Available online at www.sciencedirect.com

ScienceDirect

Electronic Notes in Theoretical Computer Science 168 (2007) 159–174

Electronic Notes in Theoretical Computer Science

www.elsevier.com/locate/entcs

Epistemic Verification of Anonymity

Jan van Eijck

CWI, Kruislaan 413, P.O. Box 94079, 1090 GB Amsterdam
UiL OTS, Utrecht University, Trans 10, 3512 JK Utrecht

Simona Orzan

Department of Mathematics and Computer Science
Eindhoven University of Technology
P.O. Box 513, 5600 MB Eindhoven

Abstract

Anonymity is not a trace-based property, therefore traditional model checkers are not directly able to express it and verify it. However, by using epistemic logic (logic of knowledge) to model the protocols, anonymity becomes an easily verifiable epistemic formula. We propose using Dynamic Epistemic Logic to model security protocols and properties, in particular anonymity properties. We have built tool support for DEL verification which reuses state-of-the-art tool support for automata-based verification. We illustrate this approach by analyzing an anonymous broadcast protocol and an electronic voting protocol. By comparison with a process-based analysis of the same protocols, we also discuss the relative (dis)advantages of the process-based and epistemic-based verification methods in general.

Keywords: dynamic epistemic modeling, protocol verification, anonymity

1 Introduction

Protocols involving sensitive personal information or private choices have anonymity as a central security requirement. Typical examples are electronic voting, web browsing, posting on bulletin boards, sharing files, auctions. In order to prove claims about the correctness of such protocols, (tool-supported) formal reasoning frameworks for anonymity are desirable. Unfortunately, providing these is not straightforward, since anonymity does not quite fit in the most successful automated verification framework to date, which is automata-based model checking. Traditional security properties like secrecy and authentication are expressible as reachability properties of finite-state systems and therefore allow for trace-based model checking. As noted by Schneider and Sidiropoulos [26], a formal interpretation of anonymity comprises for a given set of traces the existence of another, observationally indistinguishable set of traces in which different sensitive personal information is attributed

doi:10.1016/j.entcs.2006.08.026

to the same person or the same choice is attributed to different participants. Thus, anonymity is not trace-based as secrecy and authentication, but is a higher-order trace property. Note also that, unlike traditional properties, anonymity depends on the point of view of the untrusted observer: participant, outsider, coalition of participants. This means that formalisms relying on trace specifications require a remodeling of the protocol for every anonymity property which is to be verified.

Both these problems (anonymity being a higher-order trace property and a perspective-dependent property) are implicitly solved by approaches based on epistemic logic [7,29,21,16]. There, the focus is on the information flow of the protocol and not on the observable behavior. An *epistemic state* of the protocol is a snapshot of all the information and uncertainties of the participating parties about the relevant facts (modeled by propositions), as well as about the information held by the other parties. Anonymity of an agent in a given epistemic state translates as the uncertainty of the observer regarding a particular proposition which models sensitive information belonging to that agent. This is expressible as an epistemic formula involving the knowledge modality (for instance, $\neg K_c ayes$ expresses that c does not know that a voted 'yes'). However, the logic-based treatments of security protocols and anonymity in particular [7,29,25,16] are often complex and protocol modeling requires a high degree of expertise. The information updates generating the transitions between epistemic states are especially tedious to specify, because logics are geared to expressing properties rather than operational protocol steps.

In this paper, we investigate the application of the recently developed Dynamic Epistemic Logic (DEL) [3,2,27] to automatic verification of anonymity. DEL distinguishes from other logics by the introduction of *action models*, which are Kripke structures describing information updates corresponding to various forms of communications (public announcements, passing messages, sharing secrets, telling lies). The actual updates are performed by computing the *update product* of an epistemic model and an action model. This action models/update product combination has the advantage of a strong operational flavor and intuitive graphical representations of the actions being taken, which suits our main goal of modeling and verifying security protocols.

Our contribution is three-folded: we define a DEL verification methodology; we give tool-support for this methodology, built on state-of-the-art tool-support for automata-based verification; and, by experimenting on two example protocols, we derive insights on the relative (dis)advantages of the process-driven and epistemic-driven approaches to the verification of anonymity.

In our DEL methodology, protocol analysis consists of describing action models corresponding to the protocol steps, automatically generating the final epistemic state by repeatedly applying the update product operation, and model checking epistemic formulas (expressed in PDL [24]) in this final state. Since the semantics of both epistemic states and action models is given by Kripke structures, the existing algorithmic and tool support for finite-state model checking of behavior specifications is immediately reusable in the DEL setting. In this paper, we specifically use the μCRL toolset [6] and the CADP model checker [13], together with LYS, a small new dedicated toolset [11] that supports the graphical visualization of the models and the execution of the update product.

We demonstrate the DEL modeling approach on Chaum's Dining Cryptographers protocol [8] and an abstraction of the FOO voting scheme [14]. The Dining Cryptographers protocol has been analyzed with many methods and tools, from OBDDS [29] to probabilistic verification [4]. According to [30], our earlier work [31] on DEL modeling and tooling compares favorably to [29], as well as [28]. FOO is a protocol that has recently received much attention. It has been formally analyzed using applied π-calculus supported by the ProVerif tool [22] and anonymity has been proved there manually. Other anonymity protocols, Onion Routing and Crowds, have been analyzed in an epistemic modeling framework in [15]. That framework is well tailored to specifying and verifying information hiding properties, although not yet automated. Epistemic modeling has also recently been applied to secrecy protocols in [20,23], but without the action models and the update product.

2 Dynamic Epistemic Modeling and Verification

A model for representing the state of knowledge of a group of agents is a Kripke structure with labels for the individual agents, and valuations for the states. It is common to call the states of the Kripke structure *worlds* and to refer to the structures as *epistemic models* [19,9,12] or *epistemic states*. The labels i are interpreted as the epistemic alternatives for agent i. If the models depict knowledge (as opposed to mere belief, or irrational preference), the label-relations $\xrightarrow{i}$ are equivalence relations, and what we get are so-called multimodal S5 Kripke models.

Definition 2.1 (Epistemic model) Let a set of propositional variables P and a finite set of agents Ag be given. An epistemic model is a triple $M = (W_M, V_M, R_M)$ where W_M is a set of worlds, $V_M : W_M \to \mathcal{P}(P)$ assigns a valuation to each world $w \in W_M$, and $R_M : Ag \to \mathcal{P}(W_M^2)$ assigns an accessibility relation $\xrightarrow{i}$ to each agent $i \in Ag$. A pair $\mathcal{M} = (M, U)$ with $U \subseteq W_M$ is an (multiple pointed) epistemic model, indicating that the actual world is among U.

The propositions in P model the facts of interest, for instance 'voter Alice voted yes', 'the coin we've just flipped is *head*'. The valuation function describes the possible worlds, by specifying which facts hold in every world. Take, for instance, the leftmost epistemic model in Figure 1. There, P is $\{p, q\}$ and the worlds are the four different possibilities for the truth of p and q ($\emptyset$, p, q, pq). One of them is the actual world (marked with a colored background), where p is false and q true. The epistemic accessibility relations $\xrightarrow{a}$, $\xrightarrow{b}$, $\xrightarrow{c}$ represent together the situation where agent a knows the actual value of p and agent b knows the actual value of q, while the third agent c knows nothing about either p or q.

More precisely, an *epistemic formula* $K_a\phi$ evaluates to *true* in a world if in that world every a-accessible world makes ϕ true. In the actual world of the example epistemic model, $K_a\neg p$ is true, for $\neg p$ is true in all worlds that are a-accessible from the actual world. So is $K_b q$ as well. On the other hand, $K_a q$ is false in the actual

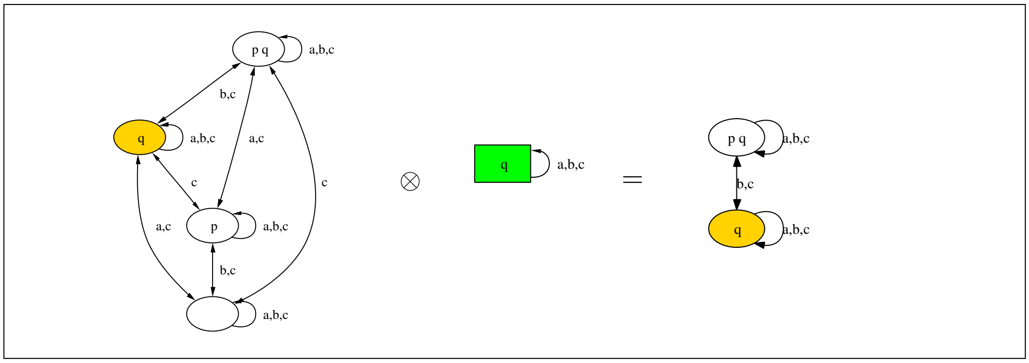


Figure 1. An epistemic model (left) updated with an epistemic action representing a public announcement (middle) transforms into a new epistemic model (right).

world. More subtly, $K_a(K_b q \vee K_b \neg q)$ is true in the actual world, for it happens to be the case that $K_b q$ is true in the actual world, and $K_b \neg q$ is true in the only other world that is a-accessible from the actual world. This illustrates how the epistemic models encode information about what agents know about the knowledge or ignorance of other agents. In the example, a does not know about q, but a knows that b knows whether q. Also, all agents know that c is ignorant about p and q.

Formally, the epistemic formulas are described in the convenient language of epistemic PDL, propositional dynamic logic [24,17,18] with the atomic actions interpreted as epistemic accessibilities:

$$\phi ::= \top \mid p \mid \neg\phi \mid \phi_1 \wedge \phi_2 \mid [\pi]\phi$$
$$\pi ::= a \mid ?\phi \mid \pi_1;\pi_2 \mid \pi_1 \cup \pi_2 \mid \pi^*$$

We use $\bot$ for $\neg\top$, $\phi_1 \vee \phi_2$ for $\neg(\neg\phi_1 \wedge \neg\phi_2)$, $\phi_1 \rightarrow \phi_2$ for $\neg(\phi_1 \wedge \neg\phi_2)$, $\langle\pi\rangle\phi$ for $\neg[\pi]\neg\psi$, $K_a\phi$ for $[a]\phi$. If $\{b_1 \cdots b_n\}$ is a finite set of agents, $[(b_1 \cup \cdots \cup b_n)^*]\phi$ is usually written as $C_{b_1 \cdots b_n}\ \phi$ and expresses that ϕ is common knowledge among the agents $\{b_1 \cdots b_n\}$. The truth definition for epistemic PDL defines the relations $M \models_w \phi$ and $[\![\pi]\!]^M$, by mutual recursion. Key clauses are:

$$\begin{array}{lll} M \models_w [\pi]\phi & \text{iff} & \text{for all } v \text{ with } (w,v) \in [\![\pi]\!]^M \\ & & \text{it holds that } M \models_v \phi \end{array}$$

$$\begin{aligned} [\![\pi_1;\pi_2]\!]^M &= [\![\pi_1]\!]^M \circ [\![\pi_2]\!]^M \\ [\![\pi^*]\!]^M &= ([\![\pi]\!]^M)^*, \end{aligned}$$

where $\circ$ denotes relational composition and * denotes transitive closure.

Intuitively, *epistemic updates* are acts of communication that change epistemic models. A breakthrough in epistemic logic occurred when [3] proposed to view epistemic updates as a kind of Kripke models, with the important difference that the worlds do not carry a valuation but a *precondition formula*. See also [1,3,2]. Formally, we define:

Definition 2.2 [Action models for a given language $\mathcal{L}$] Let a finite set of agents Ag and an epistemic language $\mathcal{L}$ be given. An action model for $\mathcal{L}$ is a triple $A = (W_A, \text{pre}, R_A)$ where W_A is a set of action states, $\text{pre} : W_A \rightarrow \mathcal{L}$ assigns a precondition to each action state, and $R_A : Ag \to \mathcal{P}(W_A^2)$ assigns an accessibility relation $\xrightarrow{i}$ to each agent $i \in Ag$. A pair $\mathbf{A} = (A, S)$ with $S \subseteq W_A$ is an action model, indicating that the actual action that takes place is a member of S.

We work with the epistemic language of PDL formulas. The simplest example of action model is the public announcement of some basic proposition. See, for instance, the middle Kripke structure of Figure 1, where the fact that q holds is publicly announced. More evolved examples are shown in Figure 2 and commented at the end of this section. It turns out that defining the result of updating an epistemic model with an action model as the *product* of the epistemic model and the action model, restricted in a suitable way to take the preconditions in the action model into account, has the desired effect. Formally:

Definition 2.3 [Update] Given an epistemic model (M, U) and an action model (A, S), we define the update $M \otimes A$ as (W', V', R') and the update $(M, U) \otimes (A, S)$ as $((W', V', R'), \{(u, s) \mid u \in U, s \in S, (u, s) \in W'\})$, where

$$\begin{aligned} W' &:= \{(w, s) \mid w \in W_M, s \in W_S, M \models_w \text{pre}(s)\}, \\ V'(w, s) &:= V_M(w), \\ (w, s) \xrightarrow{i} (w', s') \in R' &:\equiv w \xrightarrow{i} w' \in R_M \text{ and } s \xrightarrow{i} s' \in R_A. \end{aligned}$$

The language of PDL$^{\text{DEL}}$ (update PDL) is given by extending the PDL language with update constructions $[A, S]\phi$, where (A, S) is a (finite) action model. The interpretation of $[A, S]\phi$ in M, given w is: at all pairs (w, s) with $s \in S$ and w satisfying the precondition of s, ϕ is true in the update result $M \otimes A$. In the remainder of the paper, we will use the graphical representation of action models, rather than the textual one.

3 LYS - A Knowledge Analysis Toolset

In order to experiment with the idea of dynamic epistemic modeling and verification of protocols, we have built tool support for computing epistemic updates: a Haskell version (DEMO [31]) and a C version (LYS [11]). In this paper, only LYS has been used. Since Kripke models serve also as semantic models for finite-state systems, standard automata-based model checkers are applicable to epistemic model checking as well. We use the alternation-free μ-calculus model checker of CADP [13]. To gain access to this model checker, LYS uses the AUT file format [13] to represent the epistemic and action models.

We see anonymity as the intruder's inability to distinguish between the real situation and another possible one. Namely, between the actual world, where the agent that should remain anonymous is responsible for an event E, and a world where some other agent is responsible for E. Making a secret choice is expressed as setting (and thus 'knowing') the value of a propositional variable previously assigned to represent that choice. An anonymity violation amounts in this case to an unauthorized agent managing to deduce the value of that variable.

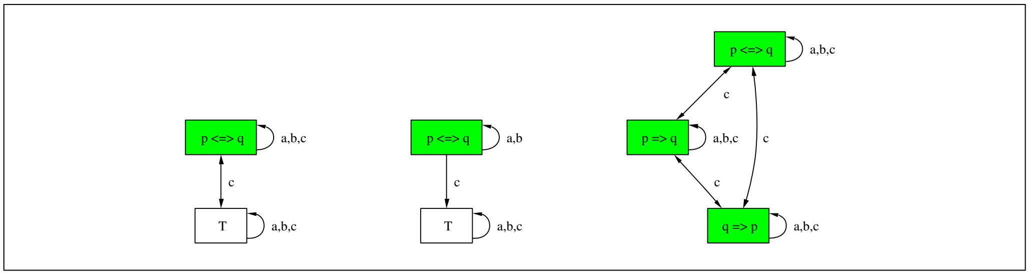


Figure 2. **The three action primitives, randomly instantiated.** A *group announcement* to a, b that $p \Leftrightarrow q$ holds, a *secret announcement* to a, b that $p \Leftrightarrow q$ holds and a *group announcement* to a, b that one *of the choices* expressed by the formulas $p \Rightarrow q, p \Leftarrow q, p \Leftrightarrow q$ has been taken. These primitives can be instantiated by using the LYS command `amnew` (generate a new action model) with the flags `-group`,`-secret` and `-choice`, respectively.

Anonymity is evaluated in an epistemic state (model), therefore it can now be seen as a reachability property. Checking whether anonymity holds means evaluating the respective knowledge formula in all epistemic states reachable from the initial epistemic state by update steps corresponding to communication steps of the protocol.

We consider the initial epistemic state as being the one containing no information, i.e., where there is one world for each valuation, all the indistinguishability relations form complete graphs, and all the worlds are marked actual. The rest of the modeling process consists of defining appropriate action models to represent the steps of the protocol as faithfully as possible. After that, the epistemic states of the protocol will be generated by means of repeated application of the update product operation. The required anonymity properties can then be verified by PDL model checking on the final epistemic states. Whenever a property is not met, a counterexample is provided, in the form of the valuation of an actual world w in which the property evaluates to false. Note that, since the knowledge grows monotonically, it is not needed to verify the intermediate epistemic states. But this is of course possible. See Sections 4 and 5 for examples and Section 6 for a further discussion on the epistemic verification process.

Building Faithful Actions Models

When modeling security protocols, translating the informal description of the protocol steps to action models can be a challenging and subjective task. But this is in fact always the case with modeling. Compared with other logical modeling approaches, action models have the advantage of being separated from the updates that they are expected to determine. They just have to faithfully represent the communication act that is taking place and all uncertainties regarding that act.

Action signatures [2] are abstract communication patterns, action models where the preconditions and labels on arrows are left unspecified. We identify three action signatures as being the most common: *group announcement*, *secret message* and *nondeterministic (private) choice*, shown in Figure 2 in some random instantiations. They are implemented as commands in LYS, and thus can be easily instantiated. In Figure 4, the group announcement and the choice primitives can be seen as they are instantiated for specific steps of the Dining Cryptographers protocol (also discussed in the next section). More elaborated actions should be constructed manually. See [2,31] for many examples and intuitions behind constructing complex action models.

The action model in Figure 1, representing a public announcement, corresponds to an usual *broadcast* message. Note that it is actually more expressive than broadcast, since the parties do not only learn the content of the message being broadcasted, but they also learn that all the other parties have heard the same message. The first action model from Figure 2 is a public communication to a group of agents (to agents a, b, in this case), and can be seen as a *secure multicast* or a *sending a message on a secure channel*. The actual action (indicated by the shaded square), is an update with $p \Leftrightarrow q$, but agent c confuses this with the trivial update $\top$ (an update where nothing gets communicated). Thus, agents a and b will learn $p \Leftrightarrow q$, but agent c will not. The second model is a secret announcement, where c's epistemic situation remains altogether unchanged, since he does not even notice that a communication is taking place. Finally, the rightmost one models a nondeterministic choice, made, in this particular case, by a and b together. c observes which are the choices, but not which one was made. This pattern can be used to model setting a variable, revealing a value to another agent, or simply branching the run of the protocol into more possible scenarios.

4 The Dining Cryptographers

Chaum's dining cryptographers protocol [8] is probably the most well-known example of a protocol where anonymity is the main requirement. The story goes as follows: three cryptographers have dinner together. At the end of the evening, they learn that the bill has been payed anonymously - by one of them, or by the NSA (the National Security Agency). They respect each other's right to anonymity, but they still wish to find out whether the payer was NSA or not. To achieve this, they come up with the following protocol: each pair of cryptographers generates a shared bit, by flipping a coin; then each cryptographer computes the exclusive or (XOR) of the two random bits she shares with the neighbors, then announces the result — or the flipped result, if she was herself the payer. The XOR of the three publicly announced results indicate whether the payer was an insider or not.

We will now show how the Dining Cryptographers protocol can be modeled and analyzed using the epistemic modeling framework introduced in Section 2. Figure 3 shows the LYS modeling script. The graphical representation of all action models used is given in the appendix. The intermediate epistemic states are too large to be viewed properly, but the very last one is readable and is shown in Figure 5.

For $i \in \{1, 2, 3\}$, let p_i be the proposition "cryptographer i is the payer". The goal of the protocol translates to: everybody should learn whether the proposition $p_1 \vee p_2 \vee p_3$ is true or not, but if it is true, nobody (except the payer herself) learns which of the propositions p_1, p_2, p_3 is true. To model the protocol, we need three more propositions q_1,q_2,q_3 to represent the result of flipping the coins shared by cryptographers 1 and 2, 2 and 3, 3 and 1, respectively. Let $\top$ represent "head" and

```
# prepare the action models
amnew   zero-one   -group   a,b,c   -form   ¬(p1 ∧ p2) ∧ ¬(p2 ∧ p3) ∧ ¬(p1 ∧ p3)
amnew   pay-a   -choice   a   -form   p1, ¬p1
amnew   pay-b   -choice   b   -form   p2, ¬p2
amnew   pay-c   -choice   c   -form   p3, ¬p3
amnew   flip-ab   -choice   a,b   -form   q1, ¬q1
amnew   flip-bc   -choice   b,c   -form   q2, ¬q2
amnew   flip-ca   -choice   a,c   -form   q2, ¬q3
amnew   asays   -choice   a,b,c   -form   p1 XOR (q1 XOR q3), ¬(p1 XOR (q1 XOR q3))
amnew   bsays   -choice   a,b,c   -form   p2 XOR (q1 XOR q2), ¬(p2 XOR (q1 XOR q2))
amnew   csays   -choice   a,b,c   -form   p3 XOR (q2 XOR q3), ¬(p3 XOR (q2 XOR q3))

# and the protocol is..
emnew   ALLWORLDS   a,b,c   p1,p2,p3,q1,q2,q3   ¬p2 ∧ ¬p3
emupdate   ALLWORLDS   zero-one   initial
emupdate   INITIAL   pay-a   pay-b   pay-c   AFTERPAYDECISION
emupdate   AFTERPAYDECISION   flip-ab   flip-bc   flip-ca   AFTERFLIP
emupdate   AFTERFLIP   asays   bsays   csays   DC3FINAL
```

Figure 3. Epistemic modeling of the Dining Cryptographers protocol with 3 participants a, b, c. The script makes calls to LYS routines for creating new action models (`amnew`), new epistemic models (`emnew`) and computing epistemic updates (`emupdate`). The argument `-choice` refers to the action pattern used, namely a nondeterministic choice between the alternatives presented after `-form`. The indicated names of agents are in all cases the agents receiving the update.

$\bot$ represent "tails".

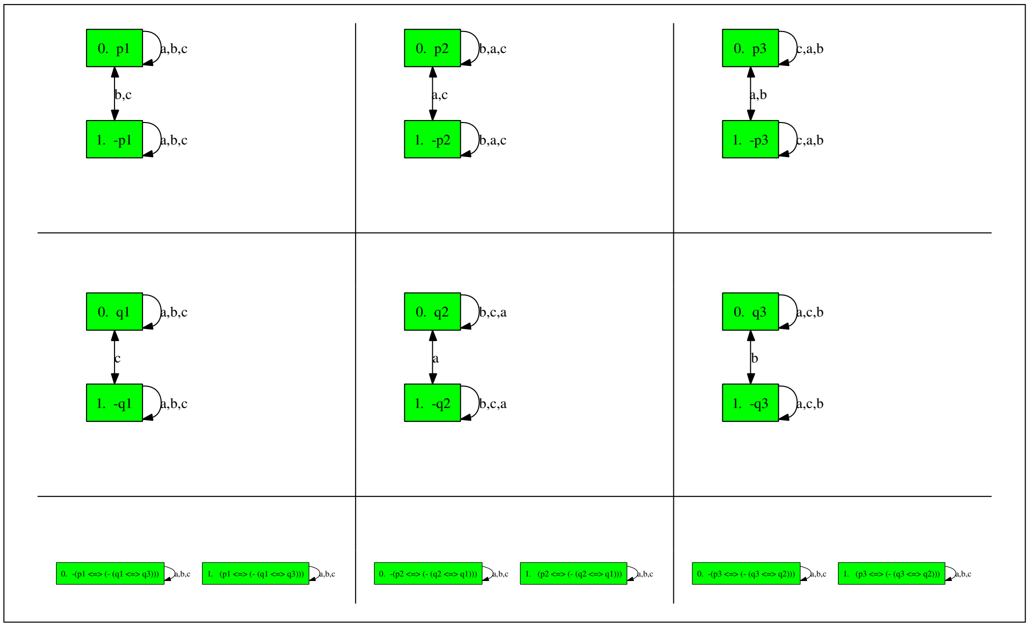


Figure 4. The action models used in the epistemic modeling of the Dining Cryptographers protocol. **Top**: *pay-a*,*pay-b*,*pay-c*. **Middle**: *flip-ab*, *flip-bc*, *flip-ca*. **Bottom**: *asays*, *bsays*, *csays*.

The protocol passes through four epistemic states (models): INITIAL, AFTERPAYDECISION, AFTERFLIP, DC3FINAL. The extra epistemic model ALLWORLDS represents the situation before the start of the protocol, when the rules have not yet been agreed upon. The transition to the start state INITIAL is made by the acknowledgment that at most one of the cryptographers pays, modeled by the public announcement *zero-one*. There are four possible paying scenarios: NSA pays, a pays, b pays or c pays. Since the cryptographers' names do not play a part in the protocol, we do not lose generality if we model only two scenarios: NSA pays, a

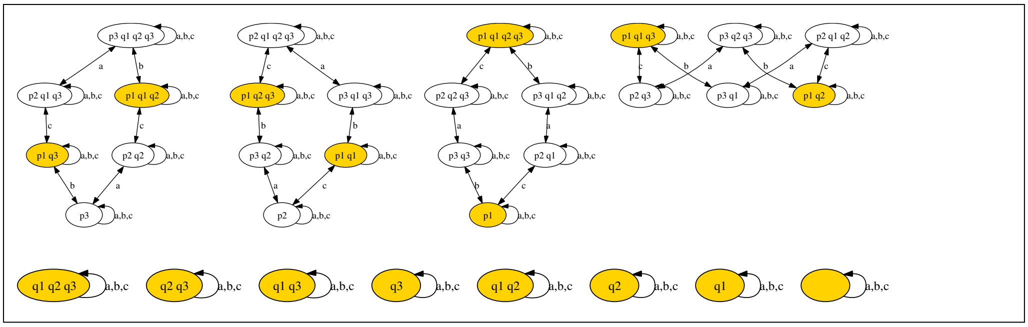


Figure 5. The final epistemic state (DC3FINAL). The two paying scenarios can be distinguished: a pays ($p_1 = \top$) on top; NSA pays, on bottom. The paying scenarios are combined with all coin flipping scenarios, given by different valuations for q_1,q_2,q_3.

pays. This fact is incorporated in the very first epistemic state, ALLWORLDS, by specifying that only worlds satisfying $\neg p_2 \wedge \neg p_3$ should be marked as actual. We do model all the flipping scenarios ($\bot, \bot, \bot$ up to $\top, \top, \top$), as seen from the fact that there is no restriction on q_1, q_2, q_3 imposed on the actual worlds. Thus, the two paying scenarios and the eight flipping scenarios will result in 16 possible situations, which are all evolving in parallel and are present in every epistemic state until the end. The rest of the worlds are there to represent the pieces of reality known to every agent.

The three big steps of the protocol — deciding who pays, flipping the coins, announcing the results — are modeled as epistemic updates and performed by the three `emupdate` calls marking the transition from INITIAL to AFTERPAYDECISION, then to AFTERFLIP, then to DC3FINAL. The first transition uses the action models *pay-a*, *pay-b*, *pay-c*, which are private announcements to a about the truth value of p_1, to b about p_2 and to c about p_3. The effect of *pay-a* on the current epistemic state is the deletion of all a-arrows between a world where $p_1 = \top$ and a world where $p_1 = \bot$. This makes the formula $K_a p_1 \vee K_a \neg p_1$ true, thus a has learned the value of p. Similar for *pay-b*, *pay-c*. In the original modeling of the protocol [26], there is a Master sending these announcements, in order to maintain the global restriction that at most one pays. We do not need to model the Master as an agent. Its role is fulfilled here by the *pay* announcements together with the public announcement *zero-one*. At further steps of the protocol, similar revealing announcements take place: *flip-ab* is the action model which reveals the value of q_1 to a and b, *asays* is the action model which computes the `XOR` of a's pay-bit and of a's two known coins. The update steps could be further refined, if the inspection of more intermediate epistemic states (for instance, the epistemic state after the coin between a and b has been flipped) is desired.

The first attempt in checking anonymity could be to check that a cryptographer can never learn whether another cryptographer has paid or not. That is, to check the epistemic formula

$$\neg(K_a p_2 \vee K_a \neg p_2) \wedge \neg(K_a p_3 \vee K_a \neg p_3) \wedge \neg(K_b p_1 \vee K_b \neg p_1) \wedge$$
$$\neg(K_b p_3 \vee K_b \neg p_3) \wedge \neg(K_c p_1 \vee K_c \neg p_1) \wedge \neg(K_c p_2 \vee K_c \neg p_2).$$

The model checker disproves this property and gives as counterexample a valuation where all of p_1, p_2, p_3 are false. Indeed, in the case where NSA pays, the cryptographers do learn that all of p_1, p_2, p_3 are false. Moreover, when one of the cryptographers pays, the payer will know that the other two did not pay, even from the very beginning. A more realistic anonymity requirement is that if $p_1 \vee p_2 \vee p_3 = \top$ then b and c do not know that a was the payer. We can express this as the epistemic formula $p_1 \Rightarrow \neg K_b p_1 \wedge \neg K_c p_1$, which does get a positive answer when checked on DC3FINAL.

Note that, due to the public announcement that at most one cryptographer is paying, the anonymity of non-payers is only guaranteed from the perspective of another non-payer. The paying cryptographer will know that the other two are not paying, and will also be able to deduce what the third flipped coin was:

$$p1 \Rightarrow K_a \neg p_2 \wedge K_a \neg p_3 \wedge (K_a q_2 \vee K_a \neg q_2)$$

Moreover, it is possible to verify that the payer herself knows that she is anonymous:

$$p_1 \Rightarrow K_a(\neg K_b p_1 \wedge \neg K_c p_1),$$

that the non payers know that the payer is aware of her anonymity,

$$p_1 \Rightarrow K_b K_a(\neg K_b p_1 \wedge \neg K_c p_1), \quad p_1 \Rightarrow K_c K_a(\neg K_b p_1 \wedge \neg K_c p_1),$$

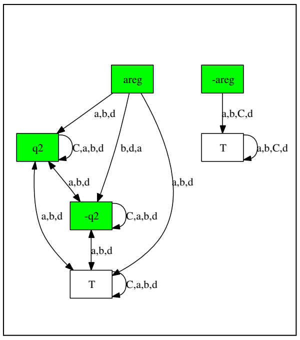
and actually that the payer's anonymity is *common knowledge* among the three cryptographers:

$$p_1 \Rightarrow C_{a,b,c}(\neg K_b p_1 \wedge \neg K_c p_1).$$

This protocol generalizes naturally to N cryptographers, and a similar analysis can be performed for $N = 4$ and $N = 5$. In these cases, we can even talk about *coalitions* of malicious cryptographers. We do this by using the *distributed knowledge* operator [12]. It is impossible to express this operator with PDL modalities and is also inconvenient from a logical perspective, but we introduce it here in order to be able to express properties of conniving coalitions. We say that a group of agents G have distributed knowledge about ϕ (denoted $D_G\phi$) if they know ϕ when they pool their individual knowledge. In terms of Kripke structures:

$$(M, s) \models D_G\phi \text{ iff } (M, t) \models t \text{ for all } t \text{ s.t. } (s, t) \in \bigcap_{a \in G} R_a.$$

We can now express the possibility that two cryptographers pool their secret knowledge and are able to learn who was the payer. After generating the final epistemic state for $N = 4$, we can employ the CADP model checker, together with a trick that we will not explain here, to verify that $p_1 \Rightarrow D_{b,d} p_1$ holds. In other words, (in DC4) a may remain anonymous to b, c, d individually, but if they decide to share their secrets, then the anonymity of a is compromised. In DC5 as well, two conniving cryptographers are able to violate payer's anonymity if they are both her direct neighbors. That is, the epistemic formula $p_1 \Rightarrow D_{b,e} p_1$ holds.



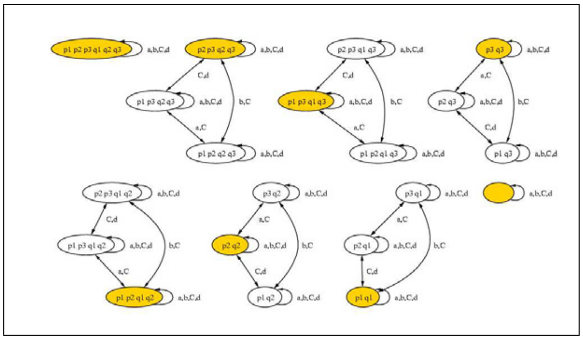


Figure 6. **Left**: the action model for "if registered to vote, voter a sends her vote to the counter C" **Right**: The final epistemic state of the voting protocol, if the three phases are separated.

Besides anonymity, we can of course also check that the protocol meets its goal, that is all agents learn whether $p_1 \vee p_2 \vee p_3$ holds, and that is even common knowledge:

$$C_{a,b,c}(p_1 \vee p_2 \vee p_3) \vee C_{a,b,c}(\neg p_1 \wedge \neg p_2 \wedge \neg p_3).$$

5 An Electronic Voting Protocol

The so-called FOO protocol proposed by Fujioka, Okamota and Ohta [14] is a rather complex voting scheme that guarantees anonymity of the voters with respect to any conspiracy of administrator, collector and external parties. In summary, it runs as follows: By interacting with the administrator and making use of a symmetric key, a nonce and a special property of the digital signing scheme, each voter obtains a so-called signed covered vote. The voter then sends anonymously her signed covered vote to the collector, who publishes it. After the voter has verified that her covered vote appears on the published list, she sends, again anonymously, the symmetric key and covered vote to the collector. The collector can now open it and add it to the clear votes list, that will be made public as soon as everybody has voted.

An anonymity analysis of an abstract version of the FOO voting scheme can be conducted in the dynamic epistemic setting of this paper. For lack of space, we will not explain the whole modeling, but only point out several interesting aspects, which could not be observed in the modeling of the Dining Cryptographers protocol. The abstractions we make are as follows: we ignore the administrator, whose main role is to prevent ineligible and double voting and plays no role for anonymity; we consider yes/no votes only; although the covering aspect can be modeled, we leave it out for simplicity of presentation. The voters are modeled as agents a, b and d, the collector as agent C. The choices of the voters are represented by the propositions p_1, p_2, p_3, meaning 'a votes yes', 'b votes yes', 'd votes yes', respectively . The list of clear votes that will eventually be hold by the collector has three positions, to which we refer by the propositions q_1, q_2, q_3: q_1 means 'the first vote in the list is yes', q_2 means 'the second vote in the list is yes', q_3 means 'the third vote in the list is yes'. We also need three propositions *areg*, *breg*, *dreg* to mark that the first (registering) phase ended for a, b, d, respectively. Then the phases of the protocol are:

- a,b,d register to vote. This includes the interaction with the administrator and sending the covered vote to C. We model this by public announcements of *areg*, *breg*, *dreg*.
- a, b, d finalize their vote, by sending their symmetric key to C, and therefore letting C add a new clear vote to its list, without learning who the respective voter is. The effect of this phase is that q_1, q_2, q_3 will be revealed to C. An interesting point here is that we need to condition this phase by the completion of the previous one. Although we did not model the covered votes explicitly, we do assume that they should be sent to the collector and we keep track of whether this happened or not through *areg*, *breg*, *dreg*. So, the action model that we need in this phase for each voter is 'if the voter registered, she (anonymously) reveals her clear vote to C, otherwise nothing happens'. This had to be designed manually, since there is no simple pattern for it (Figure 6).
- C makes the result of the vote public and everybody learns the values of q_1,q_2,q_3.

There are several facts relevant to the protocol, that everybody is aware of (actually, they are common knowledge) and modeled as public announcements. One of them is that the result of the voting is a permutation of the participants' choices. That is, any permutation of (q_1, q_2, q_3) could be equal to (p_1, p_2, p_3). Since the six scenarios are symmetric, we only model one of them, say $p_1 = q_2 \wedge p_2 = q_3 \wedge p_3 = q_1$. This is expressed by marking as actual worlds in the initial epistemic state precisely those worlds which meet the condition above. Note that, essentially, this symmetry-reduction is made outside the protocol, that is a, b, d still consider that all the permutations might be possible. Another common knowledge fact is that, if C has learned a clear vote, then there must have been a voter who sent it. For the clear vote q_1, we express this as:

$$K_C q_1 | K_C \neg q_1 \Rightarrow (areg \wedge (p_1 \Leftrightarrow q_1)) | (breg \wedge (p_2 \Leftrightarrow q_1)) | (dreg \wedge (p_3 \Leftrightarrow q_1)).$$

We write the similar formulas for q_2 and q_3 and announce them all publicly.

In the analysis of FOO performed by Kremer&Ryan [22], it has been observed that the three phases should be strictly separated, otherwise anonymity is compromised. We are able to detect this problem in our setting too, if we allow updates to be performed in any order and thus look at more than one trace. Indeed, in the trace: a registers, a votes, b registers etc., the formula $(areg \wedge \neg breg \wedge \neg dreg) \Rightarrow (K_C p_2 | K_C \neg p_2)$ becomes true, meaning that a's choice has leaked to the collector. In the 'right' trace, when the phases are separated, after performing all updates, the epistemic state shown in Figure 6 is reached, where anonymity of a voter (for instance, a) with respect to the other voters can be expressed as

$$\neg(K_b p_1 \vee K_b \neg p_1) \wedge \neg(K_c p_1 \vee K_c \neg p_1).$$

This formula is not confirmed by the model checker, because it is too general. It does not hold in *all* actual worlds, and thus a number of counterexamples are given. For instance, when all votes are 'yes', the anonymity is not preserved and it even becomes common knowledge that this is the case.

But more specific formulas are proved correct by the model checker. For instance, the one saying that the vote of a remains secret to b under the assumption that c voted differently:

$$p_1 \neq p_3 \implies \neg(K_b p_1 \vee K_b \neg p_1).$$

6 Concluding Discussion

We presented a tool-supported epistemic verification framework, built on recent work from Dynamic Epistemic Logic [2]. The central new elements are the *action models*, which give an elegant and compact representation for communication primitives, and the *update product* operation, which allows mechanical execution of epistemic updates, therefore bringing the epistemic specification closer to a behavioral specification. We showed that a traditional verification toolset can be used to verify a nontraditional and challenging property such as anonymity. We demonstrated the technique on two examples of protocols guaranteeing anonymity.

Verification of Anonymity

Due to the fact that epistemic states encode indistinguishability relations, this framework is especially suitable to specification and verification of anonymity properties. There are two major advantages when comparing to the process-based verification approach: no remodeling of the protocol is necessary if the capabilities of the (passive) observer change; once generated the final epistemic state of the protocol, every required anonymity property is verifiable by model checking, which is much faster than computing trace equivalence. Compared to other epistemic logic approaches, we believe that the action models are a more natural modeling mechanism for the protocol steps, and that the update product is very well suitable to mechanization.

	Process state spaces			**DEL models**		
	anonymity properties	states	transitions	epistemic states	worlds	arrows
DC-3	32-1	184	362	initial	64	6 144
	3-12	139	255	after flip	32	259
				after announce	32	116
DC-4	24-13	910	2160	initial	256	131 072
	234-1	1083	2673	after flip	128	5 088
	34-12	886	2096	after announce	128	864
DC-5	12345-	6615	19681	initial	1024	2 621 440
	245-13	5189	14679	after flip	192	16 992
				after announce	192	1 440

Figure 7. *Sizes of the Kripke structures representing process state spaces and epistemic models, respectively, for several instances of the DC protocol.* The "anonymity properties" column refers to the different process models, and thus different state spaces, generated for various coalitions of dishonest participants. $24 - 13$ means that $2, 4$ are honest and $1, 3$ dishonest; $12345-$ means that all five are honest. The given sizes are for the completely generated state spaces. On the DEL models side, no distinction is necessary for various coalitions, since all corresponding anonymity properties can be verified on the same final epistemic model "after announce". Note that this final model is always smaller than the state spaces generated from process specifications. However, the initial DEL model is much larger and this is actually one of the bottlenecks of the epistemic modeling. Epistemic-specific optimizations on the model representation are possible, but for now the same formats as for process specifications are used. On an average desktop machine, Kripke structures of up to ca. 50 million transitions/arrows can be handled.

DEL Modeling vs. Process Modeling

Figure 7 shows a comparison between the process and DEL models of the DC protocol. The process specification has been written in the μCRL language and is explained in [10]. The generated state spaces are in the same AUT format mentioned in Section 3. Taking into account the two DC specifications (μCRL and DEL) and the sizes of the state spaces, respectively epistemic models obtained, we have to confirm the general opinion that process languages are better suited to protocol modeling than the logic ones. Processes can be specified rather detailed, thus achieving a realistic representation of the particular protocol. Epistemic logical specifications are more abstractly geared and require some effort for constructing faithful models. However, epistemic logical modeling allows for a richer analysis of the protocol states, giving more insight into subtle effects of communication. Common knowledge, for instance, is impossible to express on a plain process specification. Moreover, action models are very promising building blocks. Due to their precise semantic model, it seems possible to define a formal translation from process steps to action models and thus getting epistemic specifications from behavioral ones, much in the line of [21].

DEL Modeling Limitations

As we have seen in the modeling of the FOO protocol, it is not always clear how communication acts can be used to represent security ingredients like keys, nonces, encryption. Some steps towards properly modeling these have been taken in previous studies [20,5,32].

Future Work

One of the features of our epistemic modeling toolset is that it can be integrated with control flow commands offered by programming shells. This allows, possibly, a more fine-grained description of the protocols, in particular the branching aspects. More interestingly, this allows generation of counterexamples in terms of steps of the protocol executed until the bad epistemic state. We plan to investigate this aspects further. Also, we plan to extend the three primitives discussed in Section 3 to real epistemic modeling guidelines that will help translating informal descriptions of protocol steps into action models. Ideally, we would also be able to define a formal correspondence between process specifications and epistemic updates. In the more general context of verifying arbitrary security properties, we aim at a framework where the advantages of specifying functional aspects with process modeling and informational aspects with epistemic logic can be combined. The fact that the two techniques are complementary and they should, ideally, be combined, has already been recognized and various formalisms to achieve this have been proposed [21,16]. However, they are quite complex. We think that the update product operation might be able to bridge the gap in a more straightforward way, by allowing to represent the behavior of a system as a 'higher-order' Kripke structure, with epistemic models as states and action models as transition labels.

Acknowledgement

We are grateful to Erik de Vink for comments on a previous version of this document.

References

[1] Baltag, A., *A logic for suspicious players: epistemic action and belief-updates in games*, Bulletin of Economic Research **54** (2002), pp. 1–45.

[2] Baltag, A. and L. S. Moss, *Logics for Epistemic Programs*, Synthese **139** (2004), pp. 165–224.

[3] Baltag, A., L. S. Moss and S. Solecki, *The logic of public announcements, common knowledge, and private suspicions*, Technical report, Dept. of Cognitive Science, Indiana University and Dept. of Computing, Oxford University (2003).

[4] Bhargava, M. and C. Palamidessi, *Probabilistic Anonymity*, in: *Proceedings CONCUR'05*, LNCS **3653**, 2005, pp. 171–185.

[5] Bleeker, A. and J. van Eijck, *Epistemic Action and Change*, in: *LOFT-4 Proceedings*, 2000.

[6] Blom, S. C. C., W. J. Fokkink, J. F. Groote, I. van Langevelde, B. Lisser and J. C. van de Pol, *μCRL: A Toolset for Analysing Algebraic Specifications*, in: *Proc. CAV'01*, LNCS **2102**, 2001, pp. 250–254.

[7] Burrows, M., M. Abadi and R. Needham, *A Logic of Authentication*, in: *Practical Cryptography for Data Internetworks*, IEEE Computer Society Press, 1996 Reprinted from the Proceedings of the Royal Society, volume 426, number 1871, 1989.

[8] Chaum, D., *The dining cryptographers problem: unconditional sender and receiver untraceability*, Journal of Cryptology **1** (1988), pp. 65–75.

[9] Chellas, B. F., "Modal Logic: An Introduction," Cambridge University Press, 1980.

[10] Chothia, T., S. M. Orzan and J. Pang, *Automatically Checking Anonymity Using Distributed mCRL* (2006), under submission. Draft available from http://www.win.tue.nl/~sorzan/anonmcrl.pdf.

[11] CWI, *LYS* (2005), available from http://www.win.tue.nl/~sorzan/lys/.

[12] Fagin, R., J. Y. Halpern, Y. Moses and M. Y. Vardi, "Reasoning about Knowledge," MIT Press, 1995.

[13] Fernandez, J.-C., H. Garavel, A. Kerbrat, L. Mounier, R. Mateescu and M. Sighireanu, *CADP – a protocol validation and verification toolbox*, in: *Proceedings CAV'96*, LNCS **1102**, 1996, pp. 437–440.

[14] Fujioka, A., T. Okamoto and K. Ohta, *A Practical Secret Voting Scheme for Large Scale Elections*, in: *ASIACRYPT*, LNCS **718**, 1992, pp. 244–251.

[15] Garcia, F. D., I. Hasuo, P. van Rossum and W. Pieters, *Provable Anonymity*, in: *Proceedings of the ACM Workshop on Formal Methods in Security Engineering (FMSE'05)* (2005), pp. 63–72.

[16] Halpern, J. Y. and K. R. O'Neill, *Anonymity and Information Hiding in Multiagent Systems*, Journal of Computer Security (2005), pp. 483–514.

[17] Harel, D., *Dynamic Logic*, in: D. Gabbay and F. Guenther, editors, *Handbook of Philosophical Logic Volume II — Extensions of Classical Logic*, Reidel, 1984 pp. 497–604.

[18] Harel, D., D. Kozen and J. Tiuryn, "Dynamic Logic," MIT Press, 2000.

[19] Hintikka, J., "Knowledge and Belief: An Introduction to the Logic of the Two Notions," Cornell University Press, Ithaca N.Y., 1962.

[20] Hommersom, A., J.-J. Meyer and E. P. de Vink, *Update Semantics of Security Protocols*, Synthese **142** (2004), pp. 229–267.

[21] Hughes, D. and V. Shmatikov, *Information Hiding, Anonymity and Privacy: A Modular Approach*, Journal of Computer Security **12** (2004), pp. 3–36.

[22] Kremer, S. and M. D. Ryan, *Analysis of an Electronic Voting Protocol in the Applied Pi-Calculus*, in: *Programming Languages and Systems – Proceedings of the 14th European Symposium on Programming (ESOP'05)*, LNCS **3444**, 2005, pp. 186–200.

[23] Nirmal, S., "Verification of Security Protocols: tool support for Update Semantics," Master's thesis, Technical University Eindhoven (2005).

[24] Pratt, V., *Application of modal logic to programming*, Studia Logica **39** (1980), pp. 257–274.

[25] Raimondi, F. and A. Lomuscio, *A Tool for Specification and Verification of Epistemic Properties in Interpreted Systems*, Electronic Notes in Theoretical Computer Science **85** (2004).

[26] Schneider, S. and A. Sidiropoulos, *CSP and Anonymity*, in: E. Bertino, H. Kurth, G. Martella and E. Montolivo, editors, *Proc. ESORICS'96*, LNCS **1146**, 1996, pp. 198–218.

[27] van Benthem, J., J. van Eijck and B. Kooi, *Logics of Communication and Change*, Information and Computation (2006), to appear.

[28] van der Meyden, R., *Common Knowledge and update in finite environments*, Information and Computation **140** (1998), pp. 115–157.

[29] van der Meyden, R. and K. Su, *Symbolic Model Checking the Knowledge of the Dining Cryptographers*, in: *Proc. CSFW'04, Pacific Grove* (2004), pp. 280–291.

[30] van Ditmarsch, H., 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*, ENTCS **149(2)**, 2006, pp. 105–123.

[31] van Eijck, J., *DEMO program and documentation* (2005), available from http://www.cwi.nl/~jve/demo/.

[32] van Eijck, J. and S. M. Orzan, *Modelling the Epistemics of Communication with Functional Programming*, in: M. van Eekelen, editor, *Sixth Symposium on Trends in Functional Programming TFP'05*, Institute of Cybernetics, Tallinn Technical University, Tallinn, 2005, pp. 44–59.