Elements of Epistemic Crypto Logic

Jan van Eijck
CWI & ILLC, Amsterdam

LogiCIC Workshop, December 2, 2013

Abstract

The talk presents an extension of DEL (dynamic epistemic logic) intended for model checking of cryptographic protocols. Key elements are a feasible epistemic representation of knowledge of large integers, using register models, and exchange of such knowledge over a network. I will demonstrate how the approach can be used for model checking Diffie-Helman key exchange and similar protocols.

Overview

- Epistemic Representation of Number Guessing Games
- Register Models for Number Guessing Games
- Language for Number Guessing Games
- Interpretation in Register Models
- Truth Value Gaps and What to Do About Them
- Probabilities and Monte Carlo Style Model Checking
- Register Creation as an Update
- Communication: "a sends φ to b"
- Feasible Computation; Protocols; Secret Key Exchange
- Program for Epistemic Crypto Logic

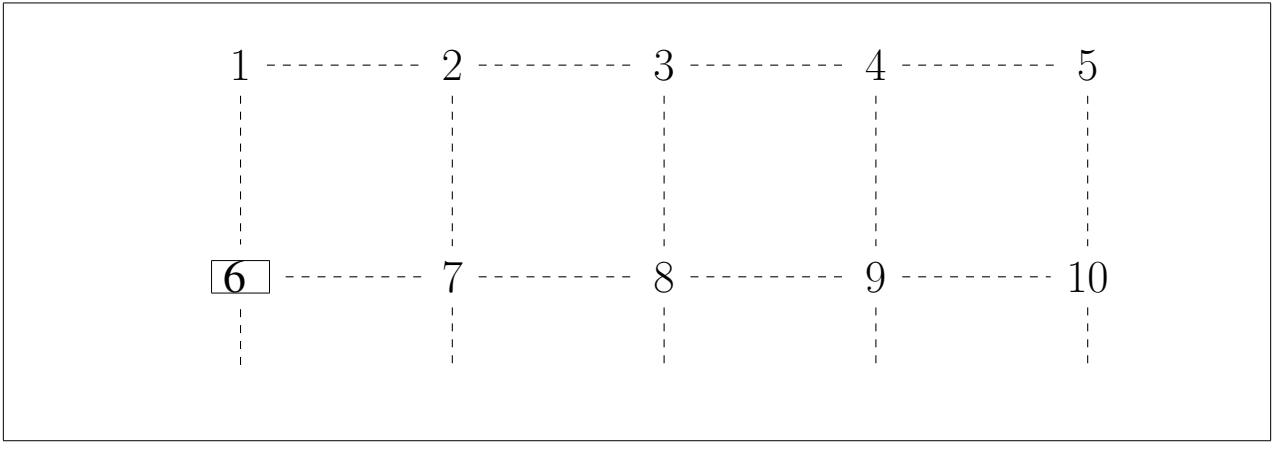
The Number Guessing Game, Naive Version

Scene: Gaia and Rosa are with their mother Heleen, and Gaia and Rosa quarrel about who can play with a toy.

Heleen proposes to play the number guessing game. She 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.”

They all agree, and after a number of rounds Heleen announces: “Rosa, you have won.”

Number Guessing: Naive Representation



The Number Guessing Game, Streetwise Version

Scene: Gaia and Rosa are with their mother Heleen, and Gaia and Rosa quarrel about who can play with a toy.

Heleen proposes to play the number guessing game. She says: "I have a number in mind, in the range from one to to ten. You may take turns guessing. Whoever guesses the number first gets the toy. It is Rosa's turn to start, for last time we played this game Gaia 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."

Now they all agree, and after a number of rounds Heleen announces: "Rosa, you have won." She shows the paper with the proof.

Knowing a Number

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

The register allows Heleen to prove that she knew (had fixed) the number beforehand.

So what is it that Heleen knew when we say that she knew the number?

What Does it Mean to Know a Number?

We are working in the context of epistemic logic.

> To know an integer number n is to be able distinguish a world where n is written in some register from all possible worlds where an integer number n' different from n is written in that register.

This can be made more precise, by clearly distinguishing between a register and its contents.

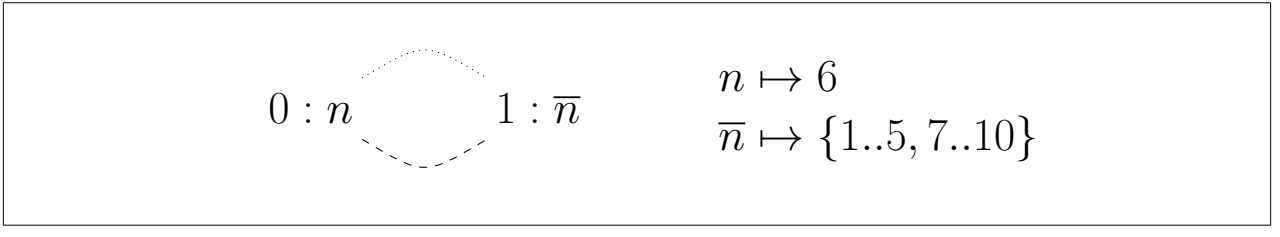
And the representation can be made more concise, by lumping together all situations where the register does not contain the number.

> To know an integer number n is to know the contents $\dot{n}$ of register n, not to know a number is not to have access to register n.

Heleen knows a number

Mother knows n, the two children do not know n. We use n for the register, $\dot{n}$ for the value of that register, and $\ddot{n}$ for the domain (set of possible values) of the register.

This leads to the following register model.

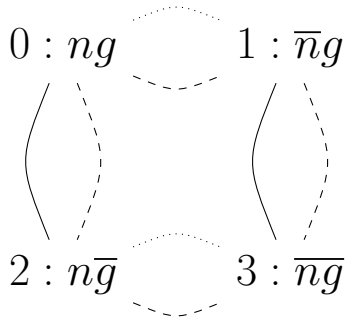


0 is a world that has register n, with number $\dot{n} = 6$ stored in it, while 1 differs from 0 in that it also has this register, but with all the possible values in it that differ from $\dot{n}$.

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

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 Heleen, dotted lines for Gaia, dashed lines for Rosa.



$n \mapsto 6, \overline{n} \mapsto \{1..5, 7..10\}$
$g \mapsto 5, \overline{g} \mapsto \{1..4, 6..10\}$

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

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

After Gaia has Revealed the Contents of her Register

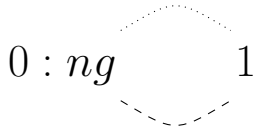
Gaia reveals the contents of her register: $g = 5$.

Gaia states her guess: $n = g$.

$n \mapsto 6, \overline{n} \mapsto \{1..5, 7..10\}$
$g \mapsto 5$

After Heleen's Reply

Since $\dot{n} \neq \dot{g}$ Heleen replies "you got it wrong", and we get:



$n \mapsto 6, \overline{n} \mapsto \{1..4, 7..10\}$
$g \mapsto 5$

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

Truth of (In)equality Statements in Register Models

- The difference between $n = g$ and $n = \dot{n}$ can perhaps be seen more clearly in the negative case.
- "You have it wrong" is paraprased as $n \neq g$, not as $n \neq \dot{n}$.
- In a world where g and n are both true, the equality $n = g$ can be checked by checking whether $\dot{g} = \dot{n}$.
- In a world where g is true and n false, the equality $n = g$ means that every value of n except for $\dot{n}$ equals g, which is certainly false (except in a domain with just two numbers, in case $\dot{n}$ and $\dot{g}$ are different).
- In a world where g and n are both false the equality $n = g$ means that every value for g except $\dot{g}$ equals every value of n except $\dot{n}$, which is certainly false.

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.

$$\begin{aligned} \varphi \ ::= \ & \top \mid p \mid L_i \mid p = E \mid \\ & \neg\varphi \mid \varphi \wedge \varphi \mid K_i\varphi \mid [A]\varphi \end{aligned}$$

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

A is some update operation (see below).

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].

Register Models

Let I be a finite agent set, let $\mathbf{P}$ be a denumerable set of propositions.

Definition 1 *A register model for I and $\mathbf{P}$ is a tuple*

$$\mathcal{M} = (W, R, V, L, \dot{V}, \ddot{V})$$

where

- (W, R, V) *is a multimodal S5 model for I and $\mathbf{P} \cup \{L_i \mid i \in I\}$,*
- $Q = (\bigcup_{w \in W} V_w) \cap \mathbf{P}$ *is finite,*
- $L : W \to \mathcal{P}(I)$,
- $\dot{V} : W \to Q \to \mathbb{Z}$,
- $\ddot{V} : W \to Q \to \mathcal{P}^{fin}(\mathbb{Z})$, *subject to constraints (below).*

Register Models (2)

Q is the part of the vocabulary that is currently used (compare [ESW11]). Call this the active vocabulary of the model.

$L : W \to \mathcal{P}(I)$ is a function that indicates which agents are listening to broadcasts in world w.

The function $\dot{V}$ maps each $w \in W$ to a function $\dot{V}_w$ that maps all members of Q to integers.

The function $\ddot{V}$ maps each $w \in W$ to a function $\ddot{V}_w$ that maps all members of Q to interpretation domains (finite sets of integers).

Constraints:

- for all $w \in W, q \in Q$: $\dot{V}_w(q) \in \ddot{V}_w(q)$.
- for all $w \in W, q \in Q$: $2 \leq |\ddot{V}_w(q)| < \infty$.

Awareness Constraint

Agents are aware of their state of attention ([DHLoS13]):

- If $L_i \in V_w$ then $(w, w') \in R_i$ implies $L_i \in V_{w'}$.
- If $L_i \notin V_w$ then $(w, w') \in R_i$ implies $L_i \notin V_{w'}$.

In register models satisfying this constraint, the following formulas are true in every world:

$$L_i \leftrightarrow K_i L_i.$$
$$\neg L_i \leftrightarrow K_i \neg L_i.$$

Interpretation in Register Models

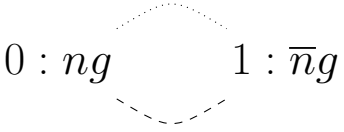
Definition 2 (Register Assignments) *Let a register model $\mathcal{M}$ with world set W, active vocabulary Q and register function $\ddot{V}$ be given. The set of $\mathcal{M}$-register assignments $H \subseteq \mathbb{Z}^Q$ is the set of all functions $h : Q \to \mathbb{Z}$ such that $h(q) \in \bigcup_{w \in W} \ddot{V}_w(q)$ for all $q \in Q$.*

Definition 3 (Agreement) *$h \in H$ agrees with w (notation $w \multimap h$) if it holds for all $q \in Q$ that $h(q) \in \ddot{V}_w(q)$ and moreover, $q \in V_w$ iff $h(q) = \dot{V}_w(q)$.*

For $w \in W$ let $[\![w]\!]^{\mathcal{M}}$ be the set $\{h \in H \mid w \multimap h\}$.

We will use these notions to define $\mathcal{M}, w, h \models \varphi$ for h agreeing with w by recursion.

Examples of (Dis-)Agreement



$n \mapsto 6, \overline{n} \mapsto \{1..4, 7..10\}$
$g \mapsto 5$

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.

$$
\begin{array}{rll}
\mathcal{M}, w, h \models \top & & \text{always} \\
\mathcal{M}, w, h \models p & \text{iff} & p \in V(w) \\
\mathcal{M}, w, h \models L_i & \text{iff} & i \in L(w) \\
\mathcal{M}, w, h \models p_1 = p_2 & \text{iff} & h(p_1) = h(p_2) \\
\mathcal{M}, w, h \models p = N & \text{iff} & h(p) = N
\end{array}
$$

$\mathcal{M}, w, h \models \neg\varphi$ 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 $w' \multimap h'$ imply $\mathcal{M}, w', h' \models \varphi$

$\mathcal{M}, w, h \models [!\, p = E]\varphi$ iff $\mathcal{M}, w, h \models p = E$ implies $\mathcal{M}', (w, e), h' \models \varphi$

where $h' = \lambda(w, e) \mapsto h(w)$

$\mathcal{M}' = \mathcal{M} \circ \mathbf{A}$

$\mathbf{A} = \boxed{\boxed{\mathrm{e}} : p = E, e' : \top, e \overset{I-G}{\longleftrightarrow} e'}$

$\circ$ is product update with adjustment of $\ddot{V}$

G is such that $\mathcal{M}, w, h \models L_G$

$\mathcal{M}, w, h \models [!\, p \neq E]\varphi_2$ iff ...

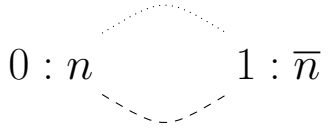
Truth and Falsity at a World

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

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

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

Truth Value Gaps



$n \mapsto 6$
$\overline{n} \mapsto \{1..5, 7..10\}$

$n = 7$ 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.

K_i closes the truth value gap

$\mathcal{M}, w \nmid K_i\varphi$ iff

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

iff

$\forall h$ with $w \multimap h \exists w', h'$ with $wR_iw', w' \multimap h'$ and $\mathcal{M}, w', h' \models \neg\varphi$

iff

$\exists h$ with $w \multimap h \exists w', h'$ with $wR_iw', w' \multimap h'$ and $\mathcal{M}, w', h' \models \neg\varphi$

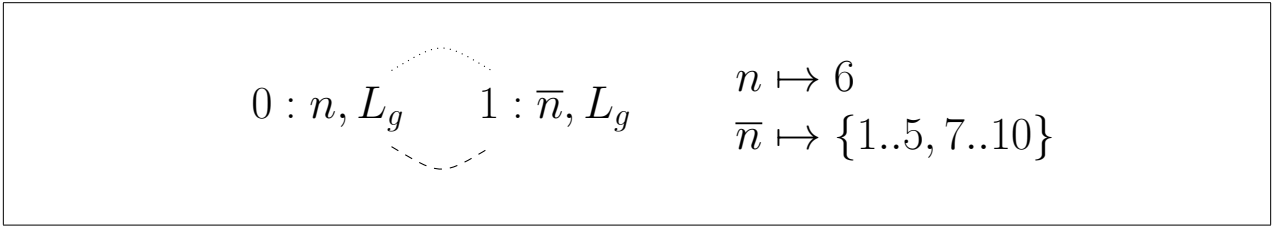
iff

$\exists h$ with $w \multimap h$ and $\mathcal{M}, w, h \not\models K_i\varphi$

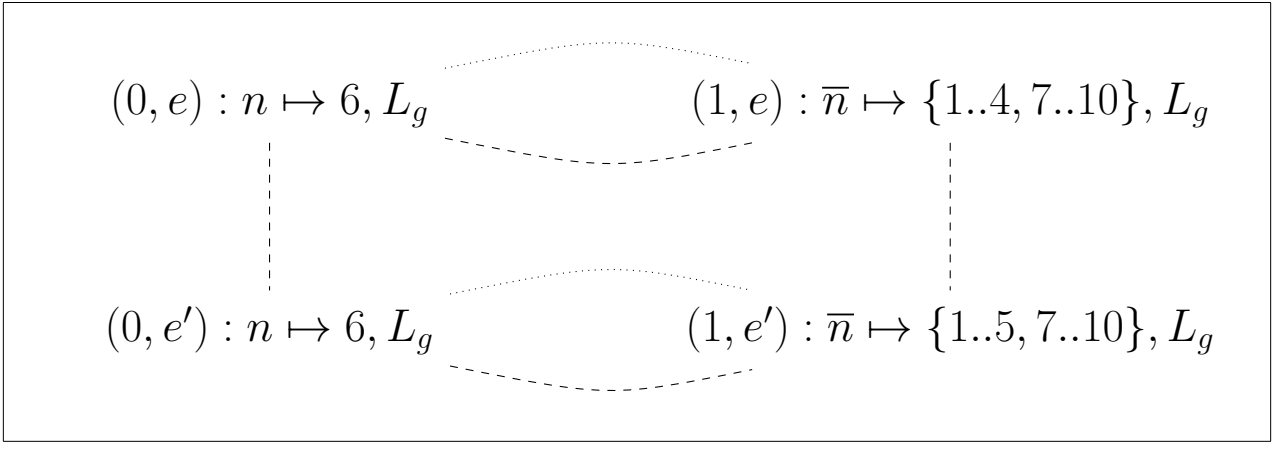
iff

$\mathcal{M}, w \not\models K_i\varphi$.

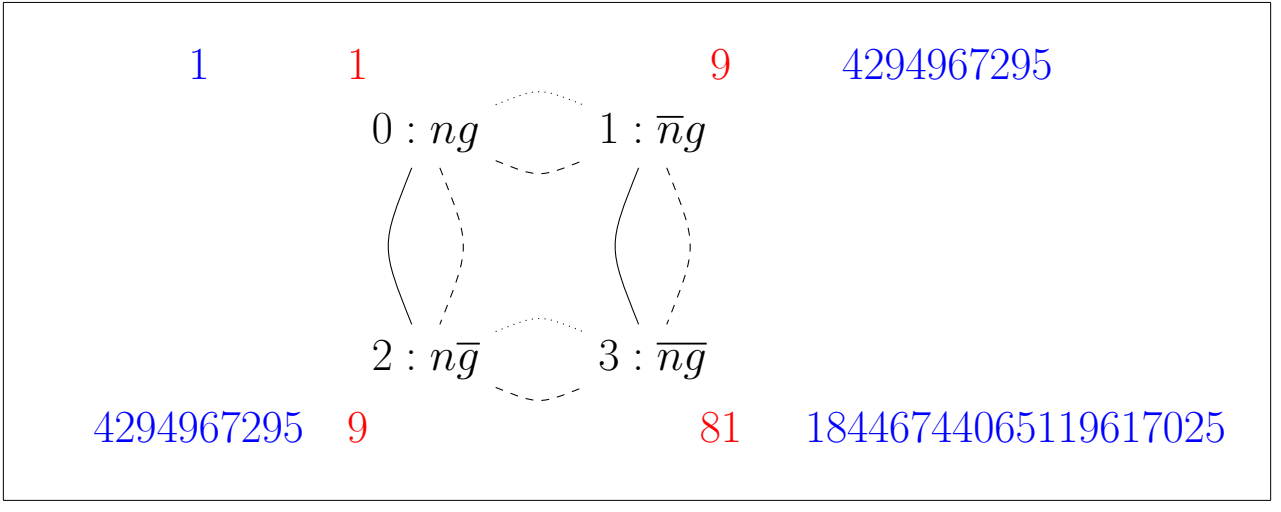
Update Example



Suppose $n \neq 5$ is announced. Since Gaia is listening, we get:



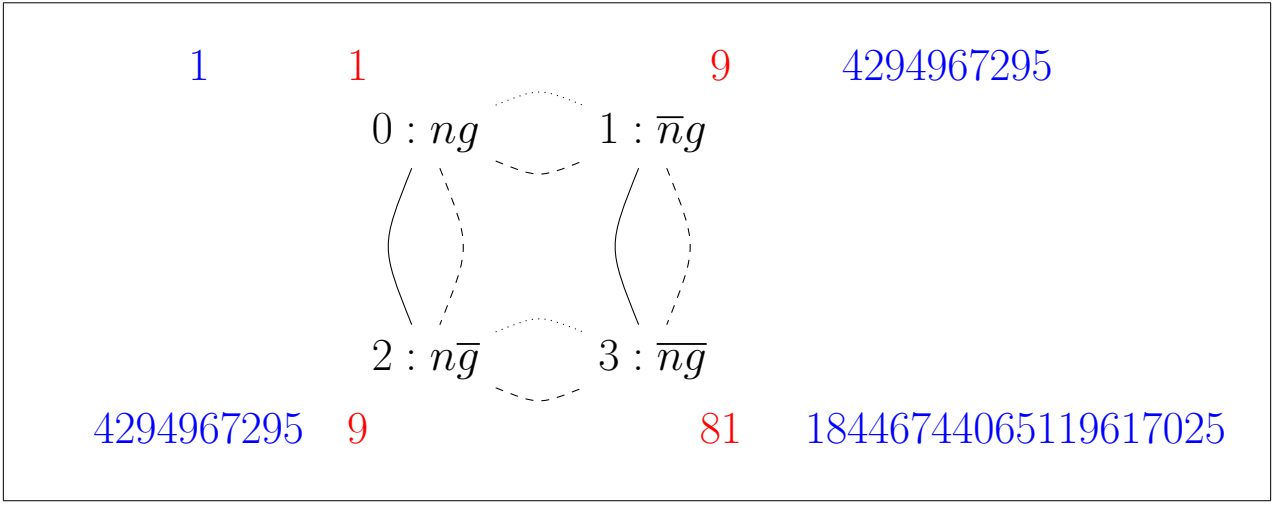
Now Let's Look at Probabilities



Numbers of agreeing assignments for register size 10.

Numbers of agreeing assignments for register size of 32 bits.

Now Let's Look at Probabilities



Numbers of agreeing assignments for register size 10.

Numbers of agreeing assignments for register size of 32 bits.

With 64 bit size: $(2^{64}-1)^2 = 340282366920938463426481119284349108225$ agreeing assignments for world 3.

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.

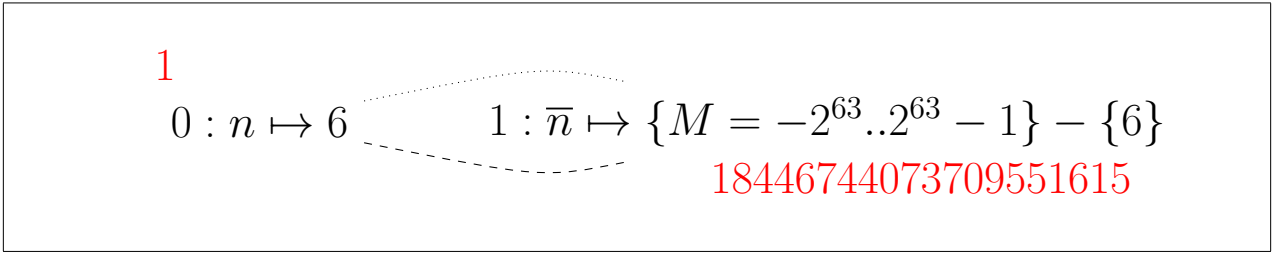
Monte Carlo Style Model Checking

$$\mathcal{M}, w \mathrel{\dot{\models}} \varphi \text{ iff for "enough" } h \text{ with } w \multimap h : \mathcal{M}, w, h \models \varphi.$$

Method:

- Randomly generate a list of n agreeing assignments $h_1, \ldots, h_n$ 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



$n = 7$ 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:

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

$p \overset{i}{\leftarrow} N$ 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: $[\![p \overset{i}{\leftarrow} N]\!]$ as the action model (in the sense of [BMS98]):

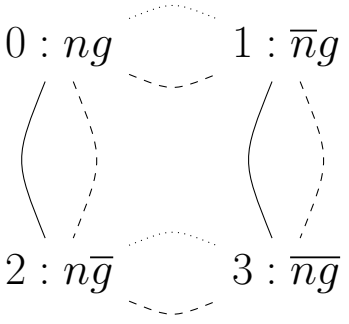
$$\begin{array}{|l} \ddot{p} := \{M..K\}, \dot{p} := N, M \leq N \leq K, \\ \boxed{\mathsf{e}} : p := \top, e' : p := \bot, e \overset{I-\{i\}}{\longleftrightarrow} e' \end{array}$$

Update Example

Start with the blissful ignorance unit model:

0

Execute this command series: $n \overset{h}{\leftarrow} 6;\ \ g \overset{g}{\leftarrow} 5;\ \ !n \neq g.$



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

Communication: "a sends φ to b"

Sequence of commands:

$?K_a\varphi;\ \mathbf{Open}\ b;\ !\varphi;\ \mathbf{Close}\ b$

- $?K_a\varphi$ tests whether $K_a\varphi$ is true in the actual world.
- $\mathbf{Open}\ b$ adds b to the set of agents that are listening.
- $!\varphi$ is the announcement of φ.
- $\mathbf{Close}\ b$ removes b from the set of agents that are listening.
- As far as b is concerned, the information φ 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).
- Fast algorithms for finding modular inverses (Euclid's extended GCD algorithm).
- Fast algorithms for addition and multiplication modulo, and for exponentiation modulo (see, e.g., [DK02, PP09]).
- + Articles of faith: factorisation, discrete logarithm, ..., are not feasible.

Extending the Register Language with Protocols

Let a countable set $\mathbf{P}$ of basic proposition letters and a finite set I of agents be given. Let p range over $\mathbf{P}$ and i over I. Let N range over $\mathbb{Z}$.

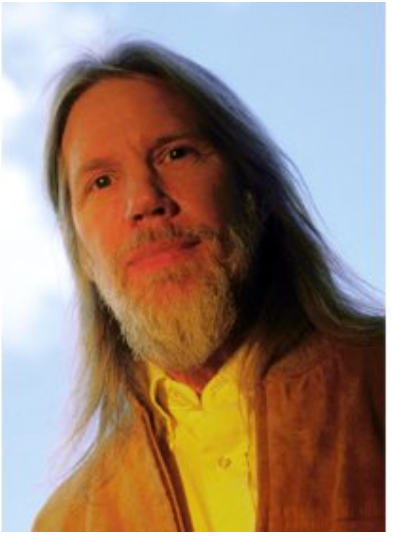
Let $f(\vec{p})$ express any feasible computation from natural numbers named by $\vec{p}$.

$$
\begin{array}{rcl}
\varphi & ::= & \top \mid p \mid L_i \mid p = E \mid \neg\varphi \mid \varphi \wedge \varphi \mid K_i\varphi \mid [A]\varphi \\
E & ::= & p \mid N \\
A & ::= & \mathbf{Open}\ i \mid \mathbf{Close}\ i \mid !\varphi \mid ?\varphi \mid p \stackrel{i}{\leftarrow} N \mid p \stackrel{i}{\leftarrow} f(\vec{p}) \mid \\
 & & A; A \mid \mathbf{if}\ \varphi\ \mathbf{then}\ A\ \mathbf{else}\ A
\end{array}
$$

Informal Explanation

- We will use a broadcast mechanism for communication. L_i expresses that agent i is listening to broadcasts.
- $\mathbf{Open}\ i$ adds i to the set of agents who are listening to broadcasts, while $\mathbf{Close}\ i$ deletes agent i from that set.
- $!\varphi$ is the command to broadcast φ, while $?\varphi$ is the command to check whether φ is true.
- $p \stackrel{i}{\leftarrow} N$ is the command to link p to number N, with the link known only to agent i. $p \stackrel{i}{\leftarrow} f(\vec{p})$ is the command to link p to the number computed by f from $\vec{p}$, with the link known only to agent i. It is assumed that i knows all of $\vec{p}$.

Example: 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)

Theorem 4 (Fermat) *If p is prime, then for every $1 \leq a < p$:*

$$a^{p-1} \equiv 1 \pmod{p}.$$

Example:

```
Prelude> [ a^28 `mod` 29 | a <- [1..28] ]
[1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1]
```

Explanation

We have:

$$(g^a)^{(p-1)-b} = g^{a((p-1)-b)} = g^{a(p-1)} \times g^{-ab} = (g^{p-1})^a \times g^{-ab}$$
$$\overset{Fermat}{=} 1^a \times g^{-ab} = g^{-ab} \bmod p,$$

and therefore:

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

Diffie-Hellman Key Exchange as a Register Language Protocol

$$
\begin{aligned}
& a \overset{i}{\leftarrow} N \;;\; A \overset{i}{\leftarrow} g^a \bmod p \;; \\
& \textbf{Open}\ j \;;\; !x = A \;;\; \textbf{Close}\ j \;; \\
& b \overset{j}{\leftarrow} M \;;\; B \overset{j}{\leftarrow} g^b \bmod p \;; \\
& \textbf{Open}\ i \;;\; !y = B \;;\; \textbf{Close}\ i \;; \\
& k \overset{i}{\leftarrow} y^a \bmod p \;; \\
& k' \overset{j}{\leftarrow} x^b \bmod p \;; \\
& ?k = k'
\end{aligned}
$$

Modelling Attacks on Protocols

The Dolev-Yao model of attacks on public key security protocols [DY83] assumes that the attacker can manipulate the network.

We can model such attacks as interleavings of the security protocol with an attack protocol.

Example attack on Diffie-Hellman key exchange:

Open e **;**

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

Open j **;**

Close i **;** **Close** j **;**

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

$c \overset{e}{\leftarrow} K$ **;** $C \overset{e}{\leftarrow} g^c \bmod p$ **;**

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

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

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

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

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

$?k = k'$

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.

Related Work

[Gat13]: Implementation in Haskell of (a slight variation on) the model checking algorithm presented in this talk.

[MS04, EO05]: Epistemic logic approach to Dining Cryptographers protocol.

[DW07, Wan10] Comparison between approaches based on dynamic epistemic logic and temporal epistemic logic.

[DHLoS13] Earlier approach to model checking cryptographic protocols with DEMO.

[Kra07, CD07] Intricate logics for cryptography, with lots of primitive notions.

[MW07] Epistemic analysis of the notion of 'secure channel'.

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.

[CD07] Mika Cohen and Mads Dams. A complete axiomatisation of knowledge and cryptography. In Proceedings of the 22nd Annual IEEE Symposium on Logic in Computer Science, pages 77–88, 2007.

[DH76] W. Diffie and M. Hellman. New directions in cryp-

tography. IEEE Transactions on Information Theory, 22(6):644–654, 1976.

[DHLoS13] Hans van Ditmarsch, Andreas Herzig, Emiliano Lorini, and Franǫis Schwarzentruber. Listen to me! public announcements to agents that pay attention – or not. In Proceedings of LORI, 2013.

[DK02] Hans Delfs and Helmut Knebl. Introduction To Cryptography — Principles and Applications. Springer, 2002.

[DW07] Francien Dechesne and Yanjing Wang. Dynamic epistemic verification of security protocols: framework and case study. In Johan van Benthem, Shier Ju, and Frank Veltman, editors, A Meeting of the minds: Proceedings of the workshop Logic, Rationality and Interaction, Bei-

jing 2007, Texts in Computer Science. College Publications London, 2007.

[DY83] Danny Dolev and Andrew C. Yao. On the security of public key protocols. IEEE Transactions on Information Theory, 29(2):198–208, March 1983.

[EO05] J. van Eijck and S. Orzan. Modelling the epistemics of communication with functional programming. In Marko van Eekelen, editor, Sixth Symposium on Trends in Functional Programming TFP 2005, pages 44–59, Tallinn, 2005. Institute of Cybernetics, Tallinn Technical University.

[ESW11] Jan van Eijck, Floor Sietsma, and Yanjing Wang. Composing models. Journal of Applied Non-Classical Logics, 21(3-4):397–425, 2011.

[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'.

[Kra07] Simon Kramer. Logical Concepts in Cryptography. PhD thesis, École Polytechnique Fédérale de Lausanne, 2007.

[Mil76] Gary L. Miller. Riemann's hypothesis and tests for primality. Journal of Computer and System Sciences, 13(3):300–317, 1976.

[MS04] Ron van der Meyden and Kaile Su. Symbolic model checking the knowledge of the dining cryptographers. In 17th IEEE Computer Security Foundations Workshop (CSFW'04), page 280, 2004.

[MW07] Ron van der Meyden and Thomas Wilke. Preservation of epistemic properties in security protocol implementations. In Proceedings of TARK'07, pages 212–221, 2007.

[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.

[Wan10] Yanjing Wang. Epistemic Modelling and Protocol Dynamics. PhD thesis, ILLC, Amsterdam, 2010.