Automated Analysis of Non-interference Security by Refinement

Thai Son Hoang [1] Annabelle McIver [2] Larissa Meinicke[2]
Anthony Sloane [2] Enrico Susatyo[2]

[1]Swiss Federal Institute of Technology Zürich (ETH Zürich), Switzerland

[2]Macquarie University, Sydney, Australia

21st June 2011, CryptoForma Workshop
(adapted from slides by Annabelle McIver)

Secure Refinement

- Specialisation of classical refinement;
- Preserves non-interference security properties;
- It is compositional;
- It supports hierarchical program development;
- Its semantics provides a link between "source code" and the "mathematics underlying secrecy".
- Morgan. *The Shadow Knows: Refinement of Ignorance in Sequential Programs*. In Math. Prog. Construction, Springer 2006.

Secure Refinement-oriented Approach

A short history (1/2)

- Traditional refinement reduces non-determinism, preserving all "relevant properties".

$$P \sqcap Q \sqsubseteq P$$

- Traditional formal approaches to security model a "secret" as a non-deterministic choice over its "type".
- Refinement paradox:

$$h :\in \{0,1\} \quad \sqsubseteq \quad h := 0$$
$$h :\in \{0,1\} \quad \not\sqsubseteq_{secure} \quad h := 0$$

- Traditional refinement is defined relative to a flat state space.
- Secure refinement uses a structured state space.

Secure Refinement-oriented Approach

A short history (2/2)

- A secret is an undisclosed choice over a set of possibilities.
- A non-deterministic choice is a disclosed choice, with the selection made as a program is developed.
- The two choices should be distinguished in the semantics.
  - Undisclosed choice cannot (accidentally) be "refined away",
  - so that refinements preserve secrecy.

Refinement with Viewpoints

- Equality between programs: There are no differences between programs, from any agent's viewpoint.
- A secret maintained by program P is also kept by Q if $P = Q$.

The Attack Model

1. During program execution, after each "atomic step":
   - can "look" at the visible variables
   - cannot "look" at the hidden variables
2. Can observe any branching.

- (1) and (2) imply compositionality of refinement.
- A qualitative approach: "run the program only once".

Hidden/Visibles in the Programming Language

- v (of type $\mathcal{V}$) is visible, h (of type $\mathcal{H}$) is hidden.
- H (of type $\mathbb{P}(\mathcal{H})$) – the shadow – the set of possible values of h.
- Program: $[\![P]\!] \in \mathcal{V} \times \mathcal{H} \times \mathbb{P}(\mathcal{H}) \to \mathbb{P}(\mathcal{V} \times \mathcal{H} \times \mathbb{P}(\mathcal{H}))$
- Assume: $v, h \in \{0, 1\}$, initially H is $\{0, 1\}$.

	Program P	$[\![P]\!]\,(v, h, H)$
Set hidden	$h := 0$	$\{(v, 0, \{0\})\}$
	$h :\in \{0, 1\}$	$\{(v, 0, \{0, 1\}), (v, 1, \{0, 1\})\}$
Set visible	$v := 0$	$\{(0, h, \{0, 1\})\}$
	$v :\in \{0, 1\}$	$\{(0, h, \{0, 1\}), (1, h, \{0, 1\})\}$
Swap hidden	$h :\in \{0, 1\}; h := 1 - h$	$\{(v, 0, \{0, 1\}), (v, 1, \{0, 1\})\}$

Secure Refinement Preserves Secrecy

- Refinement: $P_1 \sqsubseteq P_2$, if for all v, h, H, we have

$$\forall (v', h', H'_2) \in [\![P_2]\!]\,(v, h, H) \Rightarrow (\exists H'_1 \subseteq H'_2 \cdot (v', h', H'_1) \in [\![P_1]\!]\,(v, h, H))$$

- *Undisclosed* choice cannot be refined away:
  $$h :\in \{0, 1\} \quad \not\sqsubseteq \quad h := 0$$
- *Disclosed* choice can be refined away
  $$v :\in \{0, 1\} \quad \sqsubseteq \quad v := 0$$

Secure Development

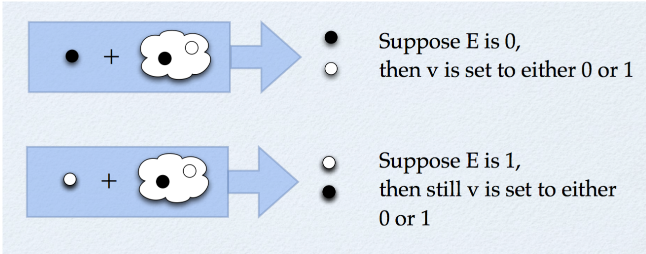
In *secure refinement-oriented* framework:

- we do not say that "a program is secure",
- we write a specification which "obviously" captures our requirements (both functional and security),
- specification summarises the intentions of the designer: inefficient or unimplementable "programs".
- we use refinement to add detail.
- Result: avoid building insecurities into the system.

Modelling Encryption

- Encryption is the most fundamental secure program.
- Publishing the exclusive-or of a "randomly" chosen, hidden bit, reveals nothing about the secret E.

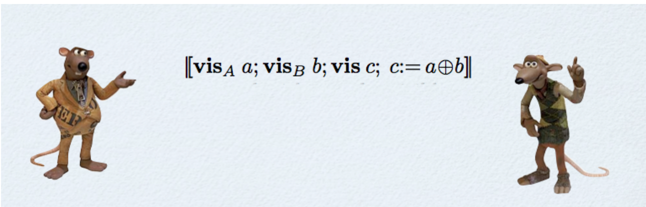
$$\textbf{vis } v; \textbf{hid } h \cdot h :\in \{0,1\}; v := E \oplus h$$

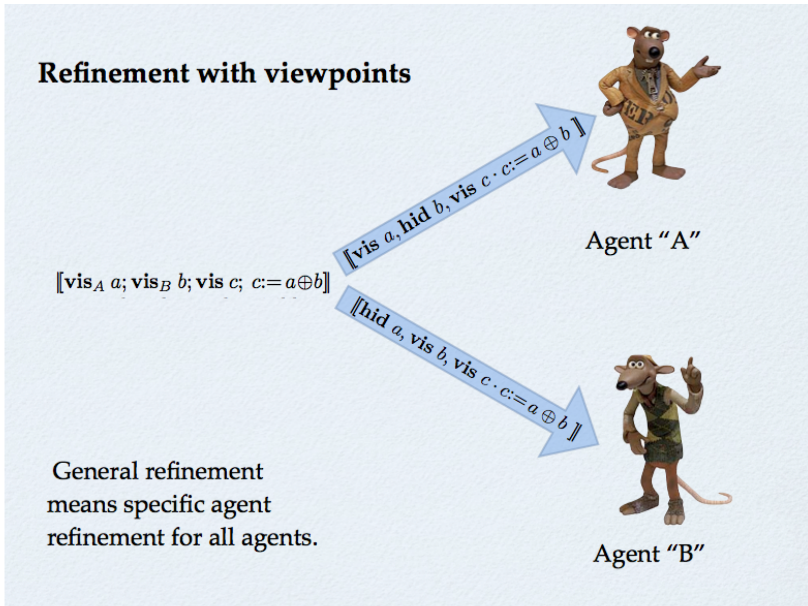


- Encryption is secure: having the same semantics as SKIP (v, h are "local variables").

Refinement with Viewpoints

- **vis** means the associated variable is visible to all agents.
- **hid** means the associated variable is hidden from all agents.
- $\textbf{vis}_{list}$ means the associated variable is visible to all agents in the (non-empty) list, and is hidden from all others (including third parties).
- $\textbf{hid}_{list}$ means the associated variable is hidden from all agents in the list, and is visible to all others (including third parties).





Encryption with Viewpoints

$$\textbf{vis}_A\ a;\ \textbf{vis}_B\ b;\ (a \oplus b) := E$$

where

- $(a \oplus b) := E$: a, b become such that to $a' \oplus b' = E$,
- it is the (atomic) choice over all possibilities of splitting E

The full formal proof of the encryption lemma looks like this

$$
\begin{array}{lll}
 & [\![\ \textbf{vis}_A\ a; \textbf{vis}_B\ b;\quad (a\oplus b) := E\ ]\!] & \text{“from (1)”} \\
= & [\![\ \textbf{vis}_A\ a; \textbf{vis}_B\ b;\quad \langle\!\langle (a\oplus b) := E \rangle\!\rangle\ ]\!] & \text{“statement is atomic already”} \\
= & [\![\ \textbf{vis}_A\ a; \textbf{vis}_B\ b;\quad \langle\!\langle a :\in \mathcal{E}; b := E\oplus a \rangle\!\rangle\ ]\!] & \text{“}\mathcal{E}\text{ is the type of } a, b, E\text{; see (i) below”} \ \heartsuit \\
= & [\![\ \textbf{vis}_A\ a; \textbf{vis}_B\ b;\quad \langle\!\langle a :\in \mathcal{E} \rangle\!\rangle; \langle\!\langle b := E\oplus a \rangle\!\rangle\ ]\!] & \text{“atomicity lemma”} \\
= & [\![\ \textbf{vis}_A\ a; \textbf{vis}_B\ b;\quad a :\in \mathcal{E}; b := E\oplus a\ ]\!] & \text{“statements are atomic anyway”} \\
= & [\![\ \textbf{vis}_A\ a; a :\in \mathcal{E};\quad [\![\ \textbf{vis}_B\ b;\ \ b := E\oplus a\ ]\!]\ ]\!] & \text{“}b\text{ is not free in }\mathcal{E}\text{; see (ii) below”} \ \heartsuit \\
= & [\![\ \textbf{vis}_A\ a;\quad a :\in \mathcal{E}; \textbf{skip}\ ]\!] & \text{“}b\text{ is hidden from }A\text{”} \ \flat \\
= & [\![\ \textbf{vis}_A\ a;\quad a :\in \mathcal{E}\ ]\!] & \text{“}\textbf{skip}\text{”} \\
= & \textbf{skip}\ . & \text{“}a\text{ is a local visible”}
\end{array}
$$

The proof for B's point of view is symmetric.[3] The crucial features ♡ of the derivation are these:

(i) For all E, a there must be some b with b=E ⊕ a.

(ii) The choice range of a is independent from that of b.

Possible Improvement

- Can we automate these proofs?
  - Event-B/Rodin Platform
- Can we strengthen the attack model to something which is closer to the assumptions used in the creation of cryptographic primitives?
  - McIver, Meinicke, Morgan. *Compositional Closure for Bayes Risk in Probabilistic Interference*, ICALP 2010.

Can We Automate These Proofs? (1/4)

- Event-B: modelling discrete transition systems using refinement.
- Event-B is supported by the Rodin Platform.
- A specialised refinement is implemented for the Rodin platform.
- An extra variable H (the “Shadow”) is generated to keep track of the possible values of hidden variables h.
- Extra refinement relations for shadow refinement.
- Rodin generates and discharges many of the obligations related to shadow refinement.
- Interactively prove the remaining obligations within Rodin.

Can We Automate These Proofs? (2/4)

- Difficulty: it was awkward to generate and supply the invariants for the shadow H.
- Solution: Implemented a “front-end” for inputting program directly, using Rodin as a “back-end” for verification.
- The shadow invariants are generated in Rodin.

Can We Automate These Proofs? (3/4)

HID E : X

result: skip;

[=

VIS v : X
HID h : X
FUN ⊕ : X x X -> X

result: v = h ⊕ E

variables: $E, fresult, H1$

result
  when
    $fresult = F$
  then
    $fresult := T$
  end

invariants:
$E \in H1$
$fresult = F \Rightarrow H1 = X$
$fresult = T \Rightarrow (\forall vb \cdot vb \in H1 \Rightarrow vb \in X)$

Can We Automate These Proofs? (4/4)

HID E : X

result: skip;

[=

VIS v : X
HID h : X, E : X
FUN ⊕ : X x X -> X

result: v = h ⊕ E

variables: $E, v, h, fresult, H2$

result
  when
    $fresult = F$
  then
    $fresult := T$
    $v := h \oplus E$
    $H2 := \{vE \mapsto vh \in H2 \mid h \oplus E = vh \oplus vE\}$
  end

invariants:
$E \mapsto h \in H2$
$fresult = F \Rightarrow H2 = X \times X$
$fresult = T \Rightarrow (\forall vE \mapsto vh \in H2 \cdot v = vh \oplus vE)$
$\forall vE \cdot vE \in H1 \Rightarrow (\exists vh \cdot vE \mapsto vh \in H2)$

Strengthen the Attack Model? (1/2)

- McIver, Meinicke, Morgan.
  Compositional Closure for Bayes Risk in Probabilistic Interference,
  ICALP 2010.
- A generalisation of the Shadow Know to deal with probability.
- v (of type $\mathcal{V}$) is visible, h (of type $\mathcal{H}$) is hidden.
- δ (of type $\mathcal{D}(\mathcal{H})$) – a distribution of h.
- Non-deterministic choices, e.g., $h :\in E(v, h)$,
  are interpreted as uniform choice over the value of $E(v, h)$.

Strengthen the Attack Model? (1/2)

We specialise that work

- to determine when Rodin certified proofs maybe lifted
  to the more general probabilistic model,
- to identify a subset of language constructs
  which preserve uniform choices in all contexts.

Sketch ideas:

- Restrict our programs to
  those preserving total uniformity of hidden distribution.
- Assuming uniformity of the initial hidden distribution,
  we can reason about distributions the same way as sets.

Conclusions and Future Work

- We shown how to automate Shadow refinement proofs using Event-B/Rodin.
- The proofs are valid for a restricted sub-sets of language of probabilistic model.
- Future work:
  - Better integration tool support.
  - Applications to other protocols.