

# Cryptographic Protocol Analysis with Kripke Models

Jan van Eijck  
CWI & ILLC, Amsterdam

(joint work with Malvin Gattinger)

PEM talk, 24 January, 2014

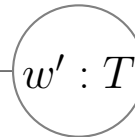
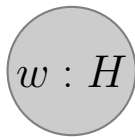
## Abstract

The talk is about the application of model checking to the analysis of cryptographic security protocols. As a step towards that, I will start with a proposal for how to represent knowledge of large numbers in Kripke models, and I will use this representation to model a protocol for secret key distribution over an insecure network in DEL.

An prototype implementation of an epistemic model checker for cryptographic protocol analysis exists.



## How to Model Ignorance?



**What does it mean to know a number?**



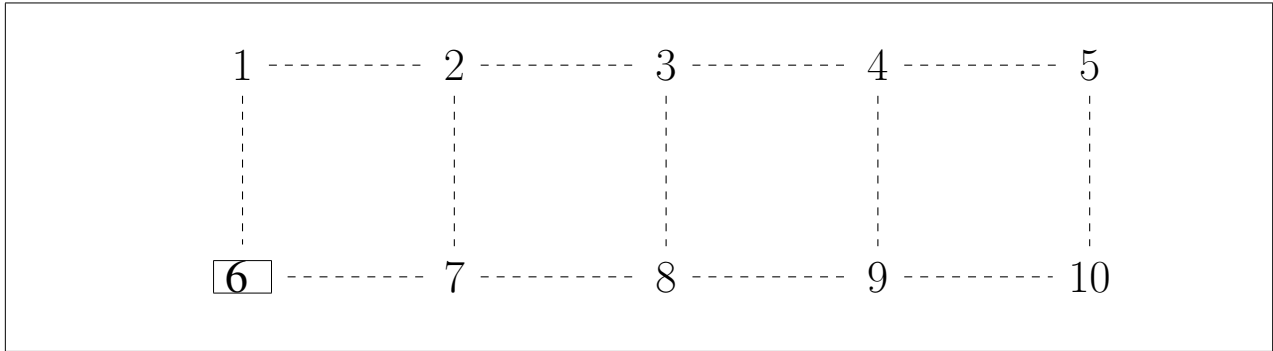
# Jan, Gaia and Rosa Play the Number Guessing Game



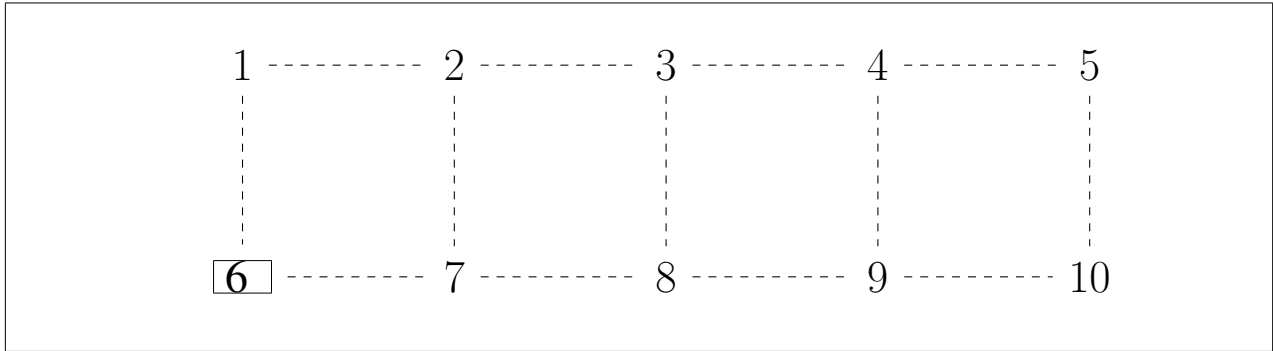
## Number Guessing Game, Naive Version

- 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 gets the toy. It is Gaia’s turn to start, for last time we played this game Rosa started the guessing.”
- Gaia and Rosa agree . . .
- After a number of rounds, Jan announces: “Rosa, you have won.”

## Number Guessing: Naive Representation



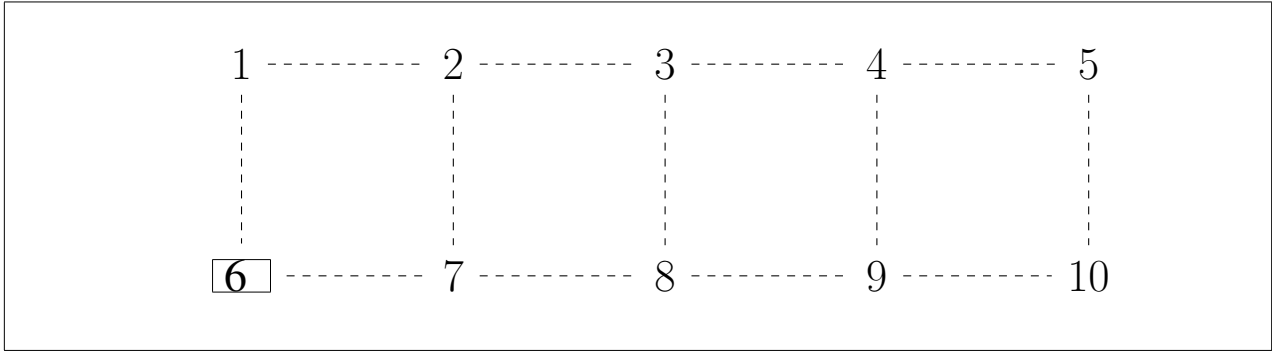
## Number Guessing: Naive Representation



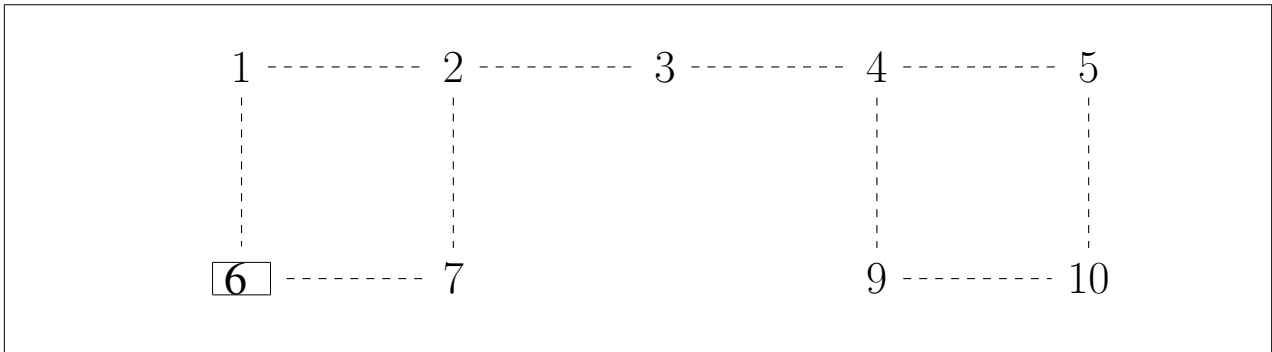
“eight” ... “no”

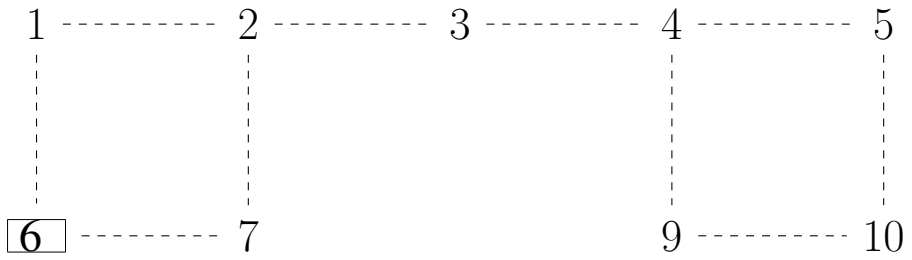


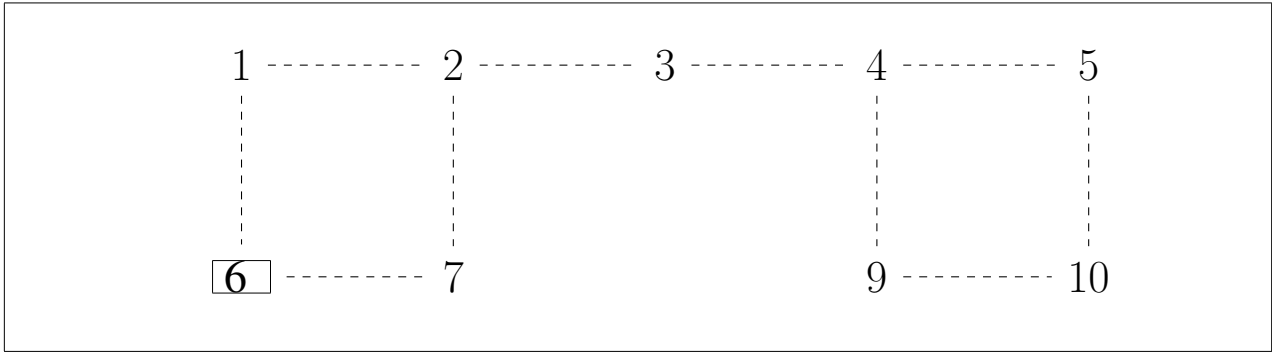
## Number Guessing: Naive Representation



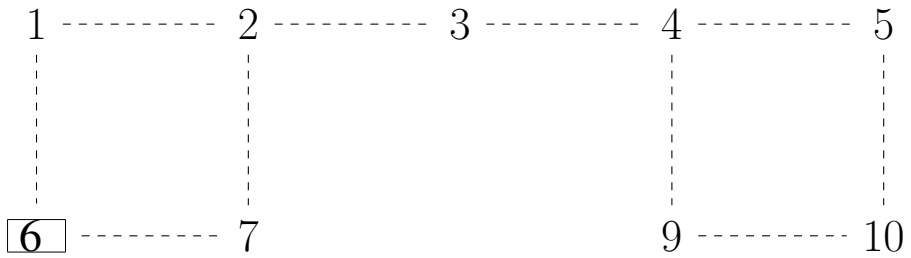
“eight” ... “no”



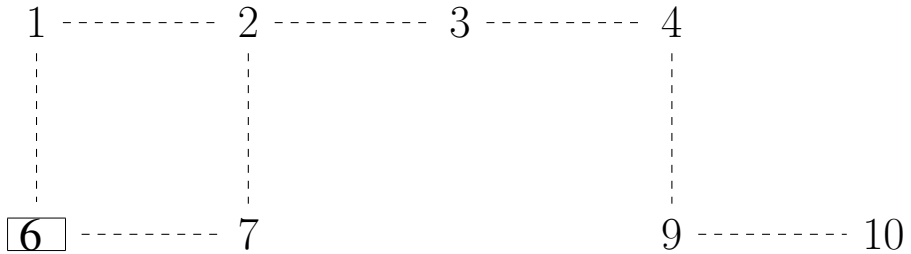




“five” ... “no”



“five” ... “no”



## Number Guessing Game, More Sophisticated Version

- 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 gets the toy. It is Gaia’s turn to start, for last time we played this game Rosa started the guessing.”
- Gaia does not agree. “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.”
- Jan writes a number on a piece of paper, hidden from Gaia and Rosa.
- After a number of rounds, Jan announces: “Rosa, you have won.”
- Jan shows the piece of paper as a proof.

## Jan, Gaia and Rosa Play the Number Guessing Game (again)



## Jan, Gaia and Rosa Play the Number Guessing Game (again)



register

## Registers, Fixing a Number

At the point where either Gaia or Rosa demands that the secret number gets **written down**, and that Jan shows it as a proof that the procedure was honest, the important notion of a **register** arises.

The register allows Jan to prove that he knew (had fixed) the number beforehand.

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.

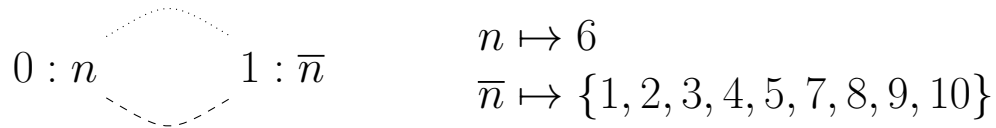
It is enough to distinguish these two possibilities.



## Jan knows a number

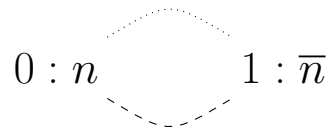
Jan knows  $n$ , the two children do not know  $n$ . We use  $n$  for the register,  $\dot{n}$  for the value of that register (its contents).

This leads to the following register model.



0 is a world that has register  $n$ , with number  $\dot{n} = 6$  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  $\dot{n}$ .

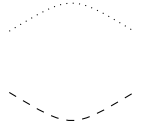
Gaia (dots) and Rosa (dashes) do not have access to the register.



$$n \mapsto 6$$

$$\bar{n} \mapsto \{1, 2, 3, 4, 5, 7, 8, 9, 10\}$$

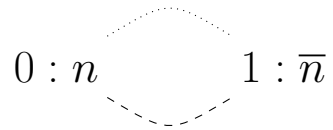
$0 : n$        $1 : \bar{n}$



$n \mapsto 6$

$\bar{n} \mapsto \{1, 2, 3, 4, 5, 7, 8, 9, 10\}$

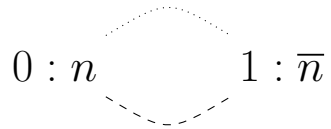
“ten” ... “no”



$$n \mapsto 6$$

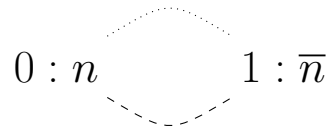
$$\bar{n} \mapsto \{1, 2, 3, 4, 5, 7, 8, 9, 10\}$$

“ten” ... “no”



$$n \mapsto 6$$

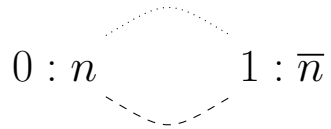
$$\bar{n} \mapsto \{1, 2, 3, 4, 5, 7, 8, 9\}$$



$$n \mapsto 6$$

$$\bar{n} \mapsto \{1, 2, 3, 4, 5, 7, 8, 9, 10\}$$

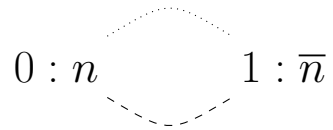
“ten” ... “no”



$$n \mapsto 6$$

$$\bar{n} \mapsto \{1, 2, 3, 4, 5, 7, 8, 9\}$$

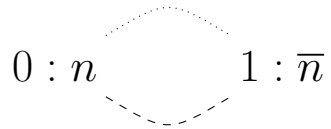
“six” ... “yes”



$$n \mapsto 6$$

$$\bar{n} \mapsto \{1, 2, 3, 4, 5, 7, 8, 9, 10\}$$

“ten” ... “no”



$$n \mapsto 6$$

$$\bar{n} \mapsto \{1, 2, 3, 4, 5, 7, 8, 9\}$$

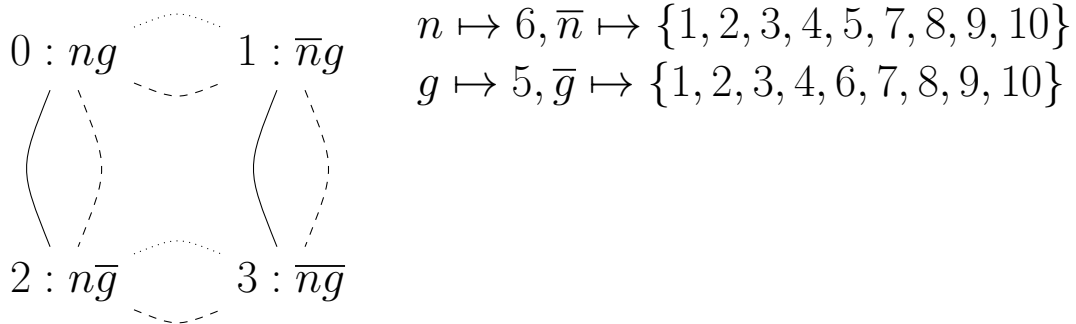
“six” ... “yes”

$$0 : n$$

$$n \mapsto 6$$

## Gaia Makes a Guess

Use  $g$  for the guess of Gaia. Gaia knows her guess, but has not yet revealed it to the others: solid lines for Jan, dotted lines for Gaia, dashed lines for Rosa.



Without registers, this blows up to a model with 100 worlds.

$k$  registers, each of  $m$  bits, blow up to  $(2^m)^k$  possibilities.

## Gaia Reveals Her Guess

Gaia reveals the contents of her register:  $g = 5$ .



## Gaia Reveals Her Guess

Gaia reveals the contents of her register:  $g = 5$ .

$$0 : ng \quad 1 : \bar{n}g \quad \begin{array}{l} n \mapsto 6, \bar{n} \mapsto \{1, 2, 3, 4, 5, 7, 8, 9, 10\} \\ g \mapsto 5 \end{array}$$

## Gaia Reveals Her Guess

Gaia reveals the contents of her register:  $g = 5$ .

$$\begin{array}{l} 0 : ng \quad \overset{\text{dotted}}{\curvearrowright} \quad 1 : \bar{n}g \\ \quad \quad \quad \underset{\text{dashed}}{\curvearrowleft} \quad \quad \quad \end{array} \quad \begin{array}{l} n \mapsto 6, \bar{n} \mapsto \{1, 2, 3, 4, 5, 7, 8, 9, 10\} \\ g \mapsto 5 \end{array}$$

Jan states that the guess is wrong:  $n \neq g$ .

## Gaia Reveals Her Guess

Gaia reveals the contents of her register:  $g = 5$ .

$$\begin{array}{ccc} 0 : ng & \begin{array}{c} \text{---} \\ \text{---} \end{array} & 1 : \bar{n}g \\ & \begin{array}{c} \text{---} \\ \text{---} \end{array} & \end{array} \quad \begin{array}{l} n \mapsto 6, \bar{n} \mapsto \{1, 2, 3, 4, 5, 7, 8, 9, 10\} \\ g \mapsto 5 \end{array}$$

Jan states that the guess is wrong:  $n \neq g$ .

$$\begin{array}{ccc} 0 : ng & \begin{array}{c} \text{---} \\ \text{---} \end{array} & 1 : \bar{n}g \\ & \begin{array}{c} \text{---} \\ \text{---} \end{array} & \end{array} \quad \begin{array}{l} n \mapsto 6, \bar{n} \mapsto \{1, 2, 3, 4, 7, 8, 9, 10\} \\ g \mapsto 5 \end{array}$$

## Registers are like Names for Numbers

- Creating a register and writing a number in it can be viewed as a **baptism**.
- Register equality statements are like name equality statements.
- Compare  $n = g$  with “Hesperus is Phosphorus”.

## Truth of Equality Statements in Register Models

- It seems reasonable to represent the announcement of Gaia's guess as the announcement of the equality statement  $g = \dot{g}$ .
- Since  $\bar{g}$  means that the register does contain a different value from  $\dot{g}$ , the statement  $g = \dot{g}$  is false in both  $\bar{g}$  worlds, so these drop out of the picture.
- Jan can respond to Gaia's guess with  $n = g$  (this is what "you guessed it" means: your guess equals my number).

## Register Language for Guessing Games

Let  $p$  range over  $\mathbf{P}$  and let  $N$  range over  $\mathbb{Z}$ . Let  $i$  range over the set of agents  $I$ .

$$\varphi ::= \top \mid p \mid L_i \mid p = E \mid \neg\varphi \mid \varphi \wedge \varphi \mid K_i\varphi \mid [!p = E]\varphi \mid [!p \neq E]\varphi$$

$$E ::= p \mid N$$

$L_i$  expresses that agent  $i$  is listening to announcements.

Let  $G \subseteq I$ .  $L_G$  expresses that  $G$  are the listeners:

$$L_G = \bigwedge_{i \in G} L_i \wedge \bigwedge_{i \notin G} \neg L_i.$$

## Update Operations

Revealing positive or negative information about  $p$ :

$$! p = E$$

“You guessed it”

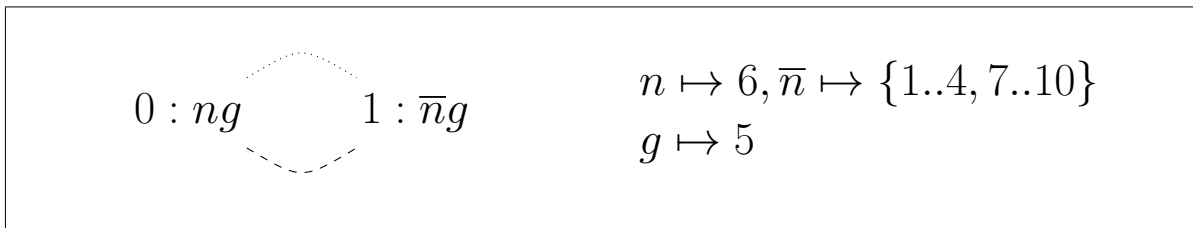
$$! p \neq E$$

“Your guess was wrong”

Interpretations of  $! p = E$  and  $! p \neq E$  are action models in the sense of [BMS98].

## Interpretation in Register Models

- A register assignment function is a function from registers to integers.
- A register assignment function  $h$  **agrees with** a possible  $w$  (notation  $w \dashv\circ h$ ) if  $q \in V_w$  iff  $h(q) = \dot{q}$ .
- Examples of (Dis-)Agreement:



The function  $h = \{n \mapsto 6, g \mapsto 5\}$  agrees with 0, but not with 1.

The function  $h' = \{n \mapsto 3, g \mapsto 5\}$  agrees with 1 but not with 0.



## Interpretation in Register Models: $\mathcal{M}, w, h \models \varphi$

Let  $h$  be a register assignment that agrees with  $w$ .

$\mathcal{M}, w, h \models \top$       always

$\mathcal{M}, w, h \models p$     iff  $p \in V(w)$

$\mathcal{M}, w, h \models p_1 = p_2$     iff  $h(p_1) = h(p_2)$

$\mathcal{M}, w, h \models p = N$     iff  $h(p) = N$

$\mathcal{M}, w, h \models \neg\varphi$     iff    not  $\mathcal{M}, w, h \models \varphi$

$\mathcal{M}, w, h \models \varphi_1 \wedge \varphi_2$     iff  $\mathcal{M}, w, h \models \varphi_1$  and  $\mathcal{M}, w, h \models \varphi_2$

$\mathcal{M}, w, h \models K_i\varphi$     iff  $(w, w') \in R_i$  and  $w' \dashv\circ h'$  imply  $\mathcal{M}, w', h' \models \varphi$

$\mathcal{M}, w, h \models [! p = E]\varphi$     iff  $\mathcal{M}, w, h \models p = E$  implies  $\mathcal{M}^{p=E}, w, h \models \varphi$

$\mathcal{M}, w, h \models [! p \neq E]\varphi$     iff  $\mathcal{M}, w, h \models p \neq E$  implies  $\mathcal{M}^{p \neq E}, w, h \models \varphi$

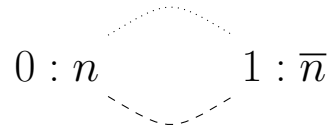
## Truth and Falsity at a World

From truth at a register assignment to truth and falsity at a world:

$$\mathcal{M}, w \models \varphi \text{ iff } \forall h \text{ with } w \multimap h : \mathcal{M}, w, h \models \varphi.$$

$$\mathcal{M}, w \models \neg \varphi \text{ iff } \forall h \text{ with } w \multimap h : \mathcal{M}, w, h \models \varphi.$$

## Truth Value Gaps



$$n \mapsto 6$$

$$\bar{n} \mapsto \{1..5, 7..10\}$$

$n = 7$  is false in 0.

$n = 7$  is neither true nor false in 1.

$n \neq 7$  is true in 0.

$n \neq 7$  is neither true nor false in 1.

## Now Let's Look at Probabilities

$$\begin{array}{ccccccc} 1 & & 1 & & & & 9 & & 4294967295 \\ & & & & \text{---} & & & & \\ & & 0 : n & & \text{---} & & 1 : \bar{n} & & \end{array}$$

Numbers of agreeing assignments for register size 10.

Numbers of agreeing assignments for register size of 32 bits.

## Now Let's Look at Probabilities

$$\begin{array}{ccccccc} 1 & & 1 & & & & 9 & & 4294967295 \\ & & & & \text{---} & & & & \\ & & 0 : n & \text{---} & & & 1 : \bar{n} & & \end{array}$$

Numbers of agreeing assignments for register size 10.

Numbers of agreeing assignments for register size of 32 bits.

With 64 bit size:  $2^{64} - 1 = 18446744073709551615$  agreeing assignments for world 1.

## Fair Game Interpretation of Probability (Christiaan Huygens)

- Suppose possible guesses run from 1 to 10.
- A fair guessing game would have 10 participants. Each guess corresponds with a ticket. Winning ticket gets the stake.
- Then a ticket is worth  $\frac{1}{10}$ th of the stake.
- World 0 corresponds to a single ticket, world 1 corresponds to a set of 9 tickets.
- With a register size of 64 bits, a ticket is worth  $1/18446744073709551616$  of the stake, and world 0 corresponds to a single ticket, world 1 corresponds to 18446744073709551615 tickets.
- This can be worked out in a logic that combines probability theory with epistemic operators [Hal03, Koo03, ES].

## Monte Carlo Checking of (In)Equality Statements

Suppose the register size is large enough, say 64 bits.

Then in a world where  $n$  is false, there are

$$2^{64} - 1 = 18446744073709551615$$

possibilities for the value of  $n$ .

Thus, in a world where  $n$  is false, equality statements  $n = X$  will be false for **almost all** values  $X$ .

Equality and inequality statements can be checked by means an approximate (Monte Carlo) model checking algorithm.

## Monte Carlo Style Model Checking

$\mathcal{M}, w \models \varphi$  iff for “enough”  $h$  with  $w \rightarrow h : \mathcal{M}, w, h \models \varphi$ .

Method:

- Randomly generate a list of  $n$  agreeing assignments  $h_1, \dots, h_n$  for  $w$ .
- Check  $\mathcal{M}, w, h_i \models \varphi$  for each  $h_i$  in the list.
- The probability that there is disagreement among the outcomes can be made arbitrarily small.



## Truth Value Gaps Almost Closed

$$\begin{array}{ccc} 1 & & \\ 0 : n \mapsto 6 & \begin{array}{c} \text{---} \\ \text{---} \end{array} & 1 : \bar{n} \mapsto \{M = -2^{63}..2^{63} - 1\} - \{6\} \\ & & 18446744073709551615 \end{array}$$

$n = 7$  is false in 0.

$n = 7$  is **almost** false in 1.

$n \neq 7$  is true in 0.

$n \neq 7$  is **almost** true in 1.

Article of faith of cryptanalysis: “Brute force attacks on number secrets are impossible.”

## Register Creation

Fix a large register size. Let  $M..K$  be the bounds of the register (say  $M = -2^{63}, K = 2^{63} - 1$ ).

Add to the language:

$$\varphi ::= [p \stackrel{i}{\leftarrow} N]\varphi$$

$p \stackrel{i}{\leftarrow} N$  is the command to link  $p$  to the integer number  $N$  with the link known only to agent  $i$ . It is assumed that  $N$  fits the register size.

Interpretation:  $\llbracket p \stackrel{i}{\leftarrow} N \rrbracket$  as the action model (in the sense of [BMS98]):

$$\boxed{e} : p := \top, \dot{p} := N \xrightarrow{I - \{i\}} e : p := \perp$$

## Update Example

Start with the blissful ignorance unit model:

0

## Update Example

Start with the blissful ignorance unit model:

0

Jan thinks of a number

$n \stackrel{j}{\leftarrow} 6$


## Update Example

Start with the blissful ignorance unit model:

0

Jan thinks of a number

$n \stackrel{j}{\leftarrow} 6$

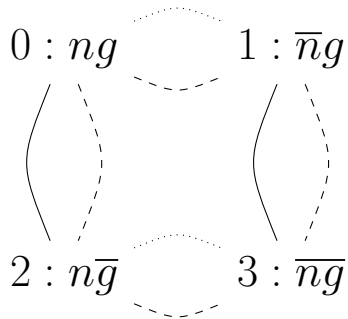
$0 : n$    $1 : \bar{n}$        $n \mapsto 6, \bar{n} \mapsto \{M..K\} - \{6\}$

Gaia thinks of a number:

$$g \xleftarrow{g} 5$$

Gaia thinks of a number:

$$g \xleftarrow{g} 5$$



$$n \mapsto 6, \bar{n} \mapsto \{M..K\} - \{6\}$$
$$g \mapsto 5, \bar{g} \mapsto \{M..K\} - \{5\}$$

## Communication: “ $a$ sends $\varphi$ to $b$ ”

Sequence of commands:

? $K_a\varphi$ ; **Open**  $b$ ; ! $\varphi$ ; **Close**  $b$

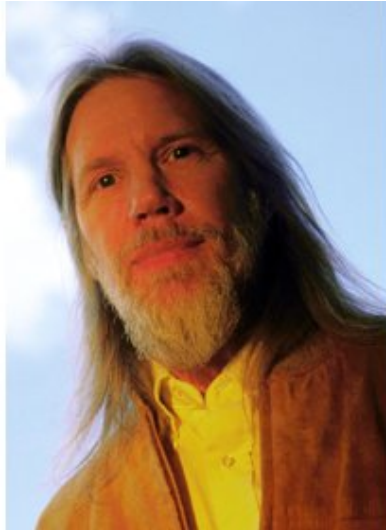
- ? $K_a\varphi$  tests whether  $K_a\varphi$  is true in the actual world.
- **Open**  $b$  adds  $b$  to the set of agents that are listening.
- ! $\varphi$  is the announcement of  $\varphi$ .
- **Close**  $b$  removes  $b$  from the set of agents that are listening.
- As far as  $b$  is concerned, the information  $\varphi$  could come from anywhere: **no authentication**.
- Anyone who is listening receives the info: the channel is **insecure**.



## Feasible Computation

- Fast algorithms for primality testing (e.g., the probabilistic Miller-Rabin test [[Mil76](#), [Rab80](#)]).
- Fast algorithms for co-primality testing (Euclid's GCD algorithm) and for finding modular inverses (Euclid's extended GCD algorithm, see above).
- Fast algorithms for addition and multiplication modulo, and for exponentiation modulo (see above, and compare [[DK02](#), [PP09](#)]).
- + Articles of faith: factorisation, discrete logarithm, . . . , are not feasible.
- Now refine the meaning of “knowing a number”: 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.

## Application: Diffie-Hellman Key Exchange [DH76]



Whitfield Diffie – Martin Hellman

## Diffie-Hellman Key Exchange Protocol

### Key Exchange Over Insecure Channel

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  $k = B^a \bmod p$ .
5. Bob calculates  $k = A^b \bmod p$ .
6. They now have a shared key  $k$ . This is because  $k = (g^a)^b = (g^b)^a \bmod p$ .

## Use of a Shared Secret Key for Secure Communication

Let  $p$  be the prime that Alice and Bob have agreed on, and let  $k$  be their shared key. Then message  $m$  is encoded as

$$m \times k \bmod p.$$

Such messages can be decoded by both Alice and Bob.

Alice knows  $p$ ,  $k$ ,  $g^b$  and  $a$ . She decodes cipher  $c$  with

$$c \times (g^b)^{(p-1)-a} \bmod p.$$

Bob knows  $p$ ,  $k$ ,  $g^a$  and  $b$ . He decodes cipher  $c$  with

$$c \times (g^a)^{(p-1)-b} \bmod p.$$

## Behind this: Fermat's Little Theorem



Pierre de Fermat (1601–1665)

If  $p$  is prime, then for every  $1 \leq a < p$ :  $a^{p-1} \equiv 1 \pmod{p}$ .



## Explanation of Alice's Decoding Recipe

We have:

$$\begin{aligned} (g^a)^{(p-1)-b} &= g^{a((p-1)-b)} = g^{a(p-1)} \times g^{-ab} = (g^{p-1})^a \times g^{-ab} \\ &\stackrel{Fermat}{=} 1^a \times g^{-ab} = g^{-ab} \pmod{p}, \end{aligned}$$

and therefore:

$$c \times (g^a)^{(p-1)-b} = (m \times g^{ab}) \times g^{-ab} = m \times (g^{ab} \times g^{-ab}) = m \pmod{p}.$$

## Diffie-Hellman Key Exchange as a Register Language Protocol

$a \stackrel{i}{\leftarrow} N ; A \stackrel{i}{\leftarrow} g^a \bmod p ;$

**Open**  $j ; !x = A ;$  **Close**  $j ;$

$b \stackrel{j}{\leftarrow} M ; B \stackrel{j}{\leftarrow} g^b \bmod p ;$

**Open**  $i ; !y = B ;$  **Close**  $i ;$

$k \stackrel{i}{\leftarrow} y^a \bmod p ;$

$k' \stackrel{j}{\leftarrow} x^b \bmod p ;$

? $k = k'$



## Program for Epistemic Crypto Logic

- Formulate complete logics. A complete calculus for register logic has axioms for propositional logic, S5 axioms for the  $K_i$  operators,  $K$  axioms for the  $A$  operators, Modus Ponens, necessitation for  $K_i$  and  $A$ , and reduction axioms for the command operators, in the style of [BvEK06].
- Build an efficient model checker for this language, using register models as defined in this talk. See [Gat13] for a first prototype.
- Use the model checker for modelling cryptographic protocols, plus attacks on them.
- Remark: Work out the connection with [WF13].

## Lessons for Epistemic Model Checking and Epistemic Planning

- No need to confine ourselves to muddy children and similar puzzles in model checking with DEL.
- What does it mean to verify a cryptographic protocol?
- One possible answer: to show that certain attacks on them are impossible.
- Attacks on cryptographic protocols can be viewed as planning problems ...

## References

- [BMS98] A. Baltag, L.S. 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.
- [BvEK06] J. van Benthem, J. van Eijck, and B. Kooi. Logics of communication and change. **Information and Computation**, 204(11):1620–1662, 2006.
- [DH76] W. Diffie and M. Hellman. New directions in cryptography. **IEEE Transactions on Information Theory**, 22(6):644–654, 1976.
- [DK02] Hans Delfs and Helmut Knebl. **Introduction To Cryptography — Principles and Applications**. Springer, 2002.

- [ES] Jan van Eijck and François Schwarzentruber. Epistemic probability logic simplified. Manuscript.
- [Gat13] Malvin Gattinger. Epistemic crypto logic — functional programming and model checking of cryptographic protocols. Technical report, ILLC, Amsterdam, 2013. Exam paper for the course ‘Functional Specification of Algorithms’.
- [Hal03] J. Halpern. **Reasoning About Uncertainty**. MIT Press, 2003.
- [Koo03] Barteld P. Kooi. **Knowledge, Chance, and Change**. PhD thesis, Groningen University, 2003.
- [Mil76] Gary L. Miller. Riemann’s hypothesis and tests for primality. **Journal of Computer and System Sciences**, 13(3):300–317, 1976.

- [PP09] Christof Paar and Jan Pelzl. **Understanding Cryptography — A Textbook for Students and Practitioners**. Springer, 2009.
- [Rab80] Michael O. Rabin. Probabilistic algorithm for testing primality. **Journal of Number Theory**, 12(1):128–138, 1980.
- [WF13] Yanjing Wang and Jie Fan. Knowing that, knowing what, and public communication: Public announcement logic with kv operators. In **IJCAI 2013**, pages 1139–1146, 2013.