View metadata, citation and similar papers at core.ac.uk
brought to you by CORE
provided by CWI's Institutional Repository

Towards Model Checking Cryptographic Protocols with Dynamic Epistemic Logic

Malvin Gattinger[1] and Jan van Eijck[2]

[1] ILLC, Amsterdam, The Netherlands, malvin@w4eg.de
[2] ILLC and CWI, Amsterdam, The Netherlands, jve@cwi.nl

Abstract. We present a variant of Kripke models to model knowledge of large numbers, applicable to cryptographic protocols. Our Epistemic Crypto Logic is a variant of Dynamic Epistemic Logic to describe communication and computation in a multi-agent setting. It is interpreted on register models which efficiently encode larger Kripke models. As an example we formalize the well-known Diffie-Hellman key exchange. The presented register models also motivate a Monte Carlo method for model checking which we compare against a standard algorithm, using the key exchange as a benchmark.

1 Introduction

Cryptographic protocols deal with the knowledge of secrets which can usually be represented as numbers. The established semantics for logics of knowledge are Kripke models, consisting of a set of possible worlds on which each agent is assigned a relation that represents her knowledge. Our question is whether we can use such semantics to analyze cryptographic protocols and — as a first step — the knowledge of numbers. Thus, our work is in the tradition of Dynamic Epistemic Logic (DEL) [1,2] rather than process logic [18].

Our approach differs from existing methods for model checking protocols based on interpreted systems [3,20] in which possible states are connected by a temporal relation or transition function. In contrast, our Kripke models are purely epistemic: Relations between possible worlds represent the knowledge of agents. Time is not part of our models but only given implicitly by commands and their interpretation as action models. Another difference is that many frameworks, including [7] which is also based on DEL, introduce cryptographic primitives like keys or nonces to the syntax and semantics. This is not necessary here, we only need numeric variables.

The paper is structured as follows. In Section 2 we argue that ordinary Kripke semantics are not suited for modeling ignorance of large numbers because the models quickly become too large. We therefore introduce register models which only distinguish one actual value of a variable from all other possibilities. Section 3 presents the syntax and semantics of Epistemic Crypto Logic, a language that is able to capture directed communication and computation and is interpreted on register models. We then describe a Monte Carlo method for model checking in

Section 4 and formalize a cryptographic protocol from the literature in Section 5. Section 6 concludes with model checking results and suggestions for further research.

2 Register Models

To introduce the idea of register models which was first presented in [9], consider the following number guessing game, played between Jan and his twin daughters Gaia and Rosa. Jan says: "I have a number in mind, in the range from one to ten. You may take turns guessing. Whoever guesses the number first wins." Gaia and Rosa agree and after a number of rounds, Jan announces: "Rosa, you have won."

A naive representation of this game can be given as a multi-agent Kripke model with ten possible worlds. In Figure 1a the actual world — where the number is 6 — is indicated by a box, and dashed lines represent the ignorance of the twins. Note that if Jan and the twins were to play the game with numbers up to 100 our model would become much larger. Now suppose the following exchange takes place: Gaia: "Eight?" ... Jan: "No". This results in an update of the model: The possibility 8 drops out, and this is common knowledge among the twins because they both heard Jan's reply. The result can be seen in Figure 1b.

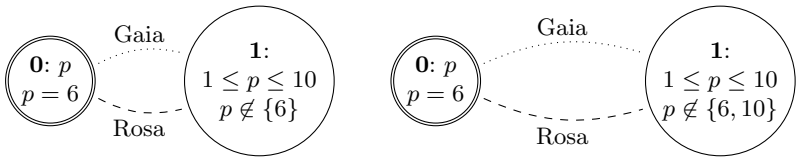
(a) Beginning of the game. (b) After a first wrong guess.

Fig. 1: A naive representation of the guessing game.

And so on until one of them guesses the correct number. But at some point the twins start to complain, and refuse to play the guessing game in this way: "How can we know you are not cheating on us? Please write down the number, so you can show it to us afterwards as a proof." When the secret number gets *written down*, the important notion of a *register* arises. It allows Jan to prove that he really knew the number because he had fixed it beforehand — and did not just accept Rosa's to make her win.

So what is it that Jan knew when we say that he knew the number? Let us say: Jan can see the difference between a register with *the correct number* written in it, and the same register with *some different number* written on it. If someone else would change the number on the piece of paper, Jan would spot it, but Gaia and Rosa would not.

To represent the game in a Kripke model it therefore suffices to distinguish two possibilities: In the actual world the register p has the value 6 and in the



(a) Beginning of the game. (b) After a first wrong guess.

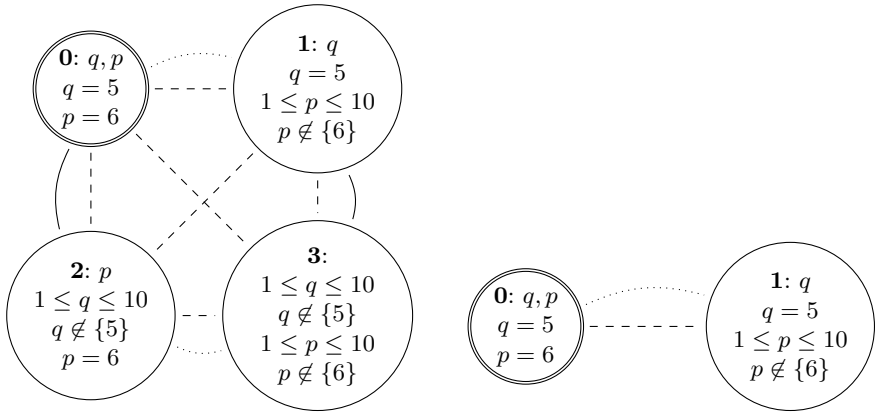
Fig. 2: Modeling the guessing game with register models.

other world it can be anything else in the range which was agreed upon earlier. Jan knows p, the two children do not know p. This leads to the register model in Figure 2a which can be seen as a translation of Figure 1a. Here **0** is a world that has register p, with number 6 stored in it, while **1** differs from **0** in that it also has this register, but with all the possible values in it that differ from 6. Gaia (dots) and Rosa (dashes) do not have access to the register.

Suppose the register model from Figure 2a gets updated with Gaia: "Ten?" and Jan: "No". Then 10 drops out of the range of p and we obtain the model shown in Figure 2b. If we would now go on with Rosa: "Six?" and Jan: "Yes", the model would be restricted to world **0**.

Also interesting are incorrect guesses. Figure 3a shows the moment when Gaia prepares to announce a guess but has not yet revealed it. She introduced the new register q and knows that its value is 5, but Jan (solid) and Rosa (dashed) do not know it yet. If Gaia reveals her guess by means of the announcement $q = 5$, the model of Figure 3a changes into that of Figure 3b. Jan can now state that the guess is wrong, by means of the announcement $p \neq q$. The result is a model just like Figure 3b, except for the constraint $p \notin \{5, 6\}$ at **1**.

Without registers, Figure 3a blows up to a model with 100 worlds. More generally, to model k registers, each of m bits to allow numbers between 0 and 2^m, we get a blow-up from 2^k to $(2^k)^m$ possibilities.



(a) Gaia is preparing her guess. (b) Gaia revealed her guess.

Fig. 3: Models with two registers.

A sound and complete axiomatization of the resulting *Guessing Game Logic* can be found in [10] and a detailed discussion in [12]. Here we will directly present an extended framework for which [10] only gives the syntax and an example.

3 Epistemic Crypto Logic

The idea of register models introduced in the previous section can also be used to analyze cryptographic protocols. In this section we define *Epistemic Crypto Logic*, a dynamic language to describe knowledge, communication and computation.

3.1 Syntax

Definition 1. *Let i range over a finite set of agents I, p over a set of propositions and N over $\mathbb{N}$. The language of epistemic crypto logic extends the language of guessing games and consists of the following formulas, commands and expressions.*

$$\phi ::= \top \mid p \mid L_i \mid p = E \mid \neg\phi \mid \phi \wedge \phi \mid K_i\phi \mid [C]\phi \mid \mathsf{Prime}E \mid \mathsf{Coprime}EE$$
$$C ::= p \leftarrow^i E \mid ?\phi \mid !p \mid !p = p \mid !p = N \mid !p \neq p \mid !p \neq N \mid \mathsf{Open}_i \mid \mathsf{Close}_i$$
$$E ::= p \mid N \mid E + E \bmod E \mid E \times E \bmod E \mid E^E \bmod E$$

Besides atomic propositions and standard boolean connectives we have L_i meaning that agent i is currently listening to announcements, $p = E$ saying that register p and expression E have the same value, $K_i\phi$ meaning that i knows ϕ and $[C]\phi$ saying that after any execution of C, ϕ will hold. The connectives Prime and Coprime say that the value of an expression is prime and that the values of two expressions are coprime, respectively.

The command $p \leftarrow^i E$ creates a new register p only known to agent i with the value of E in it. The intended meaning of the other commands are testing whether a formula holds, announcement of propositions, equalities and inequalities and opening and closing channels to modify the listener set, explained in Section 3.2.

Expressions can be registers, natural numbers and modular addition, multiplication and exponentiation of other expressions. A motivation for this selection of operators is given in Section 3.3.

The precise semantics for the full language are then given in Section 3.4.

3.2 Communication: Local Listening

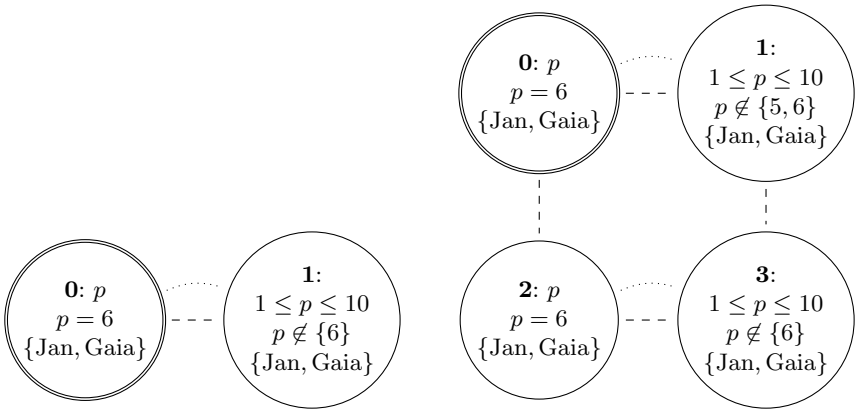
In the guessing game announcements always reach all the agents. For cryptographic protocols we usually want to model more complex situations, e.g. communication via private channels or eavesdropping. Hence we use the following idea from [8]. Our models include a local sets of listeners: For each $w \in W$ there is a $L_w \subseteq I$ describing who is listening to announcements in this world w.

This also allows us to model different knowledge about who is listening. Alice and Bob might believe that they are communicating privately when actually Eve

is spying on them. In fact local listener sets are too general and we therefore follow [8] in adding a constraint that all agents are self-aware about their attention: Everyone should know whether they are listening or not. Formally, this boils down to $\forall i \in I : \forall s, t \in W : (R_i st \to (i \in L_s \leftrightarrow i \in L_t))$.

The listener set should also determine what happens when new information is announced. Hence our interpretations of announcements are not just model restrictions as known from public announcement logic. Instead, modifications of the model are done on copies of the previous worlds. Then we update the knowledge relations such that all listeners can distinguish these new worlds from their originals but everyone who was not paying attention still confuses them.

For example, suppose during the game Rosa gets distracted. In Figure 4a only Jan and Gaia are listening (and everyone knows this). Now suppose $p \neq 5$ is announced. Since Rosa is not listening, the model changes into that of Figure 4b. Only Gaia learned something, while Rosa (dashed) did not get any new information.



(a) Rosa is not listening.

(b) Rosa did not learn anything.

Fig. 4: Announcement with local listeners

To explain the remaining commands, we show how a whole act of communication can be encoded in our language. To say that "a sends ϕ to b", the whole sequence of commands is: $?K_a\phi$; Open_b; $!\phi$; Close_b.

- $?K_a\phi$ tests whether $K_a\phi$ is true in the actual world: The agent a can only communicate ϕ if she knows it.
- Open_b adds b to the set of agents that are listening. Besides b herself, exactly those who were already listening will know that b is now paying attention.
- $!\phi$ is the announcement of ϕ. For simplicity we only allow announcements of atomic propositions and equalities and inequalities, so ϕ is of the form p, $p = E$ or $p \neq E$ where E is a register or natural number. This means announcements like $!p = q + q \mod 23$ are not allowed. However, they can be modeled by creating a new register r with the value $q + q \mod 23$ and then announcing $!p = r$.

– Close_b removes b from the set of agents that are listening. Again, this will be known by b and everyone else who is listening, but nobody else.

We note that, as far as b is concerned, the information ϕ could come from anywhere, which means our models provide *no authentication*. Furthermore, also anyone besides b who is listening receives ϕ, i.e. the channel is *not secret*.

3.3 Computation: Fast Modular Arithmetic

We also want to model feasible computation, in order to refine the meaning of "knowing a number" to two different conditions that can both be checked on register models: I know (i) the numbers that I can look up in an accessible register, and (ii) the numbers that I can feasibly compute from numbers I know.

A criterion for what is feasible is the existence of fast algorithms. Those exist for primality testing (e.g., the probabilistic Miller-Rabin test [15,17]), co-primality testing and finding modular inverses (Euclid's extended GCD algorithm) as well as addition, multiplication and exponentiation modulo (see [4,16]). We therefore include primality testing and a fragment of modular arithmetic in our language.

Note that by not including other operations we implicitly import common articles of faith from public key cryptography, that for example factorization and discrete logarithm are *not* feasible.

3.4 Semantics

Definition 2. *A* crypto model *is a tuple* $\mathcal{M} = (W, \mathcal{R}, V)$ *where*

– $(W, \mathcal{R})$ *is a multi-agent* S5 *frame for* I,
– V *is a valuation function for some* $Q \subseteq \mathbf{P}$ *(the global set of used variables): It assigns to each world* $w \in W$ *a valuation* $(P_w, L_w, f_w, C_w^+, C_w^-)$ *where*
  - $P_w \subseteq \mathbf{P}$ *(the basic propositions true at* w*),*
  - $L_w \subseteq I$ *(the agents listening at* w*) satisfying self-awareness: For all agents* i *and all* v *and* w *such that* vR_iw *we have* $i \in L_v$ *iff* $i \in L_w$.
  - f_w *is a function on* Q *that assigns to each* $q \in Q$ *a triple* (n, m, X) *(the range of* q *at* w*) with* $n, m \in \mathbb{N}$, $n \leq m$, $X \subseteq \mathbb{N}$, *such that:*
    (i) if $q \in P_w$ *then for* $f_w(q) = (n, m, X)$ *we have* $n = m$ *and* $X = \varnothing$
    (ii) if $q \in P_v \cap P_w$ *for* $v, w \in W$ *then* $f_v(q) = f_w(q)$
  - $C_w^+ \subseteq Q^2$ *and* $C_w^- \subseteq Q^2$ *(the in/equality constraints of* w*) such that no* $(p, q) \in C_w^-$ *is in the transitive symmetric reflexive closure of* C_w^+.

Given a model, we also refer to the reflexive-transitive closure of the union of all relations by $\mathcal{R}^* := \left(\bigcup_{i \in I} R_i\right)^*$. *Given* $f_w(q) = (n, m, X)$ *we also write* $f_w^0(q)$, $f_w^1(q)$ *and* $f_w^2(q)$ *for* n, m *and* X *respectively.*

Definition 3. *An* assignment *is a partial function* $h : \mathbf{P} \to \mathbb{N}$. *It* agrees with *a world* w, *written* $w \multimap h$, *if* $\mathsf{dom}(h) = Q$ *and (i) for all* $q \in Q$: $f_w^0(q) \leq h(q) \leq f_w^1(q)$ *and* $h(q) \notin f_w^2(q)$, *(ii)* $(p, q) \in C_w^+$ *implies* $h(p) = h(q)$ *and (iii)* $(p, q) \in C_w^-$ *implies* $h(p) \neq h(q)$. *In this case we also write* h *for the natural continuation of* h *on the set set of all expressions built out of elements of* Q *and* $\mathbb{N}$.

Definition 4. *We say that an expression E is* determined *at world w in $\mathcal{M}$ iff for all h and h' that agree with w we have $h(E) = h'(E)$. If this is the case we also write $[\![E]\!]^{\mathcal{M},w}$ for the value of E at w in $\mathcal{M}$.*

Definition 5. *Let $\mathcal{M} = (W, \mathcal{R}, V)$ be a crypto model, $w \in W$ and h an assignment that agrees with w. We define the satisfaction relation $\mathcal{M}, w, h \models \phi$ saying that ϕ is true at w with regard to h. Models with superscripts are from Definition 8 which runs in parallel to this one.*

$\mathcal{M}, w, h \models \top$	*always*
$\mathcal{M}, w, h \models p$	*iff* $p \in P_w$
$\mathcal{M}, w, h \models L_i$	*iff* $i \in L_w$
$\mathcal{M}, w, h \models p = E$	*iff* $h(p) = h(E)$
$\mathcal{M}, w, h \models \neg\phi$	*iff not* $\mathcal{M}, w, h \models \phi$
$\mathcal{M}, w, h \models \phi_1 \wedge \phi_2$	*iff* $\mathcal{M}, w, h \models \phi_1$ *and* $\mathcal{M}, w, h \models \phi_2$
$\mathcal{M}, w, h \models K_i\phi$	*iff* wR_iw' *and* $h' \multimap w'$ *imply* $\mathcal{M}, w', h' \models \phi$
$\mathcal{M}, w, h \models G\phi$	*iff* $w\mathcal{R}^*w'$ *and* $h' \multimap w'$ *imply* $\mathcal{M}, w', h' \models \phi$
$\mathcal{M}, w, h \models \langle \mathsf{Open}_i \rangle \phi$	*iff* $\mathcal{M}^{\mathsf{Open}_i}, (w, \alpha), h \models \phi$ *where α is from Def. 8.*
$\mathcal{M}, w, h \models \langle \mathsf{Close}_i \rangle \phi$	*iff* $\mathcal{M}^{\mathsf{Close}_i}, (w, \alpha), h \models \phi$ *where α is from Def. 8.*
$\mathcal{M}, w, h \models \langle !\ p \rangle \phi$	*iff* $\mathcal{M}, w, h \models p$ *and* $\mathcal{M}^{!p}, (w, \alpha), h \models \phi$
$\mathcal{M}, w, h \models \langle !\ p = E \rangle \phi$	*iff* $\mathcal{M}, w, h \models p = E$ *and* $\mathcal{M}^{!p=E}, (w, \alpha), h \models \phi$
$\mathcal{M}, w, h \models \langle !\ p \neq E \rangle \phi$	*iff* $\mathcal{M}, w, h \models p \neq E$ *and* $\mathcal{M}^{!p\neq E}, (w, \alpha), h \models \phi$
$\mathcal{M}, w, h \models \langle p \leftarrow^i N \rangle \phi$	*iff* $\mathcal{M}, w, h \models G\neg p$ *and* $\mathcal{M}^{p \leftarrow^i N}, (w, \alpha), h' \models \phi$ *where* $h' := h \cup \{(p, N)\}$
$\mathcal{M}, w, h \models \langle ?\psi \rangle \phi$	*iff* $\mathcal{M}, w, h \models \psi \wedge \phi$
$\mathcal{M}, w, h \models \langle A_1; A_2 \rangle \phi$	*iff* $\mathcal{M}, w, h \models \langle A_1 \rangle \langle A_2 \rangle \phi$
$\mathcal{M}, w, h \models \mathsf{Prime}\ E$	*iff $h(E)$ is a prime number*
$\mathcal{M}, w, h \models \mathsf{Coprime}\ E_1E_2$	*iff $h(E_1)$ and $h(E_2)$ are coprime*

To define the models $\mathcal{M}^{\mathsf{Open}_i}$, $\mathcal{M}^{\mathsf{Close}_i}$, $\mathcal{M}^{!p=E}$ and $\mathcal{M}^{!p\neq E}$ we use action models with factual change. The following is an adaption of [2], the main difference being that we model factual change with functions operating directly on the valuation instead of substitutions. As shown in [6, Corollary 17] this is equally expressive in general. Also note that the action models in Definition 7 below are given with respect to a certain Kripke model which they will be applied to. While this is not standard in the literature, the local listener sets make it both necessary and natural: The meaning of announcements and an operator Open_i simply depends on who is listening.

Definition 6. *An* action model *is a tuple $(A, \mathcal{R})$ where A is a set of so-called* action tokens*, $\mathcal{R} = (R_i)_{i\in I}$ is a family of equivalence relations on A and furthermore for any $\alpha \in A$ we have a formula $\mathsf{pre}(\alpha)$, called the* precondition *of α, and a function change_α which maps valuations to valuations.*

An action *is a triple $(A, \mathcal{R}, \alpha)$ where $(A, \mathcal{R})$ is an action model and α is an element $\alpha \in A$. We also write just α to refer to $(A, \mathcal{R}, \alpha)$.*

Definition 7. *The* product update *given by an action* $(A, \mathcal{R}, \alpha)$ *is a function on pointed models defined as follows.*

$$\begin{aligned}(W, \mathcal{R}, V), w \mapsto (W', \mathcal{R}', V'), (w, \alpha) \text{ where}\\ W' &:= \{(w, \alpha) \in W \times A \mid w \vDash \mathsf{pre}(\alpha)\}\\ (w, \alpha)R_i'(v, \beta) &: \iff wR_iv \text{ and } \alpha R_i \beta\\ V'(w, \alpha) &:= (\mathsf{change}_\alpha \circ V)(w)\end{aligned}$$

We write $\mathcal{M}^\alpha$ *for the result of updating* $\mathcal{M}$ *with the action* α*.*

Definition 8. *For any pointed model* $\mathcal{M}, w$ *satisfying the appropriate preconditions we define the models* $\mathcal{M}^{\mathsf{Open}_i}$*,* $\mathcal{M}^{\mathsf{Close}_i}$*,* $\mathcal{M}^{!p=E}$ *and* $\mathcal{M}^{!p\neq E}$ *as the result of the product update of* $\mathcal{M}$ *with the following actions. See below for an informal explanation what these action models do.*

(i) If E *is determined at* w *the action* $p \leftarrow^i E$ *is given by* $(\{\alpha, \beta\}, \mathcal{R}, \alpha)$ *where*

$$\begin{aligned}N &:= [\![E]\!]^{\mathcal{M}, w}\\ \mathsf{pre}(\alpha) := \mathsf{pre}(\beta) &:= G\neg p\\ \mathsf{change}_\alpha(P, L, f, C^+, C^-) &:= (P \cup \{p\}, L, f \cup \{(p, (N, N, \varnothing))\}, C^+, C^-)\\ \mathsf{change}_\beta(P, L, f, C^+, C^-) &:= (P, L, f \cup \{(p, (0, \mathsf{regsize}, \{N\}))\}, C^+, C^-)\\ \alpha R_i \beta &\text{ iff } i \neq a\end{aligned}$$

Here regsize *is a globally fixed maximum which registers can store, say* 10 *in the guessing game or* 2^n *to model* n*-bit registers in electronic systems.*

(ii) Open_i *is given by* $(\{\alpha, \beta\}, \mathcal{R}, \alpha)$ *where*

$$\begin{aligned}\mathsf{pre}(\alpha) &:= \top \quad \mathsf{change}_\alpha(P, L, f, C^+, C^-) := (P, L \cup \{i\}, f, C^+, C^-)\\ \mathsf{pre}(\beta) &:= \top \quad \mathsf{change}_\beta := \mathsf{id}\\ &\alpha R_j \beta \text{ iff } j \notin L_w \cup \{i\}\end{aligned}$$

(iii) Close_i *is given by the same action structure as* Open_i *but with the different function* $\mathsf{change}_\alpha(P, L, f, C^+, C^-) := (P, L \setminus \{i\}, f, C^+, C^-)$*.*

(iv) The command $!p = N$ *is given by* $(\{\alpha, \beta\}, \mathcal{R}, \alpha)$ *where*

$$\begin{aligned}\mathsf{pre}(\alpha) &:= (p = N) \quad \mathsf{change}_\alpha := \mathsf{id}\\ \mathsf{pre}(\beta) &:= \neg(p = N) \quad \mathsf{change}_\beta := \mathsf{id}\\ &\alpha R_i \beta \text{ iff } i \notin L_w\end{aligned}$$

(v) The command $!p = q$ *is given by* $(\{\alpha, \beta, \gamma\}, \mathcal{R}, \alpha)$ *where*

$$\begin{aligned}\mathsf{pre}(\alpha) &:= p \wedge q \wedge (p = q) \quad \mathsf{change}_\alpha := \mathsf{id}\\ \mathsf{pre}(\beta) &:= \neg p \wedge \neg q \quad \mathsf{change}_\beta(P, f, C^+, C^-) := (P, f, C^+ \cup \{(p, q)\}, C^-)\\ \mathsf{pre}(\gamma) &:= \top \quad \mathsf{change}_\gamma := \mathsf{id}\\ &\alpha R_i \beta \text{ for all } i \quad \alpha R_i \gamma \text{ and } \beta R_i \gamma \text{ iff } i \notin L_w\end{aligned}$$

(vi) The command $!p \neq N$ *is given by* $(\{\alpha, \beta, \gamma\}, \mathcal{R}, \alpha)$ *where*

$$\begin{aligned}\mathsf{pre}(\alpha) &:= p \wedge (p \neq N) \quad \mathsf{change}_\alpha := \mathsf{id}\\ \mathsf{pre}(\beta) &:= \neg p \quad \mathsf{change}_\beta(P, f, C^+, C^-) := (P, f', C^+, C^-)\\ \mathsf{pre}(\gamma) &:= \top \quad \mathsf{change}_\gamma := \mathsf{id}\\ &\alpha R_i \beta \text{ for all } i \quad \alpha R_i \gamma \text{ and } \beta R_i \gamma \text{ iff } i \notin L_w\end{aligned}$$

$$f'(q) := \begin{cases} (f^0(p), f^1(p), f^2(p) \cup \{N\}) & \text{if } q = p \\ f(q) & \text{otherwise} \end{cases}$$

(vii) The command $!p \neq q$ *is given by* $(\{\alpha, \beta, \gamma\}, \mathcal{R}, \alpha)$ *where*

$\mathsf{pre}(\alpha) := p \wedge q \wedge (p \neq q)$	$\mathsf{change}_\alpha := \mathsf{id}$
$\mathsf{pre}(\beta) := \neg(p \wedge q)$	$\mathsf{change}_\beta(P, f, C^+, C^-) := (P, f, C^+, C^- \cup \{(p, q)\})$
$\mathsf{pre}(\gamma) := \top$	$\mathsf{change}_\gamma := \mathsf{id}$
$\alpha R_i \beta$ *for all* i	$\alpha R_i \gamma$ *and* $\beta R_i \gamma$ *iff* $i \notin L_w$

Definition 8 captures the intended meanings from Section 3.1 as follows.

Register creation $p \leftarrow^i E$ makes two copies for all worlds in the previous model. It then sets p to the value of E in the actual world and allows p to have any other value than E in the other copies. The copies can only be distinguished for i, the agent for whom the register was created. Note that the value of E and not E itself is put in the register and the expression only has to be determined at the current world. This means that i does not need to have all the necessary information to evaluate the expression.

The commands Open and Close modify the listener set. Because a call for attention should also be heard by other listeners, these modifications are also done on copies of the worlds. The event is noticed by the agent who is addressed and the current listeners. Hence the order of Open commands matters: After executing $\mathsf{Open}_{\mathrm{Alice}}$; $\mathsf{Open}_{\mathrm{Bob}}$, Alice will know that Bob is listening but not necessarily vice versa. This also implies that $\mathsf{Open}_{\mathrm{Alice}}$; $\mathsf{Open}_{\mathrm{Bob}}$ is different from an alternative operator one might consider like $\mathsf{Open}_{\{\mathrm{Alice,Bob}\}}$ which would call for the attention of a group.

The four announcements behave similar to those in public announcement logic, e.g. they have to be truthful, but we have two modifications: First, announcements are received only by the current listeners and therefore also create copies of all worlds unless everyone is listening. Only in one copy the set of worlds are restricted to those where the announcement is true. However, as a consequence of using $\mathsf{S5}$, our announcements are not fully secret: Also non-listening agents will consider it possible that other agents learned something after an announcement. Second, the last three action models do not remove whole worlds but instead change their valuation, effectively removing agreeing assignments. This reflects that our register models are encodings of larger Kripke models. Adding a constraint to the valuation of a world in a register model is the same as deleting the corresponding worlds in a larger Kripke model.

We conclude this section with a definition of world-level truth and validity.

Definition 9. *For any* ϕ *we define:*

$$\mathcal{M}, w \models \phi \;\; \textit{iff} \;\; \forall h \textit{ with } w \multimap h : \mathcal{M}, w, h \models \phi$$

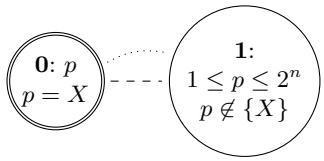
A formula ϕ *is* valid *iff for all* $\mathcal{M}, w$ *we have that* $\mathcal{M}, w \vDash \phi$*. We then write* $\vDash \phi$*.*

A sound and complete axiomatization of the resulting logic will be future work. Given that our commands are special cases of action models with factual change, we think that this will be feasible using the general ideas of [1,2,8] and standard axioms for modular arithmetic.

4 Monte Carlo Model Checking

So far, the register models are merely an encoding of larger Kripke models: Instead of many possible worlds we will still have go through lots of assignments to check a formula. But register models also motivate the following Monte Carlo method.

If we pick assignments uniformly at random, we see that situations where registers do not contain the correct information are vastly more probable than the situation where the registers are filled with the correct values. Figure 5 shows the number of agreeing assignments for each world, given a register size of n bits.



$1 \qquad\qquad 2^n - 1$

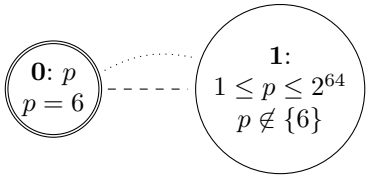
Fig. 5: Two worlds and the number of agreeing assignments.

This observation can be used to simplify model checking on our register models. In a world where p is false, it can be assigned almost any value by the agreeing assignments of that world. Thus, equality statements like $p = X$ will be false for *almost all* assignments. Hence equality and inequality statements can be checked by means of an approximate Monte Carlo model checking algorithm. The Monte Carlo method is based on a simple assumption:

$$\text{probably } \mathcal{M}, w \models \phi \text{ iff for "enough" } h \text{ with } w \multimap h : \mathcal{M}, w, h \models \phi.$$

To specify "enough" we choose some $k \in \mathbb{N}$. Then to evaluate a formula ϕ we first randomly generate a list of k agreeing assignments $h_1, \ldots, h_k$ for w. Then we check $\mathcal{M}, w, h_i \models \phi$ for each h_i in the list. We then say that ϕ is true/false iff it is true/false for all $i \leq k$ while in any mixed case it is undefined. Figure 6 shows an equality and an inequality statement and their estimated truth values.

The probability of disagreement between this Monte Carlo method and the "real" truth value which would be obtained by checking all assignments depends on the shape of the formula. However, for many formulas including simple equality statements and knowledge thereof it can be made arbitrarily small by using a larger k. Hence the choice of k provides a trade-off between computation time and reliability of the model checking results.



$0 \not\models p = 7$ $\quad p = 7$ is *almost* false in 1.
$0 \models p \neq 7$ $\quad p \neq 7$ is *almost* true in 1.

Fig. 6: Estimating the truth.

The probabilistic view on register models also suggest two extensions of our work: First, we could interpret a language on them that combines probability theory with epistemic operators, similar to [11,13,14]. Second, we could define a probabilistic satisfaction relation, making "almost false" explicit.

5 Diffie-Hellman in Dynamic Epistemic Logic

Whitfield Diffie and Martin Hellman revolutionized the field of cryptography with their proposal for public key encodings [5]. Here is their famous protocol for key exchange over an insecure channel which is widely used on the Internet:

1. Alice and Bob agree on a large prime p and a base $g < p$ such that g and $p - 1$ are co-prime.
2. Alice picks a secret a and sends $g^a \bmod p = A$ to Bob.
3. Bob picks a secret b and sends $g^b \bmod p = B$ to Alice.
4. Alice calculates $k_a = B^a \bmod p$.
5. Bob calculates $k_b = A^b \bmod p$.
6. They now have a shared key $k_a = k_b$.
   This is because $(g^a)^b = (g^b)^a \bmod p$.

This protocol is known to be secure against passive eavesdropping in the followsing sense: Note that a, b, k_a and k_b are never sent over the channel. Furthermore, we assume that computing discrete logarithms is hard, i.e. anyone who sees p, g, g^a and g^b will not be able to efficiently compute a, b or $(g^a)^b$. Hence the established key $k_a = k_b$ can be used by Alice and Bob to encrypt and decrypt messages before and after they are sent, respectively.

The language given in Definition 1 allows us to formulate the complete Diffie-Hellman key exchange as a protocol to be executed on register models:

1. $?\ \mathsf{Prime}\ p\ ;\ ?\ g \in [1..p]\ ;\ ?\ \mathsf{Coprime}\ g\ (p-1)\ ;$
2. $a \overset{\text{Alice}}{\leftarrow} N\ ;\ A \overset{\text{Alice}}{\leftarrow} g^a \bmod p\ ;$
   $\mathsf{Open}_{\text{Bob}}\ ;\ !A\ ;\ \mathsf{Close}_{\text{Bob}}\ ;$
3. $b \overset{\text{Bob}}{\leftarrow} M\ ;\ B \overset{\text{Bob}}{\leftarrow} g^b \bmod p\ ;$
   $\mathsf{Open}_{\text{Alice}}\ ;\ !B\ ;\ \mathsf{Close}_{\text{Alice}}\ ;$
4. $k_a \overset{\text{Alice}}{\leftarrow} B^a \bmod p\ ;$
5. $k_b \overset{\text{Bob}}{\leftarrow} A^b \bmod p\ ;$
6. $?k_a = k_b$

Moreover, the goal of the key exchange can be stated as a register language formula: The values of s_1 and s_2 should be equal, Alice and Bob should know them but a passive eavesdropper (who is traditionally called Eve) should not:

$$(k_a = k_b) \wedge (K_{\text{Alice}} k_a \wedge K_{\text{Bob}} k_b) \wedge (\neg K_{\text{Eve}} k_a \wedge \neg K_{\text{Eve}} k_b)$$

6 Results and Future Work

Register models make it feasible to implement knowledge of and communication about large numbers with Kripke semantics. Our models are more succinct than ordinary Kripke models which would need $(2^k)^m$ many worlds instead of 2^k in our setting to represent k registers with values ranging from 0 to 2^m. While general-purpose methods of abstraction might achieve similar compression rates, our framework has the conceptual advantage that it explicitly models which possible situations do not have to be distinguished. The big models never have to be generated in the first place.

We implemented a standard and a Monte Carlo model checking algorithm and compared their performance. Table 1 shows how many seconds it takes both algorithms to check one execution of the Diffie Hellman protocol as formalized in the previous section. As initial model we used a one-world model where it is common knowledge that Eve is listening but no other facts are known.

Increasing the register size substantially slows down the standard algorithm as expected but only has a slight effect on the Monte Carlo method which only checks a few randomly picked assignments at each world. The small non-monotone increase is probably due to the generation of random numbers within a bigger interval.

Register size	Standard	Monte Carlo
2^8	1.07	2.74
2^9	1.36	2.82
2^{10}	2.13	3.41
2^{11}	3.59	3.24
2^{12}	5.17	2.80
2^{13}	11.56	3.28
2^{14}	22.66	3.57
2^{15}	44.44	4.10
2^{16}	81.26	3.52

Table 1: Runtime of both algorithms in seconds.

The presented ideas should serve as a starting point for more research. As a conclusion we present the following program for Epistemic Crypto Logic.

1. Find a sound and complete axiomatization for the extended language for cryptographic protocols, building on the axiomatization for guessing games given in [10].
2. Use a set of reduction axioms to implement a formula-rewriting scheme. The results could then be fed into existing model checkers for S5.
3. Optimize the model checkers for register languages. A promising approach for model checking DEL is to translate it to quantified boolean formulas which are equivalent on so-called knowledge structures as presented in [19].

4. Explore the computational complexity of the presented languages and the model checking problem.

Our work also raises the question what exactly it means to verify a cryptographic protocol. A common answer to this is to show that certain attacks on the protocol are impossible. In our framework attacks on cryptographic protocols can be viewed as planning problems, so it provides a connection between cryptography, epistemic model checking and epistemic planning.

Concluding, we would like to stress that this work can be useful not just for reasoning about communication and cryptography but also provides lessons for epistemic model checking and epistemic planning itself. There is no need to confine ourselves to muddy children and similar puzzles. Debates about DEL and its variants might profit from real life cryptographic examples, e.g. when comparing different notions of knowledge.

Acknowledgments

Thanks to Alexandru Baltag, Johan van Benthem, Joshua Sack and Kaile Su who all provided valuable feedback to this work at different stages. We also thank the anonymous LAMAS referees for their useful comments and suggestions.

References

1. A. Baltag, L. Moss, and S. Solecki. The logic of public announcements, common knowledge, and private suspicions. In I. Bilboa, editor, *Proceedings of TARK'98*, pages 43–56, 1998.
2. J. v. Benthem, J. v. Eijck, and B. Kooi. Logics of communication and change. *Information and computation*, 204(11):1620–1662, 2006.
3. I. Boureanu, M. Cohen, and A. Lomuscio. Automatic verification of temporal-epistemic properties of cryptographic protocols. *Journal of Applied Non-Classical Logics*, 19(4):463–487, 2009.
4. H. Delfs and H. Knebel. *Introduction to Cryptography: Principles and applications.* Springer, 2002.
5. W. Diffie and M. Hellman. New directions in cryptography. *Information Theory, IEEE Transactions on*, 22(6):644–654, 1976.
6. H. P. v. Ditmarsch and B. P. Kooi. Semantic results for ontic and epistemic change. In G. Bonanno, W. v. d. Hoek, and M. Wooldridge, editors, *Logic and the Foundations of Game and Decision Theory (LOFT 7)*, volume 3 of *Texts in Logic and Games*, pages 88–118. Amsterdam University Press, 2006.
7. H. v. Ditmarsch, J. v. Eijck, I. Hernández-Antón, F. Sietsma, S. Simon, and F. Soler-Toscano. Modelling Cryptographic Keys in Dynamic Epistemic Logic with DEMO. In J. B. Pérez, M. A. Sánchez, P. Mathieu, J. M. C. Rodríguez, E. Adam, A. Ortega, M. N. Moreno, E. Navarro, B. Hirsch, H. Lopes-Cardoso, and V. Julián, editors, *Highlights on Practical Applications of Agents and Multi-Agent Systems*, volume 156 of *Advances in Intelligent and Soft Computing*, pages 155–162. Springer Berlin Heidelberg, 2012.

8. H. v. Ditmarsch, A. Herzig, E. Lorini, and F. Schwarzentruber. Listen to Me! Public Announcements to Agents That Pay Attention — or Not. In D. Grossi, O. Roy, and H. Huang, editors, *Logic, Rationality, and Interaction*, volume 8196 of *Lecture Notes in Computer Science*, pages 96–109. Springer Berlin Heidelberg, 2013.
9. J. v. Eijck. Elements of Epistemic Crypto Logic. Slides from a talk at the LogiCIC Workshop, Amsterdam, Dec. 2013. Available online at `http://homepages.cwi.nl/~jve/papers/13/pdfs/eeclTALK.pdf`.
10. J. v. Eijck and M. Gattinger. Elements of Epistemic Crypto Logic (Extended Abstract). In *Proceedings of the 14th International Conference on Autonomous Agents and Multiagent Systems (AAMAS 2015)*, 2015.
11. J. v. Eijck and F. Schwarzentruber. Epistemic Probability Logic Simplified. In R. Goré, B. Kooi, and A. Kurucz, editors, *Advances in Modal Logic*, volume 10, pages 158–177, 2014.
12. M. Gattinger. Dynamic Epistemic Logic for Guessing Games and Cryptographic Protocols. Master's thesis, University of Amsterdam, 2014.
13. J. Halpern. *Reasoning About Uncertainty*. MIT Press, Cambridge, MA, USA, 2003.
14. B. P. Kooi. *Knowledge, Chance, and Change*. PhD thesis, Groningen University, 2003.
15. G. L. Miller. Riemann's hypothesis and tests for primality. *Journal of computer and system sciences*, 13(3):300–317, 1976.
16. C. Paar and J. Pelzl. *Understanding cryptography: a textbook for students and practitioners*. Springer, 2010.
17. M. O. Rabin. Probabilistic algorithm for testing primality. *Journal of number theory*, 12(1):128–138, 1980.
18. P. Ryan, S. Schneider, M. Goldschmidt, G. Lowe, and B. Roscoe. *Modelling and Analysis of Security Protocols*. Addison Wesley, 2001.
19. K. Su, A. Sattar, and X. Luo. Model Checking Temporal Logics of Knowledge Via OBDDs. *The Computer Journal*, 50(4):403–420, 2007.
20. P. F. Syverson and P. C. Van Oorschot. A unified cryptographic protocol logic. Technical report, DTIC Document, 1996.