This will mark (unmark) α after the action. All postconditions in our protocol formalization in the next section will be combinations of these 'atomic' substitutions.

Updating with an action model transforms an $\mathcal{L}^{I,J}_{sec}$-model into another $\mathcal{L}^{I,J}_{sec}$-model as follows:

Definition 2.7. (Update execution) Given a model $\mathfrak{M}$ for $\mathcal{L}^{I,J}_{sec}$ and an action model $\mathcal{A}$, the updated model $M \circ \mathcal{A} = (W', \{R'_j\}_{j\in J}, IS', AM')$ is defined as follows:

$$\begin{array}{lcl} W' & = & \{\langle w,\alpha\rangle | \mathfrak{M}, w \vDash Pre(\alpha)\} \\ R'_j & = & \{(\langle w,\alpha\rangle, \langle v,\alpha'\rangle) | wR_jv \text{ and } \alpha \sim_j \alpha'\} \\ IS'(\langle w,\alpha\rangle, i) & = & \{m | M, w \vDash Pos_{IS}(\alpha)(has_i m)\} \\ AM'(\langle w,\alpha\rangle) & = & \{\beta | M, w \vDash Pos_{AM}(\alpha)(mk(\beta))\} \end{array}$$

◁

Definition 2.8. (Satisfaction) On top of the satisfaction relation for $p \in \Phi^I_{sec}$ of Definition 2.5, we define:

$\mathfrak{M}, w \vDash \top$	$\Longleftrightarrow$	true
$\mathfrak{M}, w \vDash \neg\phi$	$\Longleftrightarrow$	$\mathfrak{M}, w \nvDash \phi$
$\mathfrak{M}, w \vDash \phi \wedge \psi$	$\Longleftrightarrow$	$\mathfrak{M}, w \vDash \phi$ and $\mathfrak{M}, w \vDash \psi$
$\mathfrak{M}, w \vDash K_j\phi$	$\Longleftrightarrow$	for all v, if wR_jv then $\mathfrak{M}, v \vDash \phi$
$\mathfrak{M}, w \vDash C_G\phi$	$\Longleftrightarrow$	for all v, if wR_Gv then $\mathfrak{M}, v \vDash \phi$
$\mathfrak{M}, w \vDash [\mathcal{A},\alpha]\phi$	$\Longleftrightarrow$	if $\mathfrak{M}, w \vDash Pre(\alpha)$ then $\mathfrak{M} \circ \mathcal{A}, \langle w,\alpha\rangle \vDash \phi$
$\mathfrak{M}, w \vDash [\mathcal{A}]^*\phi$	$\Longleftrightarrow$	for all $n \in \mathcal{N}$: $\mathfrak{M}, w \vDash [\mathcal{A}]^n\phi$

where R_G is the transitive closure of the relations in $\{R_j | j \in G\}$. ◁

REMARK 1. It is easy to check the following facts:

$$\begin{array}{lcl} \mathfrak{M}, w \vDash [\mathcal{A}]\phi & \Longleftrightarrow & \text{for all } \alpha \in \mathcal{A}\text{: } \mathfrak{M}, w \vDash [\mathcal{A},\alpha]\phi \\ \mathfrak{M}, w \vDash \langle\mathcal{A}\rangle\phi & \Longleftrightarrow & \text{there is an } \alpha \in \mathcal{A}\text{: } \mathfrak{M}, w \vDash \langle\mathcal{A},\alpha\rangle\phi \\ \mathfrak{M}, w \vDash \langle\mathcal{A}\rangle^*\phi & \Longleftrightarrow & \text{there are } \alpha_0 \cdots \alpha_n\text{: } \mathfrak{M}, w \vDash \langle\mathcal{A},\alpha_0\rangle \cdots \langle\mathcal{A},\alpha_n\rangle\phi \end{array}$$

Our language $\mathcal{L}^{I,J}_{sec}$ is comparable to the DEL-language with more restricted iterated relativizations considered in [MM05]. The satisfiability problems of many fragments of that language are proven to be Σ^1_1-complete on general models, thus the logics are not recursively axiomatizable. However, the decidability of our language on $S5$ models is still open (while the models we construct in the next section are all S5).

3 Modeling protocols

3.1 The verification problem in general

Protocol A (communication) protocol *Prot* is specified by a sequence of action patterns (possibly with conditions), usually of the form $\mathtt{A} \rightarrow \mathtt{B} : m$ ('A sends message m to B'), where m is a message of some fixed pattern. Parameters of these patterns each have a fixed *domain*. We use the term *role* for any parameter in the specification that ranges over agents.

Instantiations In a concrete run of the protocol, the action patterns of the specification are instantiated by some *instantiation* θ, which is a map from the parameters of the protocol to their respective domain. Several instantiations can be interleaved in one run of the protocol. In the context of some given protocol, we use *Inst* for the set of all possible instantiations. In this paper, the set of parameters and their domains are all finite, such that *Inst* is finite as well.

Network Like [CM05], we assume that the protocol is to be executed on a network with an input-buffer (modeled as special agent *In*) and an output-buffer (*Out*), in between which a malicious agent, the 'intruder', could tamper with the messages. Trusted agents only send messages to *In* and receive messages from *Out*.

The verification problem Given a protocol *Prot* (with fixed domains for the parameters) and a requirement $\phi \in \mathcal{L}^{I,J}_{sec}$, verification of the requirement on the protocol is formalized as follows:

$$\mathfrak{M} \vDash \bigwedge_{\theta \in Inst} [Prot_{Intr}]^*[Prot_{Intr}, \alpha(\theta)]\phi(\theta)$$

where:

- $\mathfrak{M}$ is the initial model, incorporating the assumptions on the initial distribution of information, and the initial epistemic uncertainties of the agents.
- $Prot_{Intr}$ is the action model including all possible actions according of *Prot*, and all possible actions according to some intruder model *Intr*.
- α is an action pattern of *Prot* (typically the final one).
- ϕ is an $\mathcal{L}^{I,J}_{sec}$-formula scheme (with parameters) stating some static epistemic or factual properties. $\bigwedge_{\theta \in Inst}[Prot_{Intr}]^*[Prot_{Intr}, \alpha(\theta)]\phi(\theta)$ is then the $\mathcal{L}^{I,J}_{sec}$-formula obtained by taking the conjunction over all (finitely many) possible instantiations of the parameters. It expresses: "In any possible run of the protocol, property ϕ holds right after the execution of action α."

3.2 Modeling the static and action models

We now briefly indicate how to model two essential components of our framework for the verification problem of a given protocol: viz. the initial model $\mathfrak{M}$ and the action model $Prot_{Intr}$. The set Ag of agents should be given, and we will take $B \in \text{Ag}$ to be the fixed name for the agent performing the intruder actions (the 'bad guy'). In general we can take $I = \text{Ag} \cup \{In, Out\}$ and $J = \text{Ag}$ (although we might want to restrict J further when checking requirements for specified agents).

The action model $Prot_{Intr}$. The set of actions Act of $Prot_{Intr}$ consists two independent parts: actions according to the protocol specification, and intruder actions according to the intruder model. (This gives our framework the nice property of being *modular*, allowing an easy comparison of one protocol under different intruder strengths, or different protocols under the same intruder model.)

In accordance with to our network model, we split the **protocol actions** of the form $i \to j : m$ into a 'send' (α) $i \to In : m$ and a 'receive' (β) $Out \to j : m$. The precondition for α is that m can actually be constructed by i from his information set: $Pre(\alpha) = \overline{has}_i m$. The IS-postcondition is that m is delivered at In: $Pos_{IS}(\alpha) = m \in In$. The AM-postconditions are optional but are intended to order the protocol actions executed by one role. They mark the current action or unmark a previous action (e.g. α^+ and γ^-), and such action markings can occur as preconditions for a later action. Similarly, the conditions for the receive are $Pre(\beta) = has_{Out} m$ – note the use of has rather than $\overline{has}$: we assume the buffers do not construct new messages– and $Pos_{IS}\beta = m \in j$ (and a possible action marking).

We transform a protocol specification into a set of parameterized actions. The actions in Act are all concrete actions generated by instantiating the parameters in those action schemes.

In our network model, the **intruder actions** are either of the form $In \to B : m$ or $B \to Out : m$. The strength of the intruder model is reflected in the pre- and postconditions of these actions. The standard Dolev-Yao model, where the intruder can take everything from the In-buffer, and construct any message into the Out-buffer, is modeled as follows:[2]

Action	*Direction*	*Message*	*Pre*	Pos_{IS}	Pos_{AM}
`[take m]`	$In \to B$	m	$has_{In} m$	$m \in B$	–
`[fake m]`	$B \to Out$	m	$\overline{has}_B m$	$m \in Out$	–

[2]Other intruder models could have weaker actions like `[eavesdrop]`: intruder learns the message and passes it on; `[jam m]`: intruder deletes the message from In without learning.

Epistemic relations between actions A simple general way of modeling indistinguishabilities between actions, is to assume that a trusted agent i can only distinguish the actions performed by himself; all actions by others are linked by $\sim_i$.

For the intruder, we might want to model stronger observational power. This will add new layers of power to the intruder model. In our case study however, we will construct $\sim_B$ as for the trusted agents.

Note that it is safe to model weaker observational powers for the agents for whom the requirements contain positive knowledge claims (of the form $K_i\varphi$), and stronger observational powers for agents for whom the requirements contain negative knowledge claims (of the form $\neg K_i\varphi$).

Initial Model The initial model $\mathfrak{M} = (W, \{R_j\}_{j\in J}, IS, AM)$ can be obtained by following the guidelines below.

- W and IS: we can let each function $f : J \to \mathcal{P}(M)$ define one world w_f in the initial model; then every w is determined by the message distribution over the non-buffer agents.[3] We start with empty buffers. This determines IS: for $w = w_f \in W$: $IS(w, j) = f(j)$ for $j \in J$, and $IS(w, In) = IS(w, Out) = \emptyset$.
- R_j: Let wR_jv if and only if $IS(w, j) = IS(v, j)$ (making the R_j into equivalence relations).
- AM: in any world $w \in W$ we take $AM(w) = \emptyset$.

In the next section, we demonstrate the above general modeling method by analyzing an example protocol.

3.3 Analysis of an example protocol

Let us consider the situation proposed in [Tee07, p. 101]: Suppose Charley confidentially received a secret, that he would like to gossip about with Alice, whom he suspects knows about the secret as well. We propose and analyze a protocol that is intended to allow for the challenger (Charley in this case) to check whether the responder (Alice) actually possesses the secret, without revealing the secret to her if she happened not to have it, nor to anyone else (in particular the intruder).

The protocol specification We specify a protocol for confidential gossiping as follows, where `s` stands for the secret that 'challenger' `C` possesses and wants to gossip about with 'responder' `R`:

1. $\mathtt{C} \to \mathtt{R} : h(\mathtt{s})$, provided that $has_{\mathtt{C}}\mathtt{s}$

[3] By not having several worlds with the same message distribution, we will not be able to model higher order statements like 'A does not know that B knows that A has m'. This restriction may have to be avoided for more 'epistemic' protocols.

2. (a) if $has_{\mathtt{R}}\mathtt{s}$, then: $\mathtt{R} \to \mathtt{C} : h(h(\mathtt{s}), \mathtt{s})$
   (b) if $\neg has_{\mathtt{R}}\mathtt{s}$, then: $\mathtt{R} \to \mathtt{C} : (N, h(\mathtt{s}))$

Here $\mathtt{s}$, $\mathtt{C}$, $\mathtt{R}$ are the parameters, with as domain for the first a finite set of secrets $\mathcal{S}$, and for the latter two a set of agents Ag. We take $\mathrm{Ag} = \{A, B, C\}$ to consist of the trusted agents A and C, and the intruder B. The hash function h is not a parameter in this case: we fix it in the specification to be the same in every instantiation of the protocol.

The intuitive reading of the protocol specification is that $\mathtt{C}$ initiates the communication by sending $\mathtt{R}$ the hash value of the secret $h(\mathtt{s})$, and then $\mathtt{R}$ either responds the hash of the pair $(\mathtt{s}, h(\mathtt{s}))$ if she has the secret, or $h(\mathtt{s})$ paired with the Nil-term N otherwise. It is worth noting that one agent can play different roles in different instantiations. For example, while agent C challenges agent A for secret s, A might want to challenge C for some other secret s' by the same protocol.

The action model $Prot_{Intr}$ The actions in the protocol part of the action model are instantiations of the following action patterns: [4]

Action	*Direction*	*Message*	*Pre*	Pos_{AM}
$\alpha_1(\mathtt{C}, \mathtt{R}, \mathtt{s})$	$\mathtt{C} \to In$	$(h(\mathtt{s}), (\mathtt{C}, \mathtt{R}))$	$\overline{has_{\mathtt{C}}}\mathtt{s} \wedge \overline{has_{\mathtt{C}}}m$	${\alpha_1}^+$
$\beta_1(X, \mathtt{R}, Y)$	$Out \to \mathtt{R}$	$(h(Y), (X, \mathtt{R}))$	$has_{Out}m$	${\beta_1}^+$
$\alpha_{2a}(X, \mathtt{R}, Y)$	$\mathtt{R} \to In$	$(h(h(Y), Y), (\mathtt{R}, X))$	$mk(\beta_1) \wedge \overline{has_{\mathtt{R}}}Y \wedge \overline{has_{\mathtt{R}}}m$	${\beta_1}^-, {\alpha_{2a}}^+$
$\alpha_{2b}(X, \mathtt{R}, Y)$	$\mathtt{R} \to In$	$((N, h(Y)), (\mathtt{R}, X))$	$mk(\beta_1) \wedge \neg\overline{has_{\mathtt{R}}}Y \wedge \overline{has_{\mathtt{R}}}m$	${\beta_1}^-, {\alpha_{2b}}^+$
$\beta_{2a}(\mathtt{C}, \mathtt{R}, \mathtt{s})$	$Out \to \mathtt{C}$	$(h(h(\mathtt{s}), \mathtt{s}), (\mathtt{R}, \mathtt{C}))$	$mk(\alpha_1) \wedge has_{Out}m$	${\alpha_1}^-, {\beta_{2a}}^+$
$\beta_{2b}(\mathtt{C}, \mathtt{R}, \mathtt{s})$	$Out \to \mathtt{C}$	$((N, h(\mathtt{s})), (\mathtt{R}, \mathtt{C}))$	$mk(\alpha_1) \wedge has_{Out}m$	${\alpha_1}^-, {\beta_{2b}}^+$

If we split these actions according to the executer of the actions, then the challenger role $\mathtt{C}$ performs $\alpha_1(\mathtt{C}, \mathtt{R}, \mathtt{s})$, $\beta_{2a}(\mathtt{C}, \mathtt{R}, \mathtt{s})$, and $\beta_{2b}(\mathtt{C}, \mathtt{R}, \mathtt{s})$, while the responder role $\mathtt{R}$ performs $\beta_1(X, \mathtt{R}, Y)$, $\alpha_{2a}(X, \mathtt{R}, Y)$ and $\alpha_{2b}(X, \mathtt{R}, Y)$. One can then notice that variables $(X, Y, \ldots)$ are used for the parameters that are not within the control of the role performing the action: the challenger chooses whom he will challenge (he controls the value for $\mathtt{R}$) with which secret (the value for $\mathtt{s}$). The responder on the other hand is not able to choose whom he is challenged by (therefore the variable X). In this paper, we assume strict well-typedness, viz. that the variables can only be instantiated in the intended parameters' domains.[5] Thus there are only finitely

[4] To keep the table within the page margins, we uniformly use m to abbreviate the message being sent in each action and omit Pos_{IS} which uniformly adds m into the receiver's information set. We omit the parameters to the action names α_i and β_i occurring in the pre- and postconditions: they are identical to those of the action in the *Action*-column. Note that the messages in the actions include a pair indicating the (purported) sender and the (intended) receiver. One could see these as the addresses written on an envelope being part of a letter.

[5] For example, $\beta_1(A, C, h(s))$ is not an action in $Prot_{Intr}$, since $h(s)$ is not of the required type (it is not in $\mathcal{S}$).

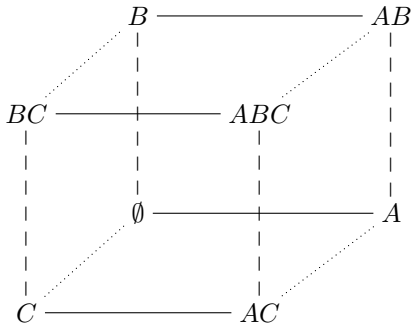
many actions for the trusted agents because the domains of the parameters and variables are finite.

We take the actions in the intruder part of the action model to be those of the Dolev-Yao intruder model as given in section 3.2. The epistemic relations between actions are generated as what we stated in section 3.2: an agent can only distinguish the actions performed by himself.

The initial model We picture the initial model for a simplified situation. Suppose we have $\mathcal{S} = \{s\}$, and let's assume that all non-buffer agents have the hash-function h, the Nil-term N, and the agents names in Ag. Then each triple

$$\langle IS(w, A), IS(w, B), IS(w, C)\rangle \in \mathcal{P}(M)^3$$

with $\{h, N\}\cup\mathrm{Ag} \subseteq IS(w, i) \subseteq \{h, N\}\cup\mathrm{Ag}\cup\{s\}$ defines a world in the initial model. If we name each world w after the agents i for whom $s \in IS(w, i)$, the initial model looks like this:



The full lines indicate the worlds that are *distinguishable* for A, the dashed lines those for B, and the dotted lines those for C. So, for example, the *in*distinguishability relation of A ($\sim_A$) is the combination of the dotted and the dashed lines.

Requirements The following are formalizations of the requirements we want the protocol to satisfy:

R1 After receiving the positive answer from $\mathtt{R}$ (by execution of $\beta_{2a}(\mathtt{C}, \mathtt{R}, \mathtt{s})$), $\mathtt{C}$ knows that $\mathtt{R}$ has $\mathtt{s}$:

$$\bigwedge_{\theta\in Inst} [Prot_{Intr}]^*[Prot_{Intr}, \beta_{2a}(\theta(\mathtt{C}), \theta(\mathtt{R}), \theta(\mathtt{s}))]K_{\theta(\mathtt{C})}has_{\theta(\mathtt{R})}\theta(\mathtt{s})$$

R2 No one, in particular not the intruder, learns $\mathtt{s}$ during the protocol execution if he didn't have the secret at the beginning:

$$\bigwedge_{\theta\in Inst} \neg has_B\theta(\mathtt{s}) \rightarrow [Prot_{Intr}]^*\neg\overline{has}_B\theta(\mathtt{s})$$

3.4 Verification of the protocol

For the verification of **R2**, we state the following useful results:

Proposition 3.1. For any set of message terms M, and any secret s: if $s \notin \overline{M}$ then $s \notin \overline{\{h(m)\} \cup M}$ for any message term m.

Proof. In the decomposing rules of Definition 2.4, $h(m)$ never occurs as premise, so the atomic message term s can not be derived by adding $h(m)$ to M. ■

In the postconditions of the actions in $Prot_{Intr}$, messages are never deleted from the information sets of the agents, so we have:

Proposition 3.2. For any agent i, and any message term m:

$$\mathfrak{M} \vDash [Prot_{Intr}]^*(\overline{has}_i m \to [Prot_{Intr}]^* \overline{has}_i m)$$

Proposition 3.3. R2 is globally true in the initial model $\mathfrak{M}$.

Proof. For a proof by contradiction, assume there is a world w in $\mathfrak{M}$ with:

$$\mathfrak{M}, w \nvDash \neg has_B s \to [Prot_{Intr}]^* \neg \overline{has}_B s$$

It follows that there exists an action sequence $\alpha_1 \cdots \alpha_n$ such that

$$\mathfrak{M}, w \vDash \neg has_B s \wedge \langle Prot_{Intr}, \alpha_1 \rangle \cdots \langle Prot_{Intr}, \alpha_n \rangle \overline{has}_B s$$

From the design of the initial model $\mathfrak{M}$, it is easy to see that:

$$\mathfrak{M}, w \vDash \neg \overline{has}_B s \wedge \langle Prot_{Intr}, \alpha_1 \rangle \cdots \langle Prot_{Intr}, \alpha_n \rangle \overline{has}_B s$$

Write the information set of agent i after executing $\alpha_1 \cdots \alpha_k$ on $\mathfrak{M}, w$ as IS^i_k. From Proposition 3.2, it follows that there must be a unique $k < n$ such that $s \notin \overline{IS^B_k}$ but $s \in \overline{IS^B_{k+1}}$. Of the actions in $Prot_{Intr}$, only the following options for α_{k+1} could add anything to B's information set:

- α_{k+1} is `[take m]`
- α_{k+1} is a β-action of the protocol where B receives m

In both cases $IS^B_{k+1} = \{m\} \cup IS^B_k$. In the first case, from the precondition of `[take m]` we know that $m \in IS^{In}_k$. Since the messages in the In-buffer only come from the protocol actions, m will have one of the following patterns: $(h(s), (\mathtt{C}, \mathtt{R}))$, $(h(h(s), s), (\mathtt{R}, X))$ or $((N, h(s)), (\mathtt{R}, X))$, with the parameters and variables instantiated with agent names. Using that these names and the Nil-term N were already in B's initial information set,

$\overline{IS^B_{k+1}}$ can be seen to be of the form $\overline{\{h(m')\} \cup IS^B_k}$ in all three cases (because $\overline{\{(h(s),(\mathtt{C},\mathtt{R}))\} \cup IS^B_k} = \overline{\{h(s)\} \cup IS^B_k}$, $\overline{\{(h(h(s),s),(\mathtt{R},X))\} \cup IS^B_k} = \overline{\{h(h(s),s)\} \cup IS^B_k}$ and $\overline{\{((N,h(s)),(\mathtt{R},X))\} \cup IS^B_k} = \overline{\{h(s)\} \cup IS^B_k}$). But then it follows from Proposition 3.1 that $s \notin \overline{\{m\} \cup IS^B_k}$. Contradiction.

In the second case, since messages in the *Out*-buffer come from a previous `fake`-action by B, and information sets are increasing (Proposition 3.2), it must be the case that $m \in \overline{IS^B_k}$. But then $\overline{IS^B_k} = \overline{IS^B_{k+1}}$. Contradiction. ■

However, requirement **R1** is violated.

Proposition 3.4. **R1** does not hold globally in the initial model $\mathfrak{M}$.

Proof. In the picture, take the world AC, where $\mathfrak{M}, AC \vDash \neg has_B s \wedge has_C s \wedge has_A s$. It is easy to check that the action sequence below can be executed on AC according to the preconditions of each action:

$\alpha_1(C,B,s)$.	$C \to In$	:	$(h(s),(C,B))$
`[take` $(h(s),(C,B))$`]`.	$In \to B$	:	$(h(s),(C,B))$
`[fake` $(h(s),(B,A))$`]`.	$B \to Out$	:	$(h(s),(B,A))$
$\beta_1(B,A,s)$.	$Out \to A$	:	$(h(s),(B,A))$
$\alpha_{2a}(B,A,s)$.	$A \to In$	:	$(h(h(s),s),(A,B))$
`[take` $(h(h(s),s),(A,B))$`]`.	$In \to B$	:	$(h(h(s),s),(A,B))$
`[fake` $(h(h(s),s),(B,C))$`]`.	$B \to Out$	:	$(h(h(s),s),(B,C))$
$\beta_{2a}(C,B,s)$.	$Out \to C$	:	$(h(h(s),s),(B,C))$

Here B as "the man in the middle" passes on C's request for checking s with B, to A as if it was from B. A responds positively to B. Now B passes her answer back to C as if it was from himself. Note that C and A are performing their actions in different instantiations of the protocol, as can be seen from the parameters to the actions. It is easy to check that the following holds (we omit the parameters):

$$\mathfrak{M}, AC \models \langle \alpha_1 \rangle \langle \mathtt{take} \rangle \langle \mathtt{fake} \rangle \langle \beta_1 \rangle \langle \alpha_{2a} \rangle \langle \mathtt{take} \rangle \langle \mathtt{fake} \rangle \langle \beta_{2a} \rangle \neg \overline{has}_B s$$

Since the epistemic relations in our model are reflexive then it follows that

$$\mathfrak{M}, AC \models \langle Prot_{Intr} \rangle^* \langle Prot_{Intr}, \beta_{2a}(C,B,s) \rangle \neg K_C \overline{has}_B s$$

which violates **R1**. ■

To prevent such man-in-the-middle attack, we should alter the protocol, for example by adding the name of the intended responder into the message being sent at the first step of the protocol, e.g. $\mathtt{C} \to \mathtt{R} : h(\mathtt{s},\mathtt{R})$. Then the message cannot be passed through to another responder.

The above is only our toy example to demonstrate how the framework works. The verification of the original set of T1-protocols in [Tee07] is expected to be done within our frame work in the future.

4 Conclusion and future work

We have presented a framework for the verification of security protocols based on dynamic epistemic logic with iteration. On the one hand, our language $\mathcal{L}_{sec}^{I,J}$ allows for a straightforward specification of security requirements, in particular if they are about what agents should (or should not) know. On the other hand, our proposed model allows to go from a protocol specification to a complete representation of the protocol, capturing both the behavior (in terms of traces of actions) and the epistemic developments of the agents (by the indistinguishability relations within the static- and action models). We demonstrated our framework by the example of a protocol with epistemic requirements, intended for confidential comparison of message terms.

There are several issues left open in this paper on which we will continue our research. Along the theoretical line of consideration, the expressivity of $\mathcal{L}_{sec}^{I,J}$ and the decidability of its verification problem are still open. With respect to the modeling choices, the observational power of the intruder can be reasonably reinforced by letting him distinguish two initially indistinguishable actions in retrospect, by acquiring information during the protocol. A promising extension is to introduce conditional epistemic relations in the action model which depend on the epistemic states of the agents. Among the approaches to the verification problem, epistemic logics are easy to be used in specifying protocol requirements while process algebras are good at generating run models. Some attempts have been tried out in [DMO07] and [HS04] to combine the power of the two. In future work, we may study using process algebra terms as the action model generators and introducing them as the dynamic modalities in the language instead of the modalities for action models. It is also an open issue to find protocols whose analysis *truly* benefits from the richness of such frameworks. In particular we are looking for protocols that involve *higher order* knowledge.

It is worth mentioning that the ultimate goal of this research is to build up a dynamic epistemic framework of security verification with tool support. A good candidate tool is DEMO developed in [Eij05]. As a dynamic epistemic modeling tool, it has been used in [EO05] and [DHMR06] to check some epistemic properties. In order to adapt our framework in DEMO, we are cooperating with its author to equip it with the fact change features which are currently still missing.

Acknowledgments We thank Sjouke Mauw, Cas Cremers and Jaco van de Pol for valuable discussions on the modeling of (security) protocols, and Wouter Teepe for providing us with the setting for our protocol. We thank Alexandru Baltag, Hans van Ditmarsch, Jan van Eijck, Barteld Kooi for insights in DEL, and we thank the anonymous referees for their insightful com-

ments on the first version of this paper. This research is part of the project *Verification and Epistemics of Multi-Party Protocol Security* funded by the Netherlands Organization for Scientific Research (NWO, project number 612.000.528).

BIBLIOGRAPHY

[BAN89] Michael Burrows, Martín Abadi, and Roger Needham. A logic of authentication. *Proceedings of the Royal Society of London*, 426(1871):233–271, December 1989.

[BM04] Alexandru Baltag and Lawrence S. Moss. Logics for epistemic programs. *Synthese*, 139:165–224, March 2004.

[CD05a] M. Cohen and M. Dam. A completeness result for BAN logic. In *Proc. Methods for Modalities*, 2005.

[CD05b] M. Cohen and M. Dam. Logical omniscience in the semantics of BAN logic. In *Proc. FCS'05*, 2005.

[CM05] C.J.F. Cremers and S. Mauw. Operational semantics of security protocols. In S. Leue and T. Systä, editors, *Scenarios: Models, Transformations and Tools*, volume 3466 of *LNCS*. Springer, 2005.

[DGFvdH04] Clare Dixon, M. Carmen Fernández Gago, Michael Fisher, and Wiebe van der Hoek. Using temporal logics of knowledge in the formal verification of security protocols. In *TIME 2004*, pages 148–151, 2004.

[DHMR06] H. van Ditmarsch, W. van der Hoek, R. van der Meyden, and J. Ruan. Model checking Russian cards. In C. Pecheur and B. Williams, editors, *Proc. MoChArt 05*, volume 149(2) of *ENTCS*, pages 105–123, 2006.

[DK06] Ditmarsch, H.P. van and B. P. Kooi. The logic of ontic and epistemic change. Technical Report OUCS-2006-11, Department of Computer Science, University of Otago, New Zealand, 2006.

[DMO07] Francien Dechesne, MohammadReza Mousavi, and Simona Orzan. Operational and epistemic approaches to protocol analysis: Bridging the gap. draft, 2007.

[Eij05] Jan van Eijck. DEMO program and documentation, 2005. Available from `http://www.cwi.nl/~jve/demo/`.

[EO05] J. van Eijck and S. Orzan. Modelling the epistemics of communication with functional programming. In *Proceedings TFP'05*, September 2005.

[HMV04] A. Hommersom, J.-J. Meyer, and E.P. de Vink. Update semantics of security protocols. *Synthese/Knowledge, Rationality and Action*, 142:229–327, 2004.

[HO05] J.Y. Halpern and K.R. O'Neill. Anonymity and information hiding in multiagent systems. *Journal of Computer Security*, pages 483–514, 2005.

[HS04] D. Hughes and V. Shmatikov. Information hiding, anonymity and privacy: A modular approach. *Journal of Computer Security*, 12(1):3–36, 2004.

[JH06] B. Jacobs and I. Hasuo. Semantics and logic for security protocols, 2006. Preprint available at `http://www.cs.ru.nl/~ichiro/papers/`.

[Low96] Gavin Lowe. Breaking and fixing the needham-schroeder public-key protocol using FDR. In *TACAs '96*, pages 147–166, London, UK, 1996. Springer-Verlag.

[MM05] Joseph Miller and Lawrence Moss. The undecidability of iterated modal relativization. *Studia Logica*, 79:373–407(35), April 2005.

[Pau97] Lawrence C. Paulson. Proving properties of security protocols by induction. In *10th Computer Security Foundations Workshop*, pages 70–83. IEEE Computer Society Press, 1997.

[RS05] R. Ramanujam and S.P. Suresh. Deciding knowledge properties of security protocols. In *Proc. TARK*, pages 219–235. Morgan Kaufmann, 2005.

[Tee07] Wouter Teepe. *Reconciling Information Exchange and Confidentiality — A Formal Approach*. PhD thesis, Rijksuniversiteit Groningen, 2007.