Program development with *Secure Refinement*
Program development with
Secure Refinement

a.k.a. The secret art of computer programming
Program development with
Secure Refinement

a.k.a. The secret art of computer programming

Invited talk ICTAC ’09

Annabelle McIver

Carroll Morgan
Secure refinement

- It is a specialisation of classical refinement, more restrictive in order to preserve security properties.

- Those properties are non-interference in style.

- The semantics is compositional.

- It supports hierarchical refinement-based program development.

- It provides a link between source-code reasoning and the “mathematics underlying secrecy.”
Secure refinement

• It is a specialisation of classical refinement, more restrictive in order to preserve security properties.

• Those properties are non-interference in style.

• The semantics is compositional.

• It supports hierarchical refinement-based program development.

• It provides a link between source-code reasoning and the “mathematics underlying secrecy.”
Example: A Gossip’s Tale

My dear Lady Bracknell,

Have you heard the latest about that disgraceful Featherstonhaugh girl?

Let me tell you...
Example: A Gossip’s Tale

My dear Lady Bracknell,

Have you heard the latest about that disgraceful Featherstonhaugh girl?

Let me tell you...

Nick’s Personal Delivery: “From door to door, by my very own paw!”

Fletcher’s Trusted Couriers: “Get it there from anywhere!”
My dear Lady Bracknell,

Have you heard the latest about that disgraceful Featherstonhaugh girl?

Let me tell you...
Example: A Gossip’s Tale
My dear Lady Bracknell,

Have you heard the latest about that disgraceful Featherstonhaugh girl?

Let me tell you...
“How do I know that the rats can’t read my secret? And even if the protocol works in principle, how do I know it will be properly coded?”
Example: A Gossip’s Tale

And how do I even describe these requirements in ordinary program-specification notation?

Even if the protocol works in principle, how do I know it will be properly coded?”
Example: A Gossip’s Tale

And how do I even describe these requirements in ordinary program-specification notation?

How do I know that the rats can't read my secret? Even if the protocol works in principle, how do I know it will be properly coded?

Do I need to adapt or extend my programming/specification language? Can I do that and still understand it?
Coding: security by contract

The specification/contract: \texttt{vis}_S s; \texttt{vis}_R r; \quad r := s

Language extensions...
Coding: security by contract

The specification/contract: \( \text{vis}_S s; \text{vis}_R r; \ r := s \)

- **sender (agent) S**
- **variable s to be sent**
- **receiver (agent) R**
- **variable r to take receipt**
- **action to be carried out**

Language extensions...
The Ladies’ protocol:

\[ \text{vis}_s s; \text{vis}_r r; \]
\[ \text{vis}_{sx} s_x; \text{vis}_{sy} s_y; \]
\[ \text{vis}_x x; \text{vis}_y y; \]
\[ \text{vis}_{rx} r_x; \text{vis}_{ry} r_y; \]

\[(s_x \oplus s_y) := s; \quad \Leftarrow S \text{ splits the message in two.}\]
\[x,y := s_x,s_y; \quad \Leftarrow \text{Messages sent from } S \text{ to } X \text{ and to } Y \text{ separately.}\]
\[r_x,r_y := x,y; \quad \Leftarrow \text{Messages sent from } X \text{ and } Y \text{ to } R.\]
\[r := r_x \oplus r_y. \quad \Leftarrow R \text{ recombines the two halves.}\]

We write \((s_x \oplus s_y) := s\) for the (atomic) choice over all possibilities of splitting the message \(s\), equivalent to the specification statement \(s_x, s_y : [s_x \oplus s_y = s]\) and interpreted atomically.

**Fig. 2.** Abstract messaging with non-colluding messengers
Secure refinement: meeting the contract

\[ \text{vis}_S \ s; \ \text{vis}_R \ r; \quad r := s \]

is refined by, is implemented by

\[
\begin{align*}
\text{vis}_S \ s; & \ \text{vis}_R \ r; \\
\text{vis}_{SX} \ s_x; & \ \text{vis}_{SY} \ s_y; \\
\text{vis}_X \ x; & \ \text{vis}_Y \ y; \\
\text{vis}_{RX} \ r_x; & \ \text{vis}_{RY} \ r_y; \\
(s_x \oplus s_y) := s; & \quad \Leftarrow S \text{ splits the message in two.} \\
x, y := s_x, s_y; & \quad \Leftarrow \text{Messages sent from } S \text{ to } X \text{ and to } Y \text{ separately.} \\
r_x, r_y := x, y; & \quad \Leftarrow \text{Messages sent from } X \text{ and } Y \text{ to } R. \\
r := r_x \oplus r_y. & \quad \Leftarrow R \text{ recombines the two halves.}
\end{align*}
\]
Private information retrieval: a more realistic example

A user \( U \) has a query \( c \) to make of a remote database \( D \); the answer arrives in variable \( u \). This is effectively the specification

\[
u := D \cdot c,
\]

to which we add the informally stated security requirement that “nobody but \( U \) is to know what \( c \) is — not even the database server.”

With the vocabulary established above, we stipulate that $u$ and $c$ both have attribute $\text{vis}_U$, while database $D$, being public, has attribute $\text{vis}$ (i.e. unadorned).

To solve the problem, we duplicate the database so that we have $D_A$ and $D_B$, and arrange for two separate lookups to be done by two separate agents $A, B$ who –crucially– do not collude.

Neither one can figure out –we will see– from his own lookup what it was that $U$ actually wanted.
Thus we will be featuring...

- Why the specifications capture our security as well as functional requirements.
- How we deal with multiple agents, and...
- ...how we say that they cannot collude.
- What “secure refinement” actually secures.
- Why it is worth trying to do things this way.
In traditional relational models of sequential programs, refinement reduces nondeterminism: in this framework refinement preserves all “relevant properties,” thus

\[ P \cap Q \subseteq P \]

Traditional formal approaches to security model a “secret” as a nondeterministic choice over its type. But there’s a paradox: the specification of secrets using traditional semantics allows refinements of secure programs to insecure programs, so that security properties are not necessarily preserved:

\[ h := 0 \cap h := 1 \sqsubseteq h := 0 \]
Traditional refinement is defined relative to a flat state space, and models programs as relations from initial to sets of final states, with nondeterminism being represented as subsets of states.

Secure refinement uses a structured state space, using that extra structure to distinguish automatically between “unsafe” refinements and those which are “safe” from a security perspective.
The difference between a secret and a choice

A secret is an *undisclosed* choice over a given set of possibilities. A nondeterministic choice is also selected from a range of possibilities, with the selection made e.g. as a program is developed: those choices are regarded as *disclosed* ...

... but programs operating in a context of undisclosed choices have distinct behaviour from those operating under disclosed choices, and therefore those two choices should be distinguished in the semantics.
Inside the box there is a sphere which is either blue or a green.

The crucial property of secrets is that observers can’t see them, and don’t know what they are until they are disclosed.

How do we model the secrecy?
An ordinary box: disclosed

Ordinary boxes, on the other hand always display their contents.

Observers can rely on the accuracy of what they see.

In other words, our semantics distinguishes between open and closed boxes.
Swap the colours in each of the boxes — and what difference can the observer notice?
Programming with the two kinds of box

On the left he sees a difference; but on the right he does not.
Schrödinger’s box looks exactly the same.
Programming with the two kinds of box

Until it is opened, and then
the uncertainty “collapses” and we can see the colour
Programming with the two kinds of box

...whatever it turns out to be: one, or the other.
The **formal semantics** uses relational-style models inspired by “Kripke semantics” to distinguish between undisclosed and disclosed choices, between the ordinary and the Schrödinger boxes.

Undisclosed choices can’t (accidentally) be “refined away”, so that refinements actually preserve the crucial aspect of secrets — that they always span a range of possibilities.
The state space is structured using designated "hidden" (h) and "visible" (v) states. An element in the structured state space is a pair (v,H), where H is a subset of hidden states and determines the correlation between the observed state and the hidden state. Programs are transitions of (v,H)-states; an adversary can always observe v during a computation, but can only deduce facts about H.
A visible, or disclosed choice between two visible variables is represented by a subset containing several pairs. (In this example there is no particular hidden state.)

\{(blue, {...}), (green, {...})\}
A visible, or disclosed choice between two visible variables is represented by a subset containing several pairs. (In this example there is no particular hidden state.)

\{(blue, {...}), (green, {...})\}
A visible, or disclosed choice between two visible variables is represented by a subset containing several pairs. (In this example there is no particular hidden state.)

\{ (blue, \{\ldots\}), (green, \{\ldots\}) \}
Refinement of a visible choice is possible, and in the semantics it reduces the result set without revealing any secret.
But we can’t refine away a secret, i.e. an undisclosed choice, because there’s only one of it anyway.

In the semantics it’s modelled as a singleton set, containing just one pair (not two). The “Schrödinger choice” is within that pair.
Hiddens and visibles in the programming language

Set hidden: \texttt{hid }h;\quad h \in \{0, 1\} \quad \{(v, \{0,1\})\}

Set visible: \texttt{vis }v;\quad v \in \{0, 1\} \quad \{(0, H), (1, H)\}

Swap hidden: \texttt{hid }h;\quad h \in \{0, 1\};\quad h := 1-h \quad \{(v, \{0,1\})\}
Hiddens and visibles in the programming language

Set hidden: \textbf{hid} \ h; \quad h : \in \{0, 1\} \quad \{(v, \{0, 1\})\}

Set visible: \textbf{vis} \ v; \quad v : \in \{0, 1\} \quad \{(0, H), (1, H)\}

Swap hidden: \textbf{hid} \ h; \quad h : \in \{0, 1\}; \quad h := 1 - h \quad \{(v, \{0, 1\})\}
Refinement must preserve secrecy

Undisclosed choice cannot be refined away:

\[
\text{hid } h; \quad h \in \{0, 1\} \not\subseteq h := 0
\]

Disclosed choice can be refined away:

\[
\text{vis } v; \quad v \in \{0, 1\} \subseteq v := 0
\]
Reveal $h$: $h \in \{0, 1\}; \cdots \left[ \text{vis } v; \ v := h \right]$
Revelations

Reveal $h$: $h \in \{0, 1\}; \cdots \lbrack \text{vis } v; \mid v := h \rbrack$

$\{(\bullet, H)\} \rightarrow \{((\bullet, v), \{0, 1\})\}$
Reveal $h$: $h \in \{0, 1\}; \cdots \left[ \text{vis } v; \ v := h \right]\}$

$\{(\bullet, H)\} \rightarrow \{((\bullet, v), \{0, 1\})\} \rightarrow \{((\bullet, 0), \{0\}, ((\bullet, 1), \{1\})\}$
Reveal $h$:

$$h \in \{0, 1\}; \cdots \left[ \text{vis } v; \ v := h \right]$$

$$\{(\bullet, H)\} \rightarrow \{((\bullet, v), \{0, 1\})\} \rightarrow \{(((\bullet, 0), \{0\}), ((\bullet, 1), \{1\}))\} \rightarrow \{((\bullet, \{0\}), (\bullet, \{1\}))\}$$
Revelations

Reveal $h$:  
\[ h \in \{0, 1\}; \cdots | [ \text{vis } v; \ v := h ] \]

\{($\bullet$, $H$)}  
\{(($\bullet$, $v$), $\{0, 1\}$)}  
\{(($\bullet$, $0$), $\{0\}$), (($\bullet$, $1$), $\{1\}$)}

Encrypt $h$:  
\| [ \text{hid } h'; \text{vis } v; \ h' \in \{0, 1\}; v := h \oplus h' ] \|

\{($\bullet$, $H$)}
Revelations

Reveal $h$: $h \in \{0, 1\}; \cdots \vert[\text{vis } v; v := h]\vert$

$\{ (\bullet, H) \} \rightarrow \{ ((\bullet, v), \{0,1\}) \} \rightarrow \{ ((\bullet, 0), \{0\}), ((\bullet, 1), \{1\}) \}$

Encrypt $h$:

$\vert[\text{hid } h'; \text{vis } v; h' \in \{0, 1\}; v := h \oplus h']\vert$

$\{ (\bullet, H) \} \rightarrow \{ ((\bullet, v), H \times \{0,1\}) \}$
Revelations

Reveal $h$: 
\[ h \in \{0, 1\}; \cdots \left\lceil \text{vis} \; v; \; v := h \right\rceil \]
\[ \{ (\bullet, H) \} \rightarrow \{ ((\bullet, v), \{0,1\}) \} \rightarrow \{ ((\bullet,0), \{0\}), ((\bullet,1),\{1\}) \} \]

Encrypt $h$: 
\[ \left\lceil \text{hid} \; h'; \text{vis} \; v; \; h' \in \{0, 1\}; v := h \oplus h' \right\rceil \]
\[ \{ (\bullet, H) \} \rightarrow \{ ((\bullet, v), H \times \{0,1\}) \} \rightarrow \{ ((\bullet,0), H \times \{0,1\}) , ((\bullet,1), H \times \{0,1\}) \} \]
Revelations

Reveal $h$:  

\[ h \in \{0, 1\}; \ldots | [ \text{vis } v; \ v := h ] |\]

\[
\{(\bullet, H)\} \rightarrow \{(\bullet,0), (\bullet,1), (\bullet, v), \{0,1\}\} \rightarrow \{(\bullet,0), (\bullet,1)\}
\]

Encrypt $h$:

\[ | [ \text{hid } h'; \text{vis } v; \ h' \in \{0, 1\}; v := h \oplus h' ] | \]

\[
\{(\bullet, H)\} \rightarrow \{(\bullet, v), H \times \{0,1\}\} \rightarrow \{(\bullet,0), H_0 \times \{0\} \cup H_1 \times \{1\}\}
\]
Revelations

Reveal $h$:

$h : \in \{0, 1\}; \cdots | [ \text{vis } v; \ v := h ]$

\[
\{ (\bullet, H) \}\quad \{ ((\bullet,v), \{0,1\}) \}\quad \{ ((\bullet,0), \{0\}), ((\bullet,1),\{1\}) \}
\]

Encrypt $h$:

$| [ \text{hid } h'; \text{vis } v; \ h' : \in \{0, 1\}; v := h \oplus h' ] |$

\[
\{ (\bullet, H) \}\quad \{ ((\bullet,v), H \times \{0,1\}) \}\quad \{ (\bullet, H) \}, \quad (\bullet, H)\}
\]
Reveal $h$: $h \in \{0, 1\}; \ldots \mid [\texttt{vis } v; \ v := h ]$

$\{(\bullet, H)\}$ $\{((\bullet, v), \{0,1\})\}$ $\{((\bullet,0), \{0\}), ((\bullet,1),\{1\})\}$ $\{\bullet, \{0\}, (\bullet,\{1\})\}$

Encrypt $h$: $\mid [\texttt{hid } h'; \texttt{vis } v; \ h' \in \{0, 1\}; v := h \oplus h' ]$

$\{(\bullet, H)\}$ $\{((\bullet, v), H \times \{0,1\})\}$ $\{((\bullet, H)\}$
But are we any closer to helping Mrs. Bennet?
The structuring between hidden and visible state defines a viewpoint for each agent in the system: the semantics for each agent’s viewpoint — say Agent $A$ — is determined by considering $\text{vis}_A$ variables to be (simply) $\text{vis}$, and all the others to be $\text{hid}$.
My dear Lady Bracknell,

Have you heard the latest about that disgraceful Featherstonhaugh girl?

Let me tell you...
My dear Lady Bracknell,

Have you heard the latest about that disgraceful Featherstonhaugh girl?

Let me tell you...
Nick’s viewpoint

M d a L d B a k e l
H v y u h a d t e l t s
a o t t a d s r c f l
F a h r t n h u h g r?
L t m t l y u..

Send
Fletcher’s viewpoint

Send

yerayrcnl,
aeoerhaet
buhitigaeu
eteseagil
elleelo.
My dear Lady Bracknell,

Have you heard the latest about that disgraceful Featherstonhaugh girl?

Let me tell you...

Lady Bracknell’s viewpoint

Decrypt

My dear Lady Bracknell,

Have you heard the latest about that disgraceful Featherstonhaugh girl?

Let me tell you...
My dear Lady Bracknell,
Have you heard the latest about that disgraceful Featherstonhaugh girl?
Let me tell you...

My dear Lady Bracknell,
Have you heard the latest about that disgraceful Featherstonhaugh girl?
Let me tell you...

Four viewpoints in the gossiping scenario

Encrypt

Send

Decrypt
Mrs Bennet’s point of view — she is “Agent $S$.”

\[
\begin{align*}
\text{vis}_S s; \text{vis}_R r; \\
\text{vis}_{SX} s_x; \text{vis}_{SY} s_y; \\
\text{vis}_X x; \text{vis}_Y y; \\
\text{vis}_{RX} r_x; \text{vis}_{RY} r_y; \\
(s_x \oplus s_y) := s; & \quad \Leftarrow S \text{ splits the message in two.} \\
x, y := s_x, s_y; & \quad \Leftarrow \text{Messages sent from } S \text{ to } X \text{ and to } Y \text{ separately.} \\
r_x, r_y := x, y; & \quad \Leftarrow \text{Messages sent from } X \text{ and } Y \text{ to } R. \\
r := r_x \oplus r_y. & \quad \Leftarrow R \text{ recombines the two halves.}
\end{align*}
\]

We write \((s_x \oplus s_y) := s\) for the (atomic) choice over all possibilities of splitting the message \(s\), equivalent to the specification statement \(s_x, s_y : [s_x \oplus s_y = s]\) and interpreted atomically.

\textbf{Fig. 2.} Abstract messaging with non-colluding messengers
Mrs Bennet’s point of view — she is “Agent S.”

\[ (s_x \oplus s_y) := s; \quad \leftarrow S \text{ splits the message in two.} \]
\[ x, y := s_x, s_y; \quad \leftarrow \text{Messages sent from } S \text{ to } X \text{ and to } Y \text{ separately.} \]
\[ r_x, r_y := x, y; \quad \leftarrow \text{Messages sent from } X \text{ and } Y \text{ to } R. \]
\[ r := r_x \oplus r_y. \quad \leftarrow R \text{ recombines the two halves.} \]

We write \((s_x \oplus s_y) := s\) for the (atomic) choice over all possibilities of splitting the message \(s\), equivalent to the specification statement \(s_x, s_y : [s_x \oplus s_y = s]\) and interpreted atomically.

Fig. 2. Abstract messaging with non-colluding messengers
Mrs Bennet’s point of view — she is “Agent S.”

\[ s; \text{vis}_R \ r; \]
\[ s_x; \text{vis}_{SY} \ s_y; \]
\[ \text{vis}_X \ x; \text{vis}_Y \ y; \]
\[ \text{vis}_{RX} \ r_x; \text{vis}_{RY} \ r_y; \]

\[(s_x \oplus s_y) := s; \quad \Leftarrow S \text{ splits the message in two.}\]
\[x, y := s_x, s_y; \quad \Leftarrow \text{Messages sent from } S \text{ to } X \text{ and to } Y \text{ separately.}\]
\[r_x, r_y := x, y; \quad \Leftarrow \text{Messages sent from } X \text{ and } Y \text{ to } R.\]
\[r := r_x \oplus r_y. \quad \Leftarrow R \text{ recombines the two halves.}\]

We write \((s_x \oplus s_y) := s\) for the (atomic) choice over all possibilities of splitting the message \(s\), equivalent to the specification statement \(s_x, s_y: [s_x \oplus s_y = s]\) and interpreted atomically.

**Fig. 2.** Abstract messaging with non-colluding messengers
Mrs Bennet’s point of view — she is “Agent S.”

\[
\begin{align*}
&\text{vis } s; \text{vis}_R r; \\
&\text{vis } s_x; \text{vis } s_y; \\
&\text{vis}_X x; \text{vis}_Y y; \\
&\text{vis}_{RX} r_x; \text{vis}_{RY} r_y;
\end{align*}
\]

\[
\begin{align*}
(s_x \oplus s_y) &:= s; & \Leftarrow S \text{ splits the message in two.} \\
x, y &:= s_x, s_y; & \Leftarrow \text{Messages sent from } S \text{ to } X \text{ and to } Y \text{ separately.} \\
r_x, r_y &:= x, y; & \Leftarrow \text{Messages sent from } X \text{ and } Y \text{ to } R. \\
r &:= r_x \oplus r_y. & \Leftarrow R \text{ recombines the two halves.}
\end{align*}
\]

We write \((s_x \oplus s_y) := s\) for the (atomic) choice over all possibilities of splitting the message \(s\), equivalent to the specification statement \(s_x, s_y : [s_x \oplus s_y = s]\) and interpreted atomically.

\textbf{Fig. 2.} Abstract messaging with non-colluding messengers
Mrs Bennet’s point of view — she is “Agent S.”

\[
\begin{align*}
\text{vis} & \quad s; \quad \text{hid} \quad r; \\
\text{vis} & \quad s_x; \quad \text{vis} \quad s_y; \\
\text{hid} & \quad x; \quad \text{hid} \quad y; \\
\text{hid} & \quad r_x; \quad \text{hid} \quad r_y; \\
(s_x \oplus s_y) := s; & \quad \Leftarrow S \text{ splits the message in two.} \\
x, y := s_x, s_y; & \quad \Leftarrow \text{Messages sent from } S \text{ to } X \text{ and to } Y \text{ separately.} \\
r_x, r_y := x, y; & \quad \Leftarrow \text{Messages sent from } X \text{ and } Y \text{ to } R. \\
r := r_x \oplus r_y. & \quad \Leftarrow R \text{ recombines the two halves.}
\end{align*}
\]

We write \((s_x \oplus s_y) := s\) for the (atomic) choice over all possibilities of splitting the message \(s\), equivalent to the specification statement \(s_x, s_y : [s_x \oplus s_y = s]\) and interpreted atomically.

**Fig. 2.** Abstract messaging with non-colluding messengers
Lady Bracknell’s point of view — she is “Agent R.”

\[
\begin{align*}
(s_x \oplus s_y) &:= s; & \Leftarrow S \text{ splits the message in two.} \\
x, y &:= s_x, s_y; & \Leftarrow \text{Messages sent from } S \text{ to } X \text{ and to } Y \text{ separately.} \\
r_x, r_y &:= x, y; & \Leftarrow \text{Messages sent from } X \text{ and } Y \text{ to } R. \\
r &:= r_x \oplus r_y. & \Leftarrow R \text{ recombines the two halves.}
\end{align*}
\]

We write \((s_x \oplus s_y) := s\) for the (atomic) choice over all possibilities of splitting the message \(s\), equivalent to the specification statement \(s_x, s_y : [s_x \oplus s_y = s]\) and interpreted atomically.

**Fig. 2.** Abstract messaging with non-colluding messengers
The specification from X’s viewpoint:

\[
\text{hid } r, s; \ r \ := \ s
\]

The protocol from X’s viewpoint:

\[
\begin{align*}
\text{hid} \ s_y, y, r_y; \\
\text{vis} \ s_x, x, r_x; \\
(s_x \oplus s_y) \ := \ s; \\
x, y \ := \ s_x, s_y; \\
r_x, r_y \ := \ x, y; \\
r \ := \ r_x \oplus r_y
\end{align*}
\]
The specification from $X$’s viewpoint:

\[ \text{hid } r, s; \quad r := s \]

The protocol from $X$’s viewpoint:

\[
\begin{align*}
\text{hid } s_y, y, r_y; \\
\text{vis } s_x, x, r_x; \\
(s_x \oplus s_y) := s; \\
x, y := s_x, s_y; \\
r_x, r_y := x, y; \\
r := r_x \oplus r_y
\end{align*}
\]

\[ r := s \]
Refinement from Nick’s point of view

The specification from X’s viewpoint:

\[ \text{hid } r, s; \quad r := s \]

The protocol from X’s viewpoint:

\[ \text{hid } r, s \]

\[ \text{vis } s_x, x, r_x; \]

\[ (s_x \oplus s_y) := s; \]

\[ x, y := s_x, s_y; \]

\[ r_x, r_y := x, y; \]

\[ r := r_x \oplus r_y \]

\[ r := s \]
Refinement from Nick’s point of view

The specification from X’s viewpoint:

\[ \text{hid } r, s; \quad r := s \]

The protocol from X’s viewpoint:

\[ \text{hid } s_y, y, r_y; \]
\[ \text{vis } s_x, x, r_x; \]
\[ (s_x \oplus s_y) := s; \]
\[ x, y := s_x, s_y; \]
\[ r_x, r_y := x, y; \]
\[ r := r_x \oplus r_y \]
\[ r := s \]
Refinement from Nick’s point of view

The specification from X’s viewpoint:

\[ \text{hid } r, s; \; r := s \]

The protocol from X’s viewpoint:

\[ \begin{align*}
\text{hid } s_y, y, r_y; \\
\text{vis } s_x, x, r_x; \\
(\text{sk}) \quad (s_x \oplus s_y) := s; \\
x, y := s_x, s_y; \\
r_x, r_y := x, y; \\
r := r_x \oplus r_y \\
r := s
\end{align*} \]
The Encryption Lemma, proved formally

\[
\begin{align*}
&\left[\text{vis}_A\ a;\ \text{vis}_B\ b;\ (a\oplus b) := E\right]\quad\text{“from (1)”} \\
= &\left[\text{vis}_A\ a;\ \text{vis}_B\ b;\ \langle\langle a\oplus b) := E\rangle\rangle\right]\quad\text{“statement is atomic already”} \\
= &\left[\text{vis}_A\ a;\ \text{vis}_B\ b;\ \langle\langle a :\in\ E;\ b := E\oplus a\rangle\rangle\right]\quad\text{“atomicity lemma”} \\
= &\left[\text{vis}_A\ a;\ \text{vis}_B\ b;\ a :\in\ E;\ b := E\oplus a\right]\quad\text{“statements are atomic anyway”} \\
= &\left[\text{vis}_A\ a;\ a :\in\ E;\ \left[\ \text{vis}_B\ b;\ b := E\oplus a\ \right]\right]\quad\text{“b is not free in } E;\ \text{see (ii) below”} \\
= &\left[\text{vis}_A\ a;\ a :\in\ E;\ \text{skip}\right]\quad\text{“b is hidden from } A\text{”} \\
= &\left[\text{vis}_A\ a;\ a :\in\ E\right]\quad\text{“skip”} \\
= &\text{skip}. \\
\end{align*}
\]

The proof for \(B\)’s point of view is symmetric.\(^3\) The crucial features \(\heartsuit\) of the derivation are these:

\(\heartsuit\)  (i) For all \(E\) and \(a\) there must be some \(b\) with \(b = E\oplus a\).
(\(ii\)) The range of \(a\) is independent of the value of \(b\).
Secure use of a remote super-computer
Secure use of a remote super-computer

But Mrs. B doesn’t want to tell the supercomputer what x is...
Secure use of a remote super-computer

Using *two* copies of the supercomputer, a similar message-passing scheme maintains secrecy...

...provided function $F$ distributes exclusive-or.
\[ y := F.x \]

\[ = \begin{array}{c}
\left[ \text{vis}_A a; \text{vis}_B b; \ (a \oplus b) := x \right]; \\
y := F.x
\end{array} \]

\[ = \begin{array}{c}
\left[ \text{vis}_A a; \text{vis}_B b; \\
(a \oplus b) := x; \\
y := F.(a \oplus b)
\right]
\end{array} \]

\[ = \begin{array}{c}
\left[ \text{vis}_A a; \text{vis}_B b; \\
(a \oplus b) := x; \\
y := F.a \oplus F.b
\right]
\end{array} \]

- “Encryption Lemma”
- “scope and context”
- “⊕-distributivity of \( F \)”
Let $U$’s secret request be some $1 \leq c \leq N$, and he wants to know $D.c$ (equivalently $D_A.c$ or $D_B.c$). He chooses randomly a subset $S \in \mathbb{P}N$, and then sends (all of) $S$ to $A$ and $S \odot c$ to $B$, where

$$S \odot c := \text{if } (c \in S) \text{ then } S \setminus c \text{ else } S \cup \{c\} \text{ fi.}$$

Next $A$ sends to $U$ the result $y_A := (\oplus_{i \in S} D_A.i)$, and $B$ similarly sends $y_B := (\oplus_{i \in S \odot c} D_B.i)$; finally $U$ decrypts the two replies by computing $y_A \oplus y_B$.

B Chor, O Goldreich, E Kushilevitz, and M Sudan. Private information retrieval. 
Again, an instantiation

\[ u := D . c \]

\[ \equiv \begin{array}{l}
\text{vis}_U x_A, x_B : \mathbb{PN}; y_A, y_B : \mathbb{Bool}; \\
\text{vis}_A a : \mathbb{PN}, z_A : \mathbb{Bool}; \text{vis}_B b : \mathbb{PN}, z_B : \mathbb{Bool}; \\
(x_A \Delta x_B) := \{ c \}; & \leftarrow \text{Split } c \text{ into two "subset" shares.} \\
(a, b := x_A, x_B); & \leftarrow \text{Send to the servers separately.} \\
z_A := (\oplus_{i \in a} D_A . i); & \leftarrow \text{Each computes the } \oplus \text{ of its shares.} \\
z_B := (\oplus_{i \in b} D_B . i); & \leftarrow \ldots \\
y_A, y_B := z_A, z_B; & \leftarrow \text{Each sends the result back to the requester.} \\
u := y_A \oplus y_B & \leftarrow \text{The results are } \oplus\text{-ed together.}
\end{array} \]
Informally...

The protocol for two independent-but-equal servers $D_A, D_B$:

1. User computes in secret two shares for his request;

2. Each share is a subset selected at random (almost) over the whole set of requests;

3. The user sends one subset-share to $D_A$, and the other subset-share to $D_B$; each server looks up all requests in its share, computes the exclusive-or and sends the result back to the user;

4. The user takes the exclusive-or of the result.
Which works because...

$$\overline{D}.(a \Delta b)$$

$$= \overline{D}.((a-b) \cup (b-a))$$

$$= \overline{D}.(a-b) \oplus \overline{D}.(b-a)$$

$$= \overline{D}.((a-b) \oplus \overline{D}.(a \cap b))$$

$$\oplus \overline{D}.((b-a) \oplus \overline{D}.(b \cap a))$$

$$= \overline{D}.a \oplus \overline{D}.b$$

$$\overline{D}.a = \bigoplus_{i \in a} D.i$$
Which works because...

\[\overline{D}.(a \Delta b)\]
\[= \overline{D}.((a-b) \cup (b-a))\]
\[= \overline{D}.(a-b) \oplus \overline{D}.(b-a)\]
\[= \overline{D}.((a-b) \oplus \overline{D}.(a \cap b))\]
\[\oplus \overline{D}.((b-a) \oplus \overline{D}.(b \cap a))\]
\[= \overline{D}.a \oplus \overline{D}.b\]

\[\overline{D}.a = \bigoplus_{i \in a} D.i\]
What can’t we do?

The encryption above turns out to be fundamental to the security correctness of a number of well-known protocols.

Let’s see what happens if we try to break the encryption: the importance of a security framework is what it prevents you from doing..

Why do we have to take the choice over the entire power set? Surely we can do better, and still retain security...
Informally... a “more efficient” version

The protocol for two independent servers $D_A, D_B$:

1. User computes in secret two shares for his request;

2. Each share is a subset selected at random (almost) over sets of requests of size no more than three;

3. The user sends one subset-share to $D_A$, and the other subset-share to $D_B$; each server looks up all requests in its share, computes the exclusive-or and sends the result back to the user;

4. The user takes the exclusive-or of the result.
The Encryption Lemma... not

\[
\begin{align*}
\text{[ vis}_A a; \text{vis}_B b; \quad (a \oplus b) &:= E ] & \quad \text{“from (1)”} \\
\text{= [ vis}_A a; \text{vis}_B b; \quad \langle\langle (a \oplus b) := E \rangle\rangle ] & \quad \text{“statement is atomic already”} \\
\text{= [ vis}_A a; \text{vis}_B b; \quad \langle\langle a \in \mathcal{E}; b := E \oplus a \rangle\rangle ] & \quad \text{“atomicity lemma”} \\
\text{= [ vis}_A a; \text{vis}_B b; \quad a \in \mathcal{E}; b := E \oplus a ] & \quad \text{“statements are atomic anyway”} \\
\text{= [ vis}_A a; a \in \mathcal{E}; \quad \text{[ vis}_B b; \quad b := E \oplus a ] & \quad \text{“b is not free in } \mathcal{E}; \text{ see (ii) below” } \heartsuit \\
\text{= [ vis}_A a; \quad a \in \mathcal{E}; \text{skip ] } & \quad \text{“b is hidden from } A \” b \\
\text{= [ vis}_A a; \quad a \in \mathcal{E] } & \quad \text{“a is a local visible”} \\
\text{= skip . }
\end{align*}
\]

The proof for B’s point of view is symmetric. The crucial features \heartsuit of the derivation are these:

(i) For all \{c\} and \(x_A\) of size 3 or less there must be some \(x_B\) of size 3 or less with \(x_B = \{c\} \triangle x_A\).

(i) For all \(E\) and \(a\) there must be some \(b\) with \(b = E \oplus a\).
\[ \begin{aligned} \text{vis}_U u, c \\ u := D \cdot c \end{aligned} \]

\[ \begin{aligned} &\text{vis}_U x_A, x_B : \mathbb{PN}; y_A, y_B : \text{Bool}; \\ &\text{vis}_A a : \mathbb{PN}, z_A : \text{Bool}; \text{vis}_B b : \mathbb{PN}, z_B : \text{Bool}; \\ &\begin{cases} (x_A \Delta x_B) := \{c\}; &\Leftarrow \text{Split } c \text{ into two “subset” shares.} \\ a, b := x_A, x_B; &\Leftarrow \text{Send to the servers separately.} \\ z_A := (\bigoplus_{i \in a} D_A.i); &\Leftarrow \text{Each computes the } \bigoplus \text{ of its shares.} \\ z_B := (\bigoplus_{i \in b} D_B.i); &\Leftarrow \cdots \\ y_A, y_B := z_A, z_B; &\Leftarrow \text{Each sends the result back to the requester.} \\ u := y_A \bigoplus y_B &\Leftarrow \text{The results are } \bigoplus \text{-ed together.} \end{cases} \end{aligned} \]
Conclusion: small extension; large opportunity

- It is a specialisation of classical refinement, more restrictive in order to preserve security properties.

- Those properties are non-interference in style.

- The semantics is compositional.

- It supports hierarchical refinement-based program development.

- It provides a link between source-code reasoning and the “mathematics underlying secrecy.”
\textbf{Conclusion: small extension; large opportunity}

\begin{itemize}
\item \textbf{vis}_U u, c
\item u := D.c
\end{itemize}

\begin{itemize}
\item \textbf{vis}_U x_A, x_B : \mathbb{PN}; y_A, y_B : \text{Bool};
\item \textbf{vis}_A a : \mathbb{PN}, z_A : \text{Bool}; \textbf{vis}_B b : \mathbb{PN}, z_B : \text{Bool};
\end{itemize}

\begin{itemize}
\item (x_A \Delta x_B) := \{c\}; \quad \Leftarrow \text{Split } c \text{ into two “subset” shares.}
\item a, b := x_A, x_B; \quad \Leftarrow \text{Send to the servers separately.}
\item z_A := (\bigoplus_{i \in a} D_A.i); \quad \Leftarrow \text{Each computes the } \bigoplus \text{ of its shares.}
\item z_B := (\bigoplus_{i \in b} D_B.i); \quad \Leftarrow \cdots
\item y_A, y_B := z_A, z_B; \quad \Leftarrow \text{Each sends the result back to the requester.}
\item u := y_A \oplus y_B \quad \Leftarrow \text{The results are } \oplus\text{-ed together.}
\end{itemize}

It is a specialisation of classical refinement, more restrictive in order to preserve security properties.
\[
\begin{align*}
\text{\textbf{vis}}_U & \ u, c \\
u := D \cdot c \\
\end{align*}
\]

\[
\begin{align*}
\sqsubseteq & \ [ \text{\textbf{vis}}_U & \ x_A, x_B : \mathbb{PN}; y_A, y_B : \text{Bool} ; \\
& \text{\textbf{vis}}_A & \ a : \mathbb{PN}, z_A : \text{Bool} ; \text{\textbf{vis}}_B & \ b : \mathbb{PN}, z_B : \text{Bool} ; \\
(x_A \Delta x_B) & := \{ c \} ; \quad \Leftarrow \text{Split } c \text{ into two “subset” shares.} \\
a, b & := x_A, x_B ; \quad \Leftarrow \text{Send to the servers separately.} \\
z_A & := (\oplus_{i \in a} D_A . i) ; \quad \Leftarrow \text{Each computes the } \oplus \text{ of its shares.} \\
z_B & := (\oplus_{i \in b} D_B . i) ; \quad \Leftarrow \cdots \\
y_A, y_B & := z_A, z_B ; \quad \Leftarrow \text{Each sends the result back to the requester.} \\
u & := y_A \oplus y_B \quad \Leftarrow \text{The results are } \oplus\text{-ed together.}
\end{align*}
\]
\[ \mathit{vis}_U u, c \]
\[ u := D.c \]
\[ \subseteq \]
\[ \begin{align*}
\mathit{vis}_U x_A, x_B : \mathbb{P}N; y_A, y_B : \mathbb{B}ool; \\
\mathit{vis}_A a : \mathbb{P}N, z_A : \mathbb{B}ool; \vspace{.15cm} \\
\mathit{vis}_B b : \mathbb{P}N, z_B : \mathbb{B}ool;
\end{align*} \]
\[ (x_A \Delta x_B) := \{ c \}; \quad \text{\textit{\(\Leftarrow\) Split } } c \text{ \textit{into two “subset” shares.}} \]
\[ a, b := x_A, x_B; \quad \text{\textit{\(\Leftarrow\) Send to the servers separately.}} \]
\[ z_A := (\oplus_{i \in a} D_A.i); \quad \text{\textit{\(\Leftarrow\) Each computes the } \oplus \text{ of its shares.}} \]
\[ z_B := (\oplus_{i \in b} D_B.i); \quad \cdots \]
\[ y_A, y_B := z_A, z_B; \quad \text{\textit{\(\Leftarrow\) Each sends the result back to the requester.}} \]
\[ u := y_A \oplus y_B \quad \text{\textit{\(\Leftarrow\) The results are } \oplus \text{-ed together.}} \]
Conclusion: small extension; large opportunity

\[\text{vis}_U u, c\]
\[u := D.c\]

\[\subseteq \begin{array}{l}
\text{vis}_U x_A, x_B : \mathbb{PN}; y_A, y_B : \mathbb{Bool}; \\
\text{vis}_A a : \mathbb{PN}, z_A : \mathbb{Bool}; \text{vis}_B b : \mathbb{PN}, z_B : \mathbb{Bool}; \\
(x_A \Delta x_B) := \{c\}; \quad \Leftrightarrow \text{Split } c \text{ into two “subset” shares}. \\
a, b := x_A, x_B; \quad \Leftrightarrow \text{Send to the servers separately}. \\
z_A := (\oplus_{i \in a} D_A.i); \quad \Leftrightarrow \text{Each computes the } \oplus \text{ of its shares}. \\
z_B := (\oplus_{i \in b} D_B.i); \quad \Leftrightarrow \cdots \\
y_A, y_B := z_A, z_B; \quad \Leftrightarrow \text{Each sends the result back to the requester}. \\
u := y_A \oplus y_B \quad \Leftrightarrow \text{The results are } \oplus\text{-ed together.}
\end{array}\]

It supports hierarchical refinement-based program development.
8.2 Collusion and visibility declarations

The above derivation explicitly separates the $U/A$ and $U/B$ correspondence by enforced by the visibility declarations $\text{vis}_A$ and $\text{vis}_B$; for Chor that separation is articulated by the "non-collusion" assumption, and theorems there depend upon it. Here there is a similar dependency, and indeed the validity of refinement depends upon it.

To investigate what would happen if $A$ and $B$ do collude, we rename all the $A/B$ variables to belong to a single server $C$ variable, and attempt the same derivation. This means that all $\text{vis}_A$ and $\text{vis}_B$ declarations become $\text{vis}_C$—then a careful review of the proofs shows that the original encryption §5, on which the whole security is built fails at the step labelled $\flat$. In this case, the relabelling would make both $a, b$ variables $\text{vis}_C$, so that the comment "b is hidden . . ." is invalid, preventing the replacement of the assignment to $b$ with skip.

8.3 Efficient perfect information retrieval

The solution presented in §8 actually does not reduce the overhead on the network at all—in fact it is the same as the single-server solution where the whole database must be sent to $U$. The full solution, combining privacy and a reduction in average network traﬃc—from $O_\infty N \times$ to $O_\infty \sqrt{N \times}$ for example—needs strictly more than two servers, and a structured addressing scheme. Again each server is sent an apparently random set of requests for which it must compute the $\oplus$ of the results, and return to the user, who can then reassemble to uncover the request. Although the addressing scheme is somewhat detailed, the principles for correctness, and the machinery for proof remain the same, namely generalised encryption §5.1 and the exclusive-or algebra §7.1.

We do this since we do not assume anything about the nature of the collusion, except that the servers are able to share all correspondence.

Conclusion: small extension; large opportunity

It provides a link between source-code reasoning and the “mathematics underlying secrecy.”