Computer-aided cryptographic proofs

Gilles Barthe

IMDEA Software Institute, Madrid, Spain

Challenges with provable security

Building or verifying reductionist proofs is hard

- Proofs are long and error-prone
- Rely on unstated and unverified invariants
- Intricate reasoning steps justified informally
- Verifying proofs is not much easier than building them

Abstraction gap

- Provable security reasons about algorithmic descriptions
- Standards constrain implementations
- Attackers target executable code

Computer-aided cryptography

aims to address both challenges using formal methods

Computer-aided cryptographic proofs: approach

Leverage existing approaches from formal methods

- program logics, VC generation, invariant generation
- SMT solvers, theorem provers, proof assistants
- verified compilers

The essence of our work

(code-based game-playing approach to) provable security
=
(relational) verification of probabilistic programs

EasyCrypt

- Interactive proof assistant for cryptographic proofs
- Back-end to SMT solvers
- libraries of common proof techniques (hybrid arguments, eager sampling, independent from adversary's view...)
- general tools: probabilistic (relational) Hoare logics

Case studies

- encryption, signatures, hash designs, key exchange protocols, zero knowledge protocols, garbled circuits...
- (computational) differential privacy
- mechanism design

Formal reasoning about cryptographic games

- Define programming language
- Give meaning to programs
- Reason about (relationships between) programs
- Reason directly about probabilities? Better not

Probabilistic couplings

- Perfect for *relational* verification of probabilistic programs
- Reasoning about probabilities implicit
- Implicit in (many/most) code-based game-playing proofs

Probabilistic couplings

Idea

- Put two probabilistic systems in the same space
- Coordinate samplings

Formal definition

- Let μ_1 and μ_2 be sub-distributions over A
- A sub-distribution μ over $A \times A$ is a coupling for (μ_1, μ_2) iff $\pi_1(\mu) = \mu_1$ and $\pi_2(\mu) = \mu_2$

Extends to distinct probabilistic spaces

Application: simple random walk on integers

- Start at position p_0
- Each step, flip coin $x \xleftarrow{\$}$ flip
- Heads: $p \leftarrow p + 1$
- Tails: $p \leftarrow p - 1$

Convergence

Asymptotically, output distribution independent of initial position

Coupling the walks to meet

Case $p_1 = p_2$: Walks have met

- Arrange samplings $x_1 = x_2$
- Continue to have $p_1 = p_2$

Case $p_1 \neq p_2$: Walks have not met

- Arrange samplings $x_1 = \neg x_2$
- Walks make mirror moves

If walks meet, they move together

Coupling the walks to meet

Case $p_1 = p_2$: Walks have met

- Arrange samplings $x_1 = x_2$
- Continue to have $p_1 = p_2$

Case $p_1 \neq p_2$: Walks have not met

- Arrange samplings $x_1 = \neg x_2$
- Walks make mirror moves

If walks meet, they move together

Coupling the walks to meet

Case $p_1 = p_2$: Walks have met

- Arrange samplings $x_1 = x_2$
- Continue to have $p_1 = p_2$

Case $p_1 \neq p_2$: Walks have not met

- Arrange samplings $x_1 = \neg x_2$
- Walks make mirror moves

If walks meet, they move together

Why is this interesting?

- Start two random walks at w and $w + 2k$
- To show: output distributions converge asymptotically
- Once walks meet, they stay equal
- Distance is at most probability walks don't meet
- Reasoning up to failure event
- Show failure event has low probability asymptotically

Next

Couplings for cryptography

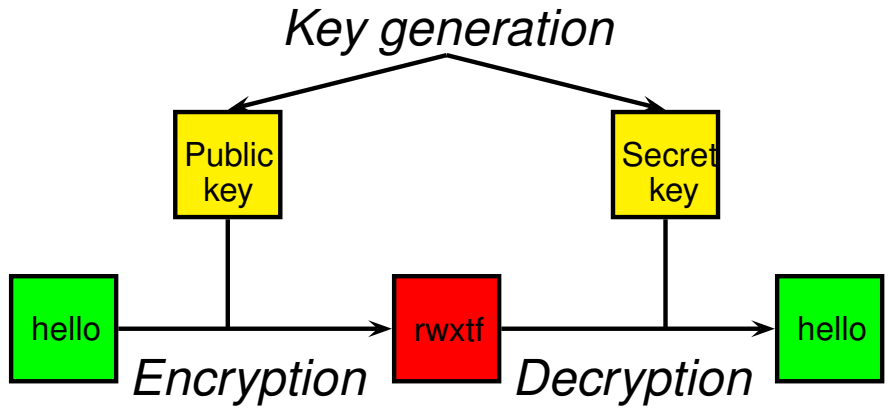
Public-key encryption

Algorithms $(\mathcal{K}, \mathcal{E}_{pk}, \mathcal{D}_{sk})$

- $\mathcal{E}$ probabilistic
- $\mathcal{D}$ deterministic and partial

If (sk, pk) is a valid key pair,

$$\mathcal{D}_{sk}(\mathcal{E}_{pk}(m)) = m$$



Indistinguishability

Game IND-CPA($\mathcal{A}$)
$(sk, pk) \leftarrow \mathcal{K}()$;
$(m_0, m_1) \leftarrow \mathcal{A}_1(pk)$;
$b \xleftarrow{\$} \{0,1\}$;
$c^\star \leftarrow \mathcal{E}_{pk}(m_b)$;
$b' \leftarrow \mathcal{A}_2(c^\star)$;
return $(b' = b)$

Indistinguishability

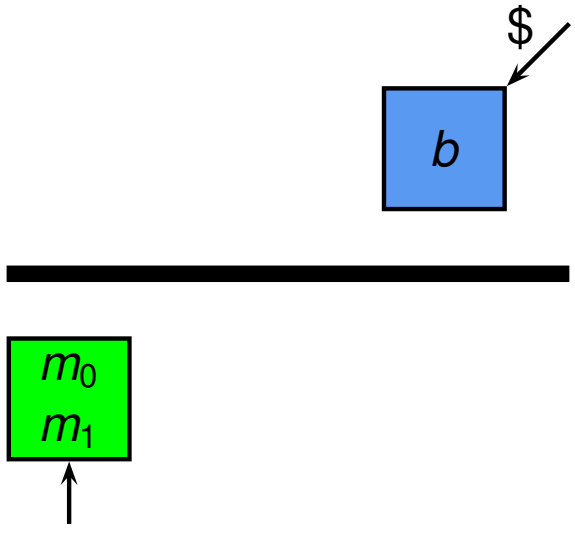
Game IND-CPA($\mathcal{A}$)
$(sk, pk) \leftarrow \mathcal{K}()$;
$(m_0, m_1) \leftarrow \mathcal{A}_1(pk)$;
$b \xleftarrow{\$} \{0,1\}$;
$c^\star \leftarrow \mathcal{E}_{pk}(m_b)$;
$b' \leftarrow \mathcal{A}_2(c^\star)$;
return $(b' = b)$

Indistinguishability

Game IND-CPA($\mathcal{A}$)
$(sk, pk) \leftarrow \mathcal{K}()$;
$(m_0, m_1) \leftarrow \mathcal{A}_1(pk)$;
$b \xleftarrow{\$} \{0,1\}$;
$c^\star \leftarrow \mathcal{E}_{pk}(m_b)$;
$b' \leftarrow \mathcal{A}_2(c^\star)$;
return $(b' = b)$

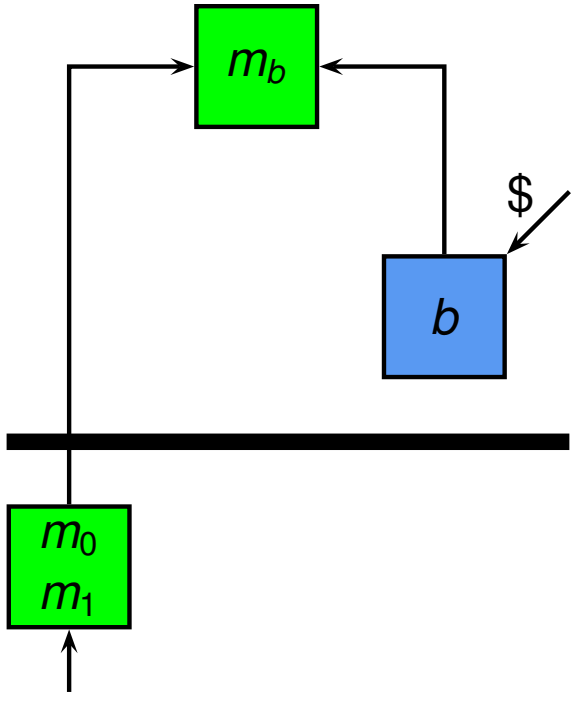
Indistinguishability

Game IND-CPA($\mathcal{A}$)
$(sk, pk) \leftarrow \mathcal{K}()$;
$(m_0, m_1) \leftarrow \mathcal{A}_1(pk)$;
$b \xleftarrow{\$} \{0,1\}$;
$c^\star \leftarrow \mathcal{E}_{pk}(m_b)$;
$b' \leftarrow \mathcal{A}_2(c^\star)$;
return $(b' = b)$



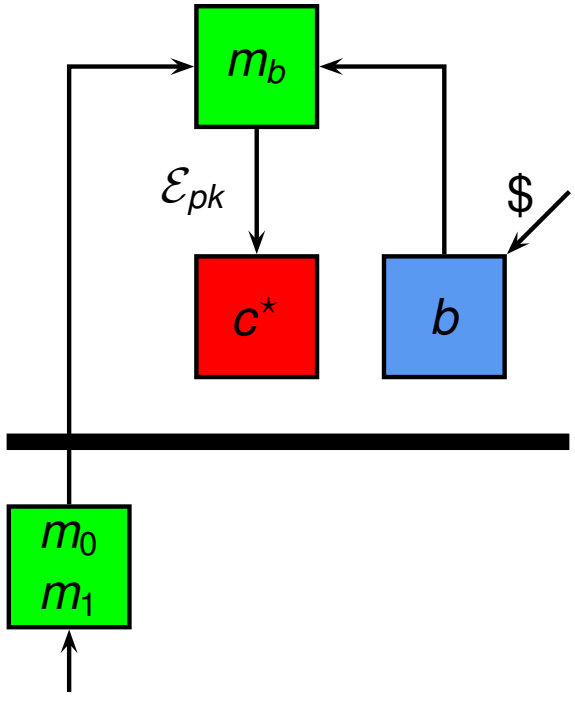
Indistinguishability

Game IND-CPA($\mathcal{A}$)
$(sk, pk) \leftarrow \mathcal{K}()$;
$(m_0, m_1) \leftarrow \mathcal{A}_1(pk)$;
$b \xleftarrow{\$} \{0,1\}$;
$c^\star \leftarrow \mathcal{E}_{pk}(m_b)$;
$b' \leftarrow \mathcal{A}_2(c^\star)$;
return $(b' = b)$



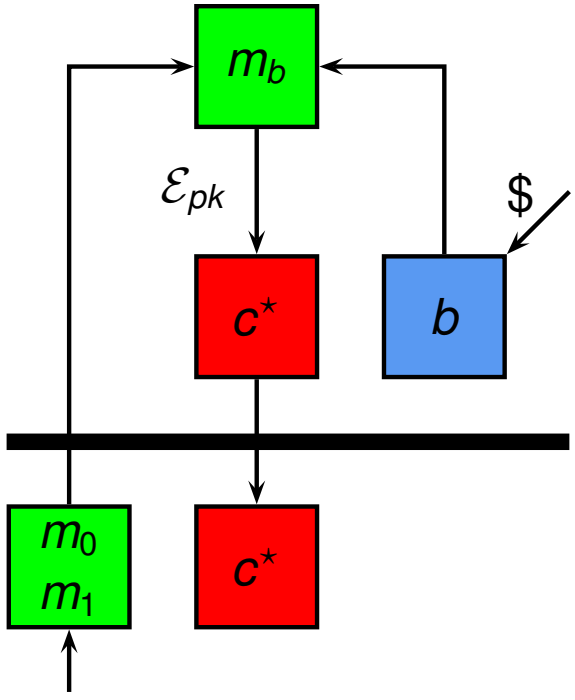
Indistinguishability

Game IND-CPA($\mathcal{A}$)
$(sk, pk) \leftarrow \mathcal{K}()$;
$(m_0, m_1) \leftarrow \mathcal{A}_1(pk)$;
$b \xleftarrow{\$} \{0,1\}$;
$c^\star \leftarrow \mathcal{E}_{pk}(m_b)$;
$b' \leftarrow \mathcal{A}_2(c^\star)$;
return $(b' = b)$



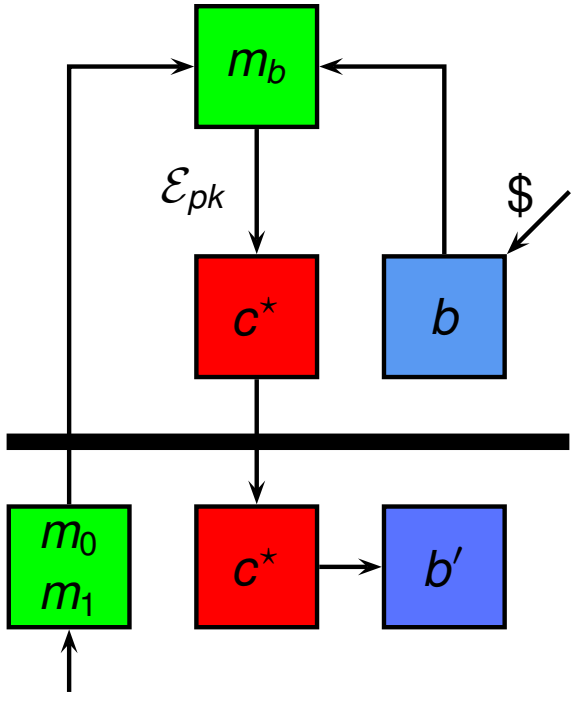
Indistinguishability

Game IND-CPA($\mathcal{A}$)
$(sk, pk) \leftarrow \mathcal{K}()$;
$(m_0, m_1) \leftarrow \mathcal{A}_1(pk)$;
$b \xleftarrow{\$} \{0,1\}$;
$c^\star \leftarrow \mathcal{E}_{pk}(m_b)$;
$b' \leftarrow \mathcal{A}_2(c^\star)$;
return $(b' = b)$



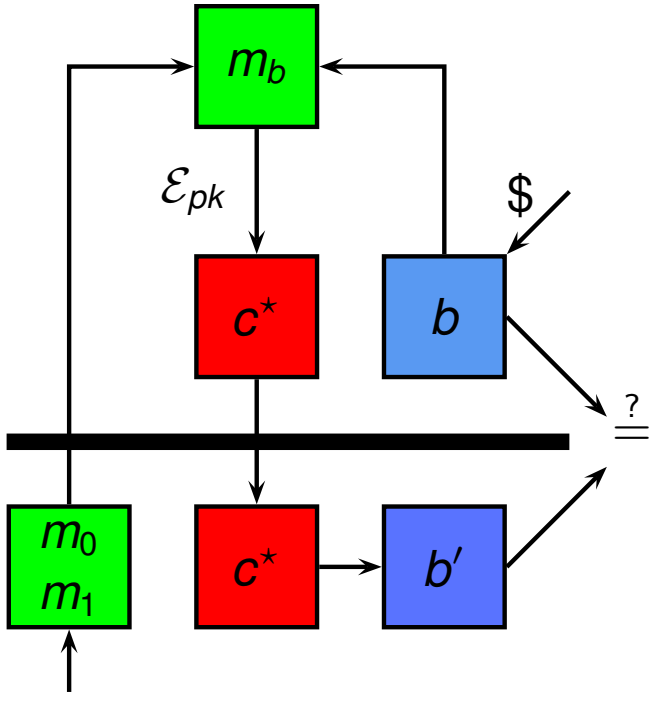
Indistinguishability

Game IND-CPA($\mathcal{A}$)
$(sk, pk) \leftarrow \mathcal{K}()$;
$(m_0, m_1) \leftarrow \mathcal{A}_1(pk)$;
$b \xleftarrow{\$} \{0,1\}$;
$c^\star \leftarrow \mathcal{E}_{pk}(m_b)$;
$b' \leftarrow \mathcal{A}_2(c^\star)$;
return $(b' = b)$



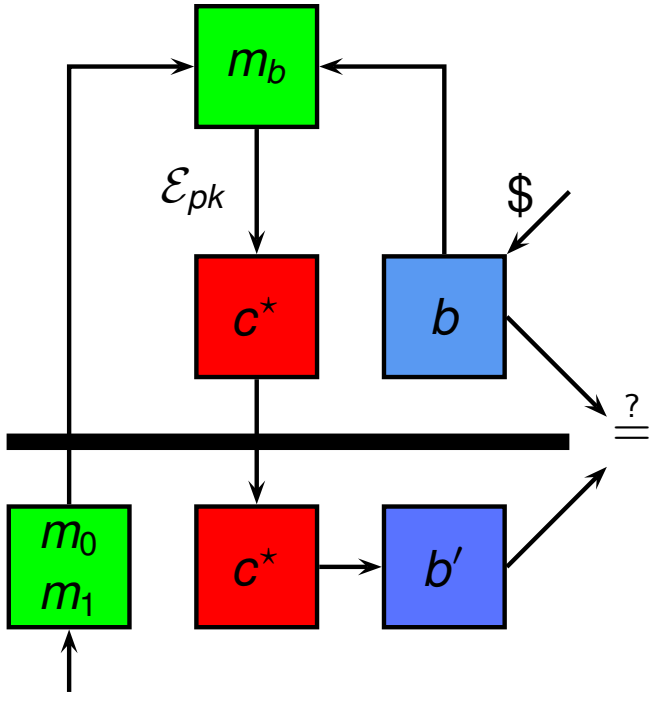
Indistinguishability

Game IND-CPA($\mathcal{A}$)
$(sk, pk) \leftarrow \mathcal{K}()$;
$(m_0, m_1) \leftarrow \mathcal{A}_1(pk)$;
$b \xleftarrow{\$} \{0,1\}$;
$c^\star \leftarrow \mathcal{E}_{pk}(m_b)$;
$b' \leftarrow \mathcal{A}_2(c^\star)$;
return $(b' = b)$



Indistinguishability

Game IND-CPA($\mathcal{A}$)
$(sk, pk) \leftarrow \mathcal{K}()$;
$(m_0, m_1) \leftarrow \mathcal{A}_1(pk)$;
$b \xleftarrow{\$} \{0,1\}$;
$c^\star \leftarrow \mathcal{E}_{pk}(m_b)$;
$b' \leftarrow \mathcal{A}_2(c^\star)$;
return $(b' = b)$



$$\left| \Pr_{\text{IND-CPA}(\mathcal{A})}\left[b' = b\right] - \frac{1}{2} \right| \quad \text{small}$$

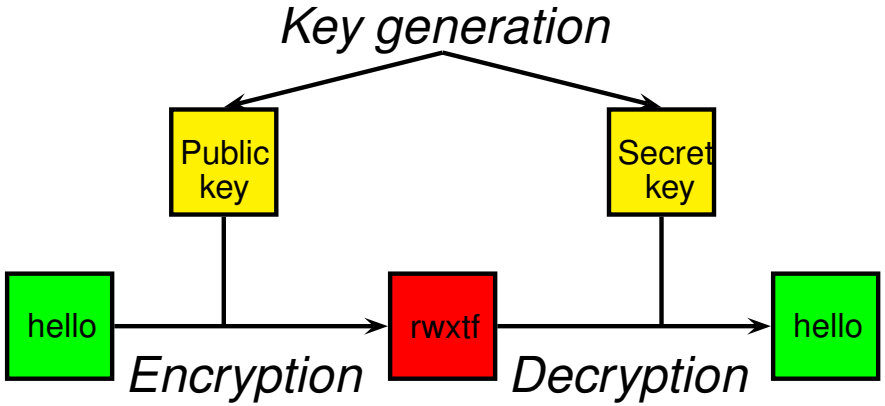
One-way trapdoor permutations

Algorithms $(\mathcal{K}, \mathsf{f}_{pk}, \mathsf{f}_{sk}^{-1})$

- f_{pk} and f_{sk}^{-1} deterministic

If (sk, pk) is a valid key pair,

$$\mathsf{f}_{sk}^{-1}(\mathsf{f}_{pk}(m)) = m$$

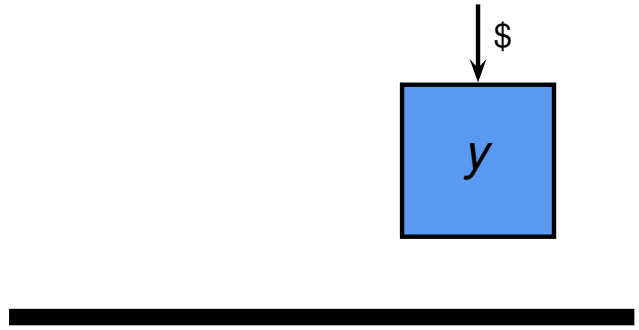


One-way trapdoor permutations

Game $\mathrm{OW}(\mathcal{I})$
$(sk, pk) \leftarrow \mathcal{K}()$;
$y \xleftarrow{\$} \{0,1\}^n$;
$x^\star \leftarrow \mathsf{f}_{pk}(y)$;
$y' \leftarrow \mathcal{I}(x^\star)$;
return $(y' = y)$

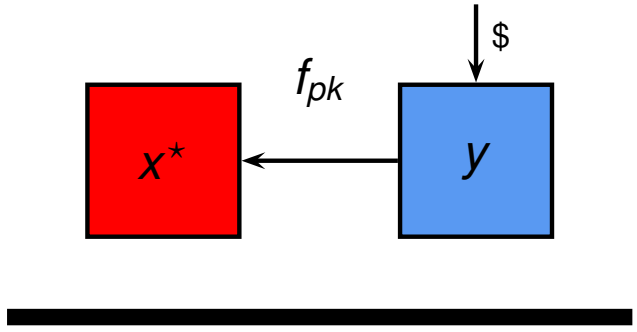
One-way trapdoor permutations

Game $\mathrm{OW}(\mathcal{I})$
$(sk, pk) \leftarrow \mathcal{K}()$;
$y \xleftarrow{\$} \{0,1\}^n$;
$x^\star \leftarrow \mathsf{f}_{pk}(y)$;
$y' \leftarrow \mathcal{I}(x^\star)$;
return $(y' = y)$



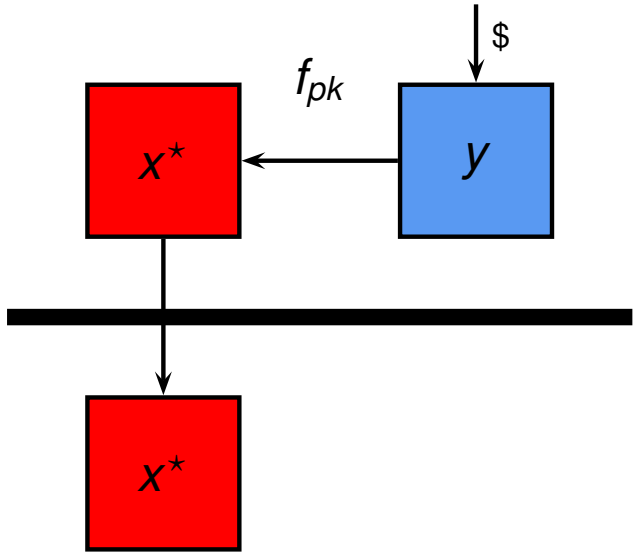
One-way trapdoor permutations

Game $\mathrm{OW}(\mathcal{I})$
$(sk, pk) \leftarrow \mathcal{K}();$
$y \xleftarrow{\$} \{0,1\}^n;$
$x^\star \leftarrow \mathsf{f}_{pk}(y);$
$y' \leftarrow \mathcal{I}(x^\star);$
return $(y' = y)$



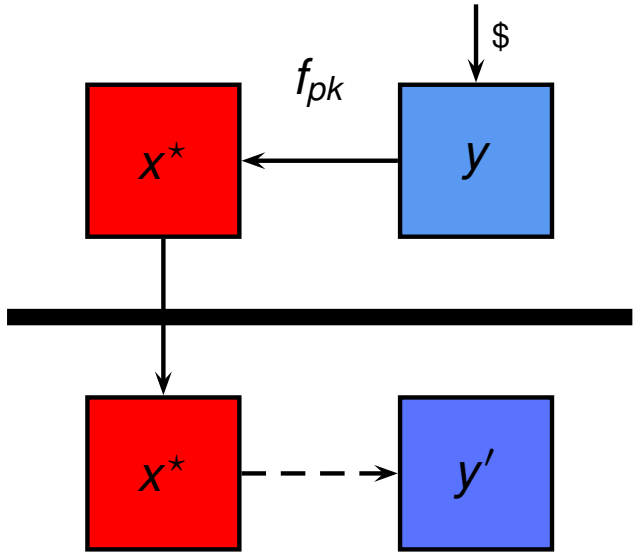
One-way trapdoor permutations

Game $\mathrm{OW}(\mathcal{I})$
$(sk, pk) \leftarrow \mathcal{K}();$
$y \xleftarrow{\$} \{0,1\}^n;$
$x^\star \leftarrow \mathsf{f}_{pk}(y);$
$y' \leftarrow \mathcal{I}(x^\star);$
return $(y' = y)$



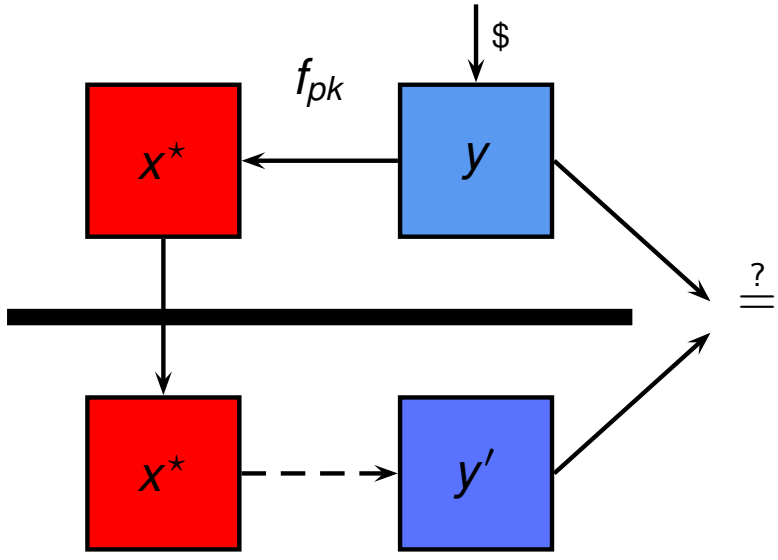
One-way trapdoor permutations

Game $\mathrm{OW}(\mathcal{I})$
$(sk, pk) \leftarrow \mathcal{K}();$
$y \xleftarrow{\$} \{0,1\}^n;$
$x^\star \leftarrow \mathsf{f}_{pk}(y);$
$y' \leftarrow \mathcal{I}(x^\star);$
return $(y' = y)$



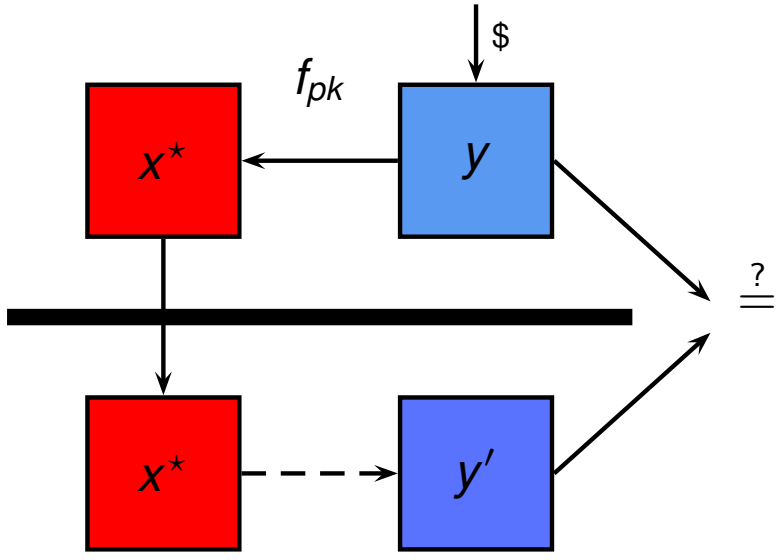
One-way trapdoor permutations

Game $\mathrm{OW}(\mathcal{I})$
$(sk, pk) \leftarrow \mathcal{K}();$
$y \xleftarrow{\$} \{0,1\}^n;$
$x^\star \leftarrow \mathsf{f}_{pk}(y);$
$y' \leftarrow \mathcal{I}(x^\star);$
return $(y' = y)$



One-way trapdoor permutations

Game $\mathrm{OW}(\mathcal{I})$
$(sk, pk) \leftarrow \mathcal{K}();$
$y \xleftarrow{\$} \{0,1\}^n;$
$x^\star \leftarrow \mathsf{f}_{pk}(y);$
$y' \leftarrow \mathcal{I}(x^\star);$
return $(y' = y)$



$$\Pr_{\mathrm{OW}(\mathcal{I})}\left[y' = y\right] \quad \text{small}$$

Random oracles

Oracle $H(x)$:
if $x \notin L$ then
  $r \xleftarrow{\$} \{0,1\}^k$;
  $L \leftarrow (x, r) :: L$;
return $L[x]$;

- Idealized model of hash function
- Modeled as stateful procedure

Example: padding-based encryption

Game IND-CPA($\mathcal{A}$) :
$(sk, pk) \leftarrow \mathcal{K}(\ )$;
$(m_0, m_1) \leftarrow \mathcal{A}_1(pk)$;
$b \xleftarrow{\$} \{0,1\}$;
$c^\star \leftarrow \mathcal{E}_{pk}(m_b)$;
$b' \leftarrow \mathcal{A}_2(c^\star)$;
return $(b' = b)$

Encryption $\mathcal{E}_{pk}(m)$:
$r \xleftarrow{\$} \{0,1\}^\ell$;
$s \leftarrow H(r) \oplus m$;
$y \leftarrow f_{pk}(r) \,\|\, s$;
return y

For every IND-CPA adversary $\mathcal{A}$, there exists an inverter $\mathcal{I}$ st

$$\left| \Pr_{\text{IND-CPA}(\mathcal{A})}\left[b' = b\right] - \frac{1}{2} \right| \leq \Pr_{\text{OW}(\mathcal{I})}\left[y' = y\right]$$

Proof

Game hopping technique

Game INDCPA :
$(sk, pk) \leftarrow \mathcal{K}();$
$(m_0, m_1) \leftarrow \mathcal{A}_1(pk);$
$b \xleftarrow{\$} \{0,1\};$
$c^\star \leftarrow \mathcal{E}_{pk}(m_b);$
$b' \leftarrow \mathcal{A}_2(c^\star);$
return $(b' = b)$
Encryption $\mathcal{E}_{pk}(m)$:
$r \xleftarrow{\$} \{0,1\}^\ell;$
$h \leftarrow H(r);$
$s \leftarrow h \oplus m;$
$c \leftarrow f_{pk}(r) \,\|\, s;$
return c

Game G :
$(sk, pk) \leftarrow \mathcal{K}();$
$(m_0, m_1) \leftarrow \mathcal{A}_1(pk);$
$b \xleftarrow{\$} \{0,1\};$
$c^\star \leftarrow \mathcal{E}_{pk}(m_b);$
$b' \leftarrow \mathcal{A}_2(c^\star);$
return $(b' = b)$
Encryption $\mathcal{E}_{pk}(m)$:
$r \xleftarrow{\$} \{0,1\}^\ell;$
$h \xleftarrow{\$} \{0,1\}^k;$
$s \leftarrow h \oplus m;$
$c \leftarrow f_{pk}(r) \,\|\, s;$
return c

Game G′ :
$(sk, pk) \leftarrow \mathcal{K}();$
$(m_0, m_1) \leftarrow \mathcal{A}_1(pk);$
$b \xleftarrow{\$} \{0,1\};$
$c^\star \leftarrow \mathcal{E}_{pk}(m_b);$
$b' \leftarrow \mathcal{A}_2(c^\star);$
return $(b' = b)$
Encryption $\mathcal{E}_{pk}(m)$:
$r \xleftarrow{\$} \{0,1\}^\ell;$
$s \xleftarrow{\$} \{0,1\}^k;$
$h \leftarrow s \oplus m;$
$c \leftarrow f_{pk}(r) \,\|\, s;$
return c

Game OW :
$(sk, pk) \leftarrow \mathcal{K}();$
$y \xleftarrow{\$} \{0,1\}^\ell;$
$y' \leftarrow \mathcal{I}(f_{pk}(y));$
return $y = y'$
Adversary $\mathcal{I}(x)$:
$(m_0, m_1) \leftarrow \mathcal{A}_1(pk);$
$s \xleftarrow{\$} \{0,1\}^k;$
$c^\star \leftarrow x \,\|\, s;$
$b' \leftarrow \mathcal{A}_2(c^\star);$
$y' \leftarrow [z \in L_H^{\mathcal{A}} \,|\, f_{pk}(z) = x];$
return y'

1. Prove a probability claim for each hop
2. Obtain security bound by combining claims
3. Check execution time of constructed adversary

Conditional equivalence

$\mathcal{E}_{pk}(m)$:
$r \xleftarrow{\$} \{0,1\}^{\ell}$;
$h \leftarrow H(r)$;
$s \leftarrow h \oplus m$;
$c \leftarrow f_{pk}(r) \,\|\, s$;
return c

$\mathcal{E}_{pk}(m)$:
$r \xleftarrow{\$} \{0,1\}^{\ell}$;
$h \xleftarrow{\$} \{0,1\}^{k}$;
$s \leftarrow h \oplus m$;
$c \leftarrow f_{pk}(r) \,\|\, s$;
return c

By the Fundamental Lemma

$$\left|\Pr_{\text{IND-CPA}}\left[b' = b\right] - \Pr_{\mathbf{G}}\left[b' = b\right]\right| \leq \Pr_{\mathbf{G}}\left[r \in L_H^{\mathcal{A}}\right]$$

Equivalence

$\mathcal{E}_{pk}(m)$:
$r \xleftarrow{\$} \{0,1\}^{\ell}$;
$h \xleftarrow{\$} \{0,1\}^{k}$;
$s \leftarrow h \oplus m$;
$c \leftarrow f_{pk}(r) \| s$;
return c

$\mathcal{E}_{pk}(m)$:
$r \xleftarrow{\$} \{0,1\}^{\ell}$;
$s \xleftarrow{\$} \{0,1\}^{k}$;
$h \leftarrow s \oplus m$;
$c \leftarrow f_{pk}(r) \| s$;
return c

By optimistic sampling

$$\Pr_{\mathbf{G}}\left[r \in L_H^{\mathcal{A}}\right] = \Pr_{\mathbf{G}'}\left[r \in L_H^{\mathcal{A}}\right] \qquad \Pr_{\mathbf{G}}[b' = b] = \Pr_{\mathbf{G}'}[b' = b] = \tfrac{1}{2}$$

Equivalence

$\mathcal{E}_{pk}(m)$:
$r \xleftarrow{\$} \{0,1\}^{\ell}$;
$h \xleftarrow{\$} \{0,1\}^{k}$;
$s \leftarrow h \oplus m$;
$c \leftarrow f_{pk}(r) \| s$;
return c

$\mathcal{E}_{pk}(m)$:
$r \xleftarrow{\$} \{0,1\}^{\ell}$;
$s \xleftarrow{\$} \{0,1\}^{k}$;
$h \leftarrow s \oplus m$;
$c \leftarrow f_{pk}(r) \| s$;
return c

By optimistic sampling

$$\left|\Pr_{\text{IND-CPA}}[b' = b] - \tfrac{1}{2}\right| \leq \Pr_{\mathbf{G}'}\left[r \in L_H^{\mathcal{A}}\right]$$

Reduction

Game INDCPA :
$(sk, pk) \leftarrow \mathcal{K}()$;
$(m_0, m_1) \leftarrow \mathcal{A}_1(pk)$;
$b \xleftarrow{\$} \{0,1\}$;
$c^\star \leftarrow \mathcal{E}_{pk}(m_b)$;
$b' \leftarrow \mathcal{A}_2(c^\star)$;
return $(b' = b)$
Encryption $\mathcal{E}_{pk}(m)$:
$r \xleftarrow{\$} \{0,1\}^\ell$;
$s \xleftarrow{\$} \{0,1\}^k$;
$c \leftarrow f_{pk}(r) \,\|\, s$;
return c

Game OW :
$(sk, pk) \leftarrow \mathcal{K}()$;
$y \xleftarrow{\$} \{0,1\}^\ell$;
$y' \leftarrow \mathcal{I}(f_{pk}(y))$;
return $y = y'$
Adversary $\mathcal{I}(x)$:
$(m_0, m_1) \leftarrow \mathcal{A}_1(pk)$;
$b \xleftarrow{\$} \{0,1\}$;
$s \xleftarrow{\$} \{0,1\}^k$;
$c^\star \leftarrow x \,\|\, s$;
$b' \leftarrow \mathcal{A}_2(c^\star)$;
$y' \leftarrow [z \in L_H^{\mathcal{A}} \mid f_{pk}(z) = x]$;
return y'

$$\Pr_{\mathbf{G}'}\left[r \in L_H^{\mathcal{A}}\right] \leq \Pr_{\mathrm{OW}(\mathcal{I})}[y' = y]$$

Reduction

Game INDCPA :
$(sk, pk) \leftarrow \mathcal{K}()$;
$(m_0, m_1) \leftarrow \mathcal{A}_1(pk)$;
$b \xleftarrow{\$} \{0,1\}$;
$c^\star \leftarrow \mathcal{E}_{pk}(m_b)$;
$b' \leftarrow \mathcal{A}_2(c^\star)$;
return $(b' = b)$
Encryption $\mathcal{E}_{pk}(m)$:
$r \xleftarrow{\$} \{0,1\}^\ell$;
$s \xleftarrow{\$} \{0,1\}^k$;
$c \leftarrow f_{pk}(r) \,\|\, s$;
return c

Game OW :
$(sk, pk) \leftarrow \mathcal{K}()$;
$y \xleftarrow{\$} \{0,1\}^\ell$;
$y' \leftarrow \mathcal{I}(f_{pk}(y))$;
return $y = y'$
Adversary $\mathcal{I}(x)$:
$(m_0, m_1) \leftarrow \mathcal{A}_1(pk)$;
$b \xleftarrow{\$} \{0,1\}$;
$s \xleftarrow{\$} \{0,1\}^k$;
$c^\star \leftarrow x \,\|\, s$;
$b' \leftarrow \mathcal{A}_2(c^\star)$;
$y' \leftarrow [z \in L_H^{\mathcal{A}} \mid f_{pk}(z) = x]$;
return y'

$$\left|\Pr_{\text{IND-CPA}(\mathcal{A})}[b' = b] - \tfrac{1}{2}\right| \leq \Pr_{\text{OW}(\mathcal{I})}[y' = y]$$

Code-based approach to probabilistic liftings

$$
\begin{array}{rcll}
\mathcal{C} & ::= & \text{skip} & \text{skip} \\
& | & \mathcal{V} \leftarrow \mathcal{E} & \text{assignment} \\
& | & \mathcal{V} \xleftarrow{\$} \mathcal{D} & \text{random sampling} \\
& | & \mathcal{C};\ \mathcal{C} & \text{sequence} \\
& | & \text{if } \mathcal{E} \text{ then } \mathcal{C} \text{ else } \mathcal{C} & \text{conditional} \\
& | & \text{while } \mathcal{E} \text{ do } \mathcal{C} & \text{while loop} \\
& | & \mathcal{V} \leftarrow \mathcal{P}(\mathcal{E}, \ldots, \mathcal{E}) & \text{procedure call}
\end{array}
$$

- $\mathcal{E}$: (higher-order) expressions
- $\mathcal{D}$: discrete sub-distributions
- $\mathcal{P}$: procedures

} user extensible

. oracles: concrete procedures

. adversaries: constrained abstract procedures

pRHL: probabilistic relational Hoare logic

- Judgment

$$\models \{P\}\ c_1 \sim c_2\ \{Q\}$$

  where P and Q denote relations on memories
- Validity

$$\forall m_1, m_2.\ (m_1, m_2) \models P \implies (\llbracket c_1 \rrbracket\ m_1, \llbracket c_2 \rrbracket\ m_2) \models Q^\sharp$$

- Definition of $\cdot^\sharp$ internalizes existence of coupling

Lifting

- Let R be a binary relation on A and B, i.e. $R \subseteq A \times B$
- Let μ_1 and μ_2 be sub-distributions over A and B
- $\mu_1 R^\# \mu_2$ iff there exists a coupling μ s.t. $\Pr_{y \leftarrow \mu}[y \notin R] = 0$

Lifting: applications

Assume $\mu_1 \; R^{\#} \; \mu_2$.

- Fund. lemma of lifting: If $R(x, y) \triangleq F(x) \Rightarrow G(y)$, then

$$\Pr_{x \leftarrow \mu_2}[G] \leq \Pr_{y \leftarrow \mu_1}[F]$$

- If $R(x, y) \triangleq x = y$, then

$$\Pr_{z \leftarrow \mu_1}[X] = \Pr_{z \leftarrow \mu_2}[X]$$

- If $R(x, y) \triangleq (F(x) \Rightarrow x = y) \wedge (F(x) \Leftrightarrow F(y))$, then

$$|\Pr_{z \leftarrow \mu_1}[X] - \Pr_{z \leftarrow \mu_2}[X]| \leq \max(\Pr_{z \leftarrow \mu_1}[\neg F], \Pr_{z \leftarrow \mu_2}[\neg F])$$

where X is any event.

Random assignment

$$\frac{h \text{ is 1-1 and } \forall a,\ \mu(a) = \mu'(h(a))}{\vDash \{\top\}\ x \xleftarrow{\$} \mu \sim x' \xleftarrow{\$} \mu'\ \{h(x\langle 1\rangle) = x'\langle 2\rangle\}}$$

$$\frac{\mu \Phi^{\#} \mu'}{\vDash \{\top\}\ x \xleftarrow{\$} \mu \sim x' \xleftarrow{\$} \mu'\ \{\Phi\}}$$

- First rule captures a special case of lifting where the post-condition is the graph of a bijection
- Second rule is more complete but seldom used

Conditionals and loops

Conditionals

$$\frac{P \Rightarrow e\langle 1\rangle = e'\langle 2\rangle \qquad \vDash \{P \wedge e\langle 1\rangle\}\ c_1 \sim c_1'\ \{Q\} \qquad \vDash \{P \wedge \neg e\langle 1\rangle\}\ c_2 \sim c_2'\ \{Q\}}{\vDash \{P\}\ \text{if } e \text{ then } c_1 \text{ else } c_2 \sim \text{if } e' \text{ then } c_1' \text{ else } c_2'\ \{Q\}}$$

$$\frac{\vDash \{P \wedge e\langle 1\rangle\}\ c_1 \sim c\ \{Q\} \qquad \vDash \{P \wedge \neg e\langle 1\rangle\}\ c_2 \sim c\ \{Q\}}{\vDash \{P\}\ \text{if } e \text{ then } c_1 \text{ else } c_2 \sim c\ \{Q\}}$$

Loops

$$\frac{P \Rightarrow e\langle 1\rangle = e'\langle 2\rangle \qquad \vDash \{P \wedge e\langle 1\rangle\}\ c \sim c'\ \{P\}}{\vDash \{P\}\ \text{while } e \text{ do } c \sim \text{while } e' \text{ do } c'\ \{P \wedge \neg e\langle 1\rangle\}}$$

$$\frac{\vDash \{P_1\}\ c \sim c'\ \{P\} \quad \vDash \{P_2\}\ c \sim \text{skip}\ \{P\} \quad \vDash \{P_3\}\ \text{skip} \sim c'\ \{P\}}{\vDash \{P\}\ \text{while } e \text{ do } c \sim \text{while } e' \text{ do } c'\ \{P \wedge \neg e\langle 1\rangle\}}$$

Adversaries

$$\frac{\forall \mathcal{O}.\ \vDash \{Q \wedge =_W\}\ z \leftarrow \mathcal{O}(\vec{w})\ \sim\ z \leftarrow \mathcal{O}(\vec{w})\ \left\{Q \wedge =_{\{z\}}\right\}}{\vDash \{Q \wedge =_Y\}\ x \leftarrow \mathcal{A}(\vec{y})\ \sim\ x \leftarrow \mathcal{A}(\vec{y})\ \left\{Q \wedge =_{\{x\}}\right\}}$$

- Adversaries perform arbitrary sequences of oracle calls (and intermediate computations)
- No functional specification
- Given the same inputs, provide the same outputs

Example: padding-based encryption

Game IND-CPA($\mathcal{A}$) :
$(sk, pk) \leftarrow \mathcal{K}(\ )$;
$(m_0, m_1) \leftarrow \mathcal{A}_1(pk)$;
$b \xleftarrow{\$} \{0,1\}$;
$c^\star \leftarrow \mathcal{E}_{pk}(m_b)$;
$b' \leftarrow \mathcal{A}_2(c^\star)$;
return $(b' = b)$

Encryption $\mathcal{E}_{pk}(m)$:
$r \xleftarrow{\$} \{0,1\}^\ell$;
$s \leftarrow H(r) \oplus m$;
$y \leftarrow f_{pk}(r) \,\|\, s$;
return y

For every IND-CPA adversary $\mathcal{A}$, there exists an inverter $\mathcal{I}$ st

$$\left| \Pr_{\text{IND-CPA}(\mathcal{A})}\left[b' = b\right] - \frac{1}{2} \right| \leq \Pr_{\text{OW}(\mathcal{I})}\left[y' = y\right]$$

Proof

Game hopping technique

Game INDCPA :
$(sk, pk) \leftarrow \mathcal{K}();$
$(m_0, m_1) \leftarrow \mathcal{A}_1(pk);$
$b \xleftarrow{\$} \{0,1\};$
$c^\star \leftarrow \mathcal{E}_{pk}(m_b);$
$b' \leftarrow \mathcal{A}_2(c^\star);$
return $(b' = b)$
Encryption $\mathcal{E}_{pk}(m)$:
$r \xleftarrow{\$} \{0,1\}^\ell;$
$h \leftarrow H(r);$
$s \leftarrow h \oplus m;$
$c \leftarrow f_{pk}(r) \,\|\, s;$
return c

Game G :
$(sk, pk) \leftarrow \mathcal{K}();$
$(m_0, m_1) \leftarrow \mathcal{A}_1(pk);$
$b \xleftarrow{\$} \{0,1\};$
$c^\star \leftarrow \mathcal{E}_{pk}(m_b);$
$b' \leftarrow \mathcal{A}_2(c^\star);$
return $(b' = b)$
Encryption $\mathcal{E}_{pk}(m)$:
$r \xleftarrow{\$} \{0,1\}^\ell;$
$h \xleftarrow{\$} \{0,1\}^k;$
$s \leftarrow h \oplus m;$
$c \leftarrow f_{pk}(r) \,\|\, s;$
return c

Game G′ :
$(sk, pk) \leftarrow \mathcal{K}();$
$(m_0, m_1) \leftarrow \mathcal{A}_1(pk);$
$b \xleftarrow{\$} \{0,1\};$
$c^\star \leftarrow \mathcal{E}_{pk}(m_b);$
$b' \leftarrow \mathcal{A}_2(c^\star);$
return $(b' = b)$
Encryption $\mathcal{E}_{pk}(m)$:
$r \xleftarrow{\$} \{0,1\}^\ell;$
$s \xleftarrow{\$} \{0,1\}^k;$
$h \leftarrow s \oplus m;$
$c \leftarrow f_{pk}(r) \,\|\, s;$
return c

Game OW :
$(sk, pk) \leftarrow \mathcal{K}();$
$y \xleftarrow{\$} \{0,1\}^\ell;$
$y' \leftarrow \mathcal{I}(f_{pk}(y));$
return $y = y'$
Adversary $\mathcal{I}(x)$:
$(m_0, m_1) \leftarrow \mathcal{A}_1(pk);$
$s \xleftarrow{\$} \{0,1\}^k;$
$c^\star \leftarrow x \,\|\, s;$
$b' \leftarrow \mathcal{A}_2(c^\star);$
$y' \leftarrow [z \in L_H^{\mathcal{A}} \mid f_{pk}(z) = x];$
return y'

1. For each hop
   - prove validity of pRHL judgment
   - derive probability claims
   - (possibly) resolve some probability expressions using pHL
2. Obtain security bound by combining claims
3. Check execution time of constructed adversary

Conditional equivalence

$\mathcal{E}_{pk}(m)$:
$r \xleftarrow{\$} \{0,1\}^{\ell}$;
$h \leftarrow H(r)$;
$s \leftarrow h \oplus m$;
$c \leftarrow f_{pk}(r) \,\|\, s$;
return c

$\mathcal{E}_{pk}(m)$:
$r \xleftarrow{\$} \{0,1\}^{\ell}$;
$h \xleftarrow{\$} \{0,1\}^{k}$;
$s \leftarrow h \oplus m$;
$c \leftarrow f_{pk}(r) \,\|\, s$;
return c

$$\models \{\top\} \text{ IND-CPA } \sim \mathbf{G} \left\{ (\neg r \in L_H^{\mathcal{A}})\langle 2 \rangle \rightarrow =_{b,b'} \right\}$$

$$\left| \Pr_{\text{IND-CPA}} \left[b' = b \right] - \Pr_{\mathbf{G}} \left[b' = b \right] \right| \leq \Pr_{\mathbf{G}} \left[r \in L_H^{\mathcal{A}} \right]$$

Equivalence

$\mathcal{E}_{pk}(m)$:
$r \xleftarrow{\$} \{0,1\}^{\ell}$;
$h \xleftarrow{\$} \{0,1\}^{k}$;
$s \leftarrow h \oplus m$;
$c \leftarrow f_{pk}(r) \| s$;
return c

$\mathcal{E}_{pk}(m)$:
$r \xleftarrow{\$} \{0,1\}^{\ell}$;
$s \xleftarrow{\$} \{0,1\}^{k}$;
$h \leftarrow s \oplus m$;
$c \leftarrow f_{pk}(r) \| s$;
return c

$$\models \{\top\}\ \mathbf{G}\ \sim\ \mathbf{G}'\ \left\{=_{b,b',r,L_H^{\mathcal{A}}}\right\}$$

$$\Pr_{\mathbf{G}}\left[r \in L_H^{\mathcal{A}}\right] = \Pr_{\mathbf{G}'}\left[r \in L_H^{\mathcal{A}}\right] \qquad \Pr_{\mathbf{G}}[b' = b] = \Pr_{\mathbf{G}'}[b' = b] = \frac{1}{2}$$

Equivalence

$\mathcal{E}_{pk}(m)$:
$r \xleftarrow{\$} \{0,1\}^{\ell}$;
$h \xleftarrow{\$} \{0,1\}^{k}$;
$s \leftarrow h \oplus m$;
$c \leftarrow f_{pk}(r) \| s$;
return c

$\mathcal{E}_{pk}(m)$:
$r \xleftarrow{\$} \{0,1\}^{\ell}$;
$s \xleftarrow{\$} \{0,1\}^{k}$;
$h \leftarrow s \oplus m$;
$c \leftarrow f_{pk}(r) \| s$;
return c

$$\models \{\top\}\ \mathbf{G}\ \sim\ \mathbf{G}'\ \left\{=_{b,b',r,L_H^{\mathcal{A}}}\right\}$$

$$\left|\Pr_{\text{IND-CPA}}[b' = b] - \tfrac{1}{2}\right| \leq \Pr_{\mathbf{G}'}\left[r \in L_H^{\mathcal{A}}\right]$$

Reduction

Game INDCPA :
$(sk, pk) \leftarrow \mathcal{K}();$
$(m_0, m_1) \leftarrow \mathcal{A}_1(pk);$
$b \xleftarrow{\$} \{0,1\};$
$c^\star \leftarrow \mathcal{E}_{pk}(m_b);$
$b' \leftarrow \mathcal{A}_2(c^\star);$
return $(b' = b)$
Encryption $\mathcal{E}_{pk}(m)$:
$r \xleftarrow{\$} \{0,1\}^\ell;$
$s \xleftarrow{\$} \{0,1\}^k;$
$c \leftarrow f_{pk}(r) \,\|\, s;$
return c

Game OW :
$(sk, pk) \leftarrow \mathcal{K}();$
$y \xleftarrow{\$} \{0,1\}^\ell;$
$y' \leftarrow \mathcal{I}(f_{pk}(y));$
return $y = y'$
Adversary $\mathcal{I}(x)$:
$(m_0, m_1) \leftarrow \mathcal{A}_1(pk);$
$b \xleftarrow{\$} \{0,1\};$
$s \xleftarrow{\$} \{0,1\}^k;$
$c^\star \leftarrow x \,\|\, s;$
$b' \leftarrow \mathcal{A}_2(c^\star);$
$y' \leftarrow [z \in L_H^{\mathcal{A}} \mid f_{pk}(z) = x];$
return y'

$$\models \{\top\}\ \mathbf{G}'\ \sim\ \mathrm{OW}\ \left\{(r \in L_H^{\mathcal{A}})\langle 1\rangle \rightarrow (y' = y)\langle 2\rangle\right\}$$

$$\Pr_{\mathbf{G}'}\left[r \in L_H^{\mathcal{A}}\right] \leq \Pr_{\mathrm{OW}(\mathcal{I})}[y' = y]$$

Reduction

Game INDCPA :
$(sk, pk) \leftarrow \mathcal{K}();$
$(m_0, m_1) \leftarrow \mathcal{A}_1(pk);$
$b \xleftarrow{\$} \{0,1\};$
$c^\star \leftarrow \mathcal{E}_{pk}(m_b);$
$b' \leftarrow \mathcal{A}_2(c^\star);$
return $(b' = b)$

Encryption $\mathcal{E}_{pk}(m)$:
$r \xleftarrow{\$} \{0,1\}^\ell;$
$s \xleftarrow{\$} \{0,1\}^k;$
$c \leftarrow f_{pk}(r) \,\|\, s;$
return c

Game OW :
$(sk, pk) \leftarrow \mathcal{K}();$
$y \xleftarrow{\$} \{0,1\}^\ell;$
$y' \leftarrow \mathcal{I}(f_{pk}(y));$
return $y = y'$

Adversary $\mathcal{I}(x)$:
$(m_0, m_1) \leftarrow \mathcal{A}_1(pk);$
$b \xleftarrow{\$} \{0,1\};$
$s \xleftarrow{\$} \{0,1\}^k;$
$c^\star \leftarrow x \,\|\, s;$
$b' \leftarrow \mathcal{A}_2(c^\star);$
$y' \leftarrow [z \in L_H^{\mathcal{A}} \mid f_{pk}(z) = x];$
return y'

$$\models \{\top\}\ \mathbf{G}'\ \sim\ \mathrm{OW}\ \left\{(r \in L_H^{\mathcal{A}})\langle 1\rangle \rightarrow (y' = y)\langle 2\rangle\right\}$$

$$\left|\Pr_{\text{IND-CPA}(\mathcal{A})}[b' = b] - \tfrac{1}{2}\right| \leq \Pr_{\text{OW}(\mathcal{I})}[y' = y]$$

Optimal Asymmetric Encryption Padding

Encryption $\mathcal{E}_{\mathrm{OAEP}(pk)}(m)$:
$r \xleftarrow{\$} \{0,1\}^{k_0}$;
$s \leftarrow G(r) \oplus (m \| 0^{k_1})$;
$t \leftarrow H(s) \oplus r$;
return $\mathsf{f}_{pk}(s \| t)$

Decryption $\mathcal{D}_{\mathrm{OAEP}(sk)}(c)$:
$(s,t) \leftarrow \mathsf{f}_{sk}^{-1}(c)$;
$r \leftarrow t \oplus H(s)$;
if $([s \oplus G(r)]_{k_1} = 0^{k_1})$
then $\{m \leftarrow [s \oplus G(r)]^{k}; \}$
else $\{m \leftarrow \perp; \}$
return m

OAEP: provable security

Game IND-CCA($\mathcal{A}$)
$(sk, pk) \leftarrow \mathcal{K}()$;
$(m_0, m_1) \leftarrow \mathcal{A}_1(pk)$;
$b \xleftarrow{\$} \{0,1\}$;
$c^\star \leftarrow \mathcal{E}_{pk}(m_b)$;
$b' \leftarrow \mathcal{A}_2(c^\star)$;
return $(b' = b)$

Game SPDOW($\mathcal{I}$)
$(sk, pk) \leftarrow \mathcal{K}()$;
$y \xleftarrow{\$} \{0,1\}^{k_2}$; $z \xleftarrow{\$} \{0,1\}^{k_3}$;
$x^\star \leftarrow \mathsf{f}_{pk}(y \,\|\, z)$;
$Y' \leftarrow \mathcal{I}(x^\star)$;
return $(y \in Y')$

FOR ALL IND-CCA adversary $\mathcal{A}$ against $(\mathcal{K}, \mathcal{E}_{\mathrm{OAEP}}, \mathcal{D}_{\mathrm{OAEP}})$,
THERE EXISTS a SPDOW adversary $\mathcal{I}$ against $(\mathcal{K}, \mathsf{f}, \mathsf{f}^{-1})$ st

$$\left|\Pr_{\text{IND-CCA}(\mathcal{A})}[b' = b] - \tfrac{1}{2}\right| \leq$$

$$\Pr_{\text{SPDOW}(\mathcal{I})}[y \in Y'] + \frac{3q_D q_G + q_D^2 + 4q_D + q_G}{2^{k_0}} + \frac{2q_D}{2^{k_1}}$$

and

$$t_{\mathcal{I}} \leq t_{\mathcal{A}} + q_D\, q_G\, q_H\, T_{\mathsf{f}}$$

Automated synthesis of padding-based encryption

Goals:

- Capture the essence of cryptographic proofs
- Minimize time and expertise for verification
- Explore design space of schemes

Approach:

- Isolate high-level proof principles
- Automate proofs
- Synthesize and analyze candidate schemes

Evaluation

- Generated and analyzed over 1,000,000 schemes
- Nearly complete coverage for CPA, high coverage for CCA
- Around 11% of schemes are IND-CPA

ZAEP: $f(r \parallel m \oplus G(r))$

For every INDCCA adversary $\mathcal{A}$ there exists an inverter $\mathcal{I}$ st

$$\left| \Pr_{\text{IND-CCA}}[b = b'] - \frac{1}{2} \right| \leq \mathbf{Succ}_f^{\text{OW}}(\mathcal{I}) + \frac{q_D}{2^n}$$

assuming existence of two efficient algorithms

CPA logic for padding-based encryption

Judgment:

$$c :_p \varphi$$

where φ is an event and p is an "asymptotic" probability:

- $\varphi = \mathsf{Ask}(H, e)$ and $p = 0$
- $\varphi = \mathsf{Guess}$ and $p = \frac{1}{2}$

Combines computational and symbolic cryptography

- Proof rules are computational (high-level principles)
- Side-conditions are discharged by symbolic methods

Main proof principles

Failure event	Replace $H(e)$ by fresh r
Optimistic sampling	Replace $e \oplus r$, where r is fresh, by r
Probability	Compute probability of $b = b'$ or $e \in L$
Reduction	Find inverter and apply one-wayness

Syntax of expressions

$$
\begin{array}{rcll}
\mathcal{E} & ::= & m & \text{input message} \\
 & | & 0 & \text{zero bitstring} \\
 & | & \mathcal{R} & \text{uniform random bitstring} \\
 & | & \mathcal{E} \oplus \mathcal{E} & \text{xor} \\
 & | & \mathcal{E} \parallel \mathcal{E} & \text{concatenation} \\
 & | & [\mathcal{E}]_s^s & \text{projection} \\
 & | & H(\mathcal{E}) & \text{hash} \\
 & | & f(\mathcal{E}) & \text{trapdoor permutation}
\end{array}
$$

Compilation to pWHILE procedure

$f((G(r) \oplus (m \| 0)) \| H(G(r) \oplus (m \| 0)) \oplus r)$ interpreted as:

$$
\begin{array}{l}
r \xleftarrow{\$} \{0,1\}^k; \\
g \leftarrow G(r); \\
s \leftarrow g \oplus (m \| 0); \\
h \leftarrow H(s); \\
\text{return } f_{pk}(s \| (h \oplus r))
\end{array}
$$

Proof system for IND-CPA

$$\frac{m \notin c^\star}{c^\star :_{\frac{1}{2}} \mathsf{Guess}}[\mathsf{Ind}] \qquad \frac{c^\star :_p \varphi \quad r \text{ fresh}}{(c^\star :_p \varphi)\{e \oplus r/r\}}[\mathsf{Opt}]$$

$$\frac{e \vdash \vec{r} \qquad \vec{r} \cap \mathcal{R}(c^\star) = \emptyset}{c^\star :_0 \mathsf{Ask}(H, e)}[\mathsf{Rnd}]$$

$$\frac{e \vdash^{\mathcal{A}} \vec{r} \qquad f(\vec{r}) \,\|\, \vec{r_0} \,\|\, m \vdash^{\mathcal{A}} c^\star \qquad \vec{r} \cap \vec{r_0} = \emptyset}{c^\star :_0 \mathsf{Ask}(H, e)}[\mathsf{OW}]$$

$$\frac{c^\star :_p \varphi \quad c^\star :_0 \mathsf{Ask}(H, e) \quad r \text{ and } H \text{ fresh}}{c^\star[H(e)/r] :_p \varphi[H(e)/r]}[\mathsf{Fail}]$$

Deducibility

$$\frac{}{e \vdash e} \qquad \frac{e \vdash e_1 \quad e \vdash e_2}{e \vdash e_1 \parallel e_2}[\text{Conc}] \qquad \frac{e \vdash e_1 \qquad e \vdash e_2}{e \vdash e_1 \oplus e_2}[\text{Xor}]$$

$$\frac{e \vdash e'}{e \vdash [e']_n^\ell}[\text{Proj}] \qquad \frac{e \vdash e_1 \qquad \vdash e_1 \doteq e_2}{e \vdash e_2}[\text{Conv}]$$

$$\frac{e \vdash e'}{e \vdash H(e')}[\text{H}] \qquad \frac{e \vdash e'}{e \vdash f(e')}[\text{F}] \qquad \boxed{\frac{e \vdash e'}{e \vdash f^{-1}(e')}[\text{Finv}]}$$

Example

$$\mathsf{f}(r) \,\|\, (H(r) \oplus m) :_{1/2} \text{Guess}$$

Example

$$\mathsf{f}(r) \,\|\, (H(r) \oplus m) :_{1/2} \text{Guess}$$

By [Fail],

$$\mathsf{f}(r) \,\|\, (r' \oplus m) :_{0} \text{Ask}(H, r)$$

$$\mathsf{f}(r) \,\|\, (r' \oplus m) :_{1/2} \text{Guess}$$

Example

$$\mathsf{f}(r) \,\|\, (H(r) \oplus m) :_{1/2} \text{Guess}$$

By [Opt],

$$\mathsf{f}(r) \,\|\, r' :_{0} \text{Ask}(H, r)$$

$$\mathsf{f}(r) \,\|\, r' :_{1/2} \text{Guess}$$

Example

$$\mathsf{f}(r) \,\|\, (H(r) \oplus m) :_{1/2} \text{Guess}$$

By [Opt],

$$\mathsf{f}(r) \,\|\, r' :_{0} \text{Ask}(H, r)$$

$$\mathsf{f}(r) \,\|\, r' :_{1/2} \text{Guess}$$

By [Ind], we discharge the second goal

Example

$$\mathsf{f}(r) \,\|\, (H(r) \oplus m) :_{1/2} \text{Guess}$$

By [Opt],

$$\mathsf{f}(r) \,\|\, r' :_0 \text{Ask}(H, r)$$

$$\mathsf{f}(r) \,\|\, r' :_{1/2} \text{Guess}$$

By [Ind], we discharge the second goal
By [OW], we discharge the first goal

OW rule

$$\frac{e \vdash^{\mathcal{A}} \vec{r} \qquad f(\vec{r}) \,\|\, \vec{r_0} \,\|\, m \vdash^{\mathcal{A}} c^\star \qquad \vec{r} \cap \vec{r_0} = \emptyset}{c^\star :_0 \text{Ask}(H, e)}\,[\text{OW}]$$

Filtering bad schemes

Use symbolic cryptography

- Apply correcteness check
  - ☞ is decryption possible with a key? $r \parallel f(r)$
- Apply simple filters, eg
  - ☞ is decryption possible without a key? $m \parallel f(r)$
  - ☞ is encryption randomized? $f(m)$
  - ☞ is randomness extractable without a key? $r \parallel f(m \oplus r)$
- Apply static equivalence

Applications of synthesis (so far)

- Padding-based encryption (2013)
- SPS (2015, 2016)
- Modes of operation (Malozemoff, Katz, Green, 2014)
- Tweakable blockciphers (Hoang, Katz, Malozemoff, 2015)
  - ☞ Analyzed 1,000s of schemes
  - ☞ Discovered several schemes competitive with OCB
- Transformative synthesis
  (Akinyele, Garman, Green, Hohenberger, Pagano, 2012, 2013, 2015)
  (Abe, Groth, Hoshino, Ohkubo, Tango, 2014, 2016)

Provable security of OAEP – reconsidered

Encryption $\mathcal{E}_{\mathrm{OAEP}(pk)}(m)$:
$r \xleftarrow{\$} \{0,1\}^{k_0}$;
$s \leftarrow G(r) \oplus (m \| 0^{k_1})$;
$t \leftarrow H(s) \oplus r$;
return $\mathsf{f}_{pk}(s \| t)$

Decryption $\mathcal{D}_{\mathrm{OAEP}(sk)}(c)$:
$(s,t) \leftarrow \mathsf{f}_{sk}^{-1}(c)$;
$r \leftarrow t \oplus H(s)$;
if $([s \oplus G(r)]_{k_1} = 0^{k_1})$
then $\{m \leftarrow [s \oplus G(r)]^{k}; \}$
else $\{m \leftarrow \perp; \}$
return m

Implementation of OAEP

Decryption $\mathcal{D}_{\mathrm{OAEP}(sk)}(c)$:
$(s,t) \leftarrow \mathsf{f}_{sk}^{-1}(c)$;
$r \leftarrow t \oplus H(s)$;
if $([s \oplus G(r)]_{k_1} = 0^{k_1})$
then $\{m \leftarrow [s \oplus G(r)]^k;\}$
else $\{m \leftarrow \perp;\}$
return m

Decryption $\mathcal{D}_{\mathrm{PKCS\text{-}C}(sk)}(res, c)$:
if $(c \in \mathrm{MsgSpace}(sk))$ then
$\{ (b0, s, t) \leftarrow \mathsf{f}_{sk}^{-1}(c)$;
$h \leftarrow MGF(s, hL)$; $i \leftarrow 0$;
while $(i < hLen + 1)$
$\{ s[i] \leftarrow t[i] \oplus h[i]$; $i \leftarrow i + 1$; $\}$
$g \leftarrow MGF(r, dbL)$; $i \leftarrow 0$;
while $(i < dbLen)$
$\{ p[i] \leftarrow s[i] \oplus g[i]$; $i \leftarrow i + 1$; $\}$
$l \leftarrow payload\_length(p)$;
if $(b0 = 0^8 \wedge [p]_l^{hLen} = 0..01 \wedge$
$[p]_{hLen} = LHash)$
then
$\{rc \leftarrow Success$;
$memcpy(res, 0, p, dbLen - l, l);\}$
else $\{rc \leftarrow DecryptionError;\}\}$
else $\{rc \leftarrow CiphertextTooLong;\}$
return rc;

Provable security of executable code

Proof by reduction:

FOR ALL adversary that breaks the assembly code,
THERE EXISTS an adversary that breaks the algorithm,
GIVEN the assembly code does not leak via side-channels

Proof relies on

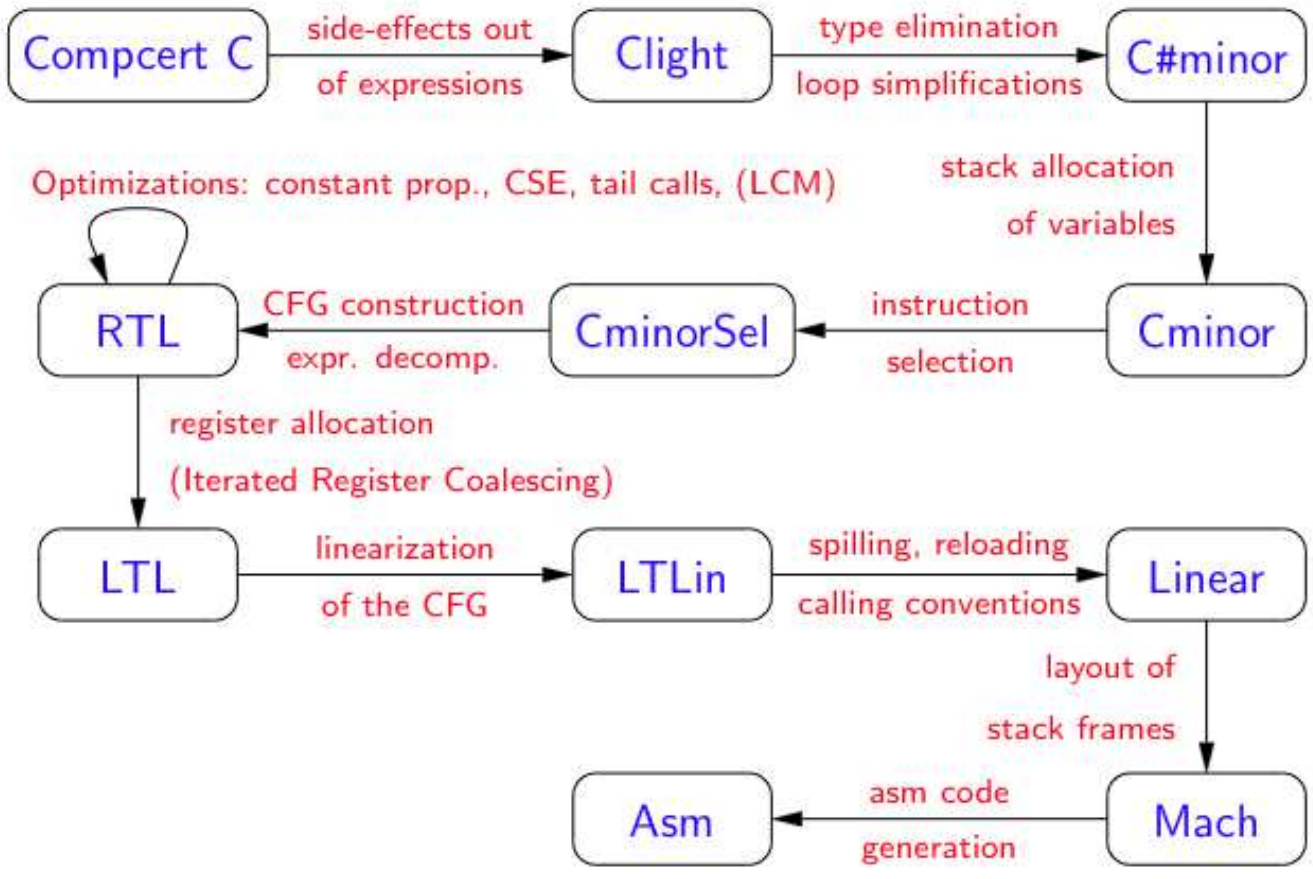
- provable security of algorithm
- functional correctness of assembly code
- side-channel security of assembly code

Caveats:

- semantics of assembly code idealized
- side-channel leakage idealized
- perfect randomness
- core operations are correctly implemented

From algorithms to implementations

- Algorithms to C: EasyCrypt C-mode, Frama-C
- C to assembly: CompCert (Leroy, 2006)



Side-channels: constant-time cryptography

A programming methodology against timing and cache attacks

- do not branch on secrets
- do not access arrays with secret indices

A good methodology?

- Non-constant-time implementations broken time and again
- May not rule out all attacks, but a great baseline

Synchronized product programs

$$\frac{c_1 \times c_2 \longrightarrow c \quad c_1' \times c_2' \longrightarrow c'}{c_1 ; c_1' \times c_2 ; c_2' \longrightarrow c ; c'}$$

$$\frac{c_1 \times c_2 \longrightarrow c}{\text{while } b_1 \text{ do } c_1 \times \text{while } b_2 \text{ do } c_2 \longrightarrow \{b_1 = b_2\}; \text{while } b_1 \text{ do } (c; \{b_1 = b_2\})}$$

$$\frac{c_1 \times c_2 \longrightarrow c \quad c_1' \times c_2' \longrightarrow c'}{\text{if } b_1 \text{ then } c_1 \text{ else } c_1' \times \text{if } b_2 \text{ then } c_2 \text{ else } c_2' \longrightarrow \{b_1 = b_2\}; \text{if } b_1 \text{ then } c \text{ else } c'}$$

- A variant of self-composition (2004)
- Applied to translation validation (2008)
- A particular instance of product program (2011)

Product programs and constant-time

- PC security = control flow independent of secrets
- Low equivalence $\sim$: memories coincide on public variables
- c is PC-secure iff

$$c_1 \times c_2 \longrightarrow c \quad \text{and} \quad \{\sim\}\, c\, \{\top\}$$

- CT security = PC security + memory access security
- c is CT-secure iff

$$c_1 \times c_2 \longrightarrow c \quad \text{and} \quad \{\sim\}\, c\, \{\top\}$$

  for a modified product construction
  (checking equality of array indices)
- Extends to public outputs

Applications of constant-time verifier

- Prototype (based on Smack)
  - ☞ Front-end (C to LLVM) unchanged
  - ☞ Boogie-to-Boogie translation
  - ☞ Simple invariant generation mechanism
  - ☞ Back-end to SMT unchanged
- Experimental results: NaCl, openSSL (part of MEE-CBC), libfixedtimefixedpoint, FourQ...
- Issues: LLVM vs asm, vectorized instructions, LLVM static analysis, counterexample generation, different from finding attack

Provably secure implementations

DPA attacks

- Measuring power consumption allows to retrieve keys
- Masking uses secret sharing to protect against DPA
  - ☞ each input is divided into d shares
  - ☞ computation operates on shares
- Achieves probabilistic non-intereference (PNI) wrt bounded sets of observations: the marginal distribution for any $t \leq d$ observations can be simulated from t shares of each input;
- PNI is easy to check for a fixed set of observations, but hard for all sets of observations is hard. Explosion as masking order d grows:
  - ☞ size of programs increases
  - ☞ number of observation sets explodes

Our Solution

Logic-based analysis for gadgets

- given a set of intermediate values known to be safe, efficiently extend it as much a possible
- still exponential, but pretty good in practice
- based on pRHL

Type-based analysis for algorithms

- automated checker (returns valid or violating tuple)
- compiler with automated insertion of refresh gadgets
- used to generate masked implementations of AES, Keccak, Simon, Speck at high orders
- type system based on fragment of theory of finite sets with cardinality constraints
- exploits a novel notion of strong non-interference, verified by many gadgets

Conclusion

- Probabilistic couplings are a useful tool for crypto proofs
- Formally verified many emblematic case studies
- Formal methods
  - can narrow the gap between provable security and implementations
  - are well-suited for reasoning about side-channels
  - efficient verifiers for masking and constant-time
  - can help advance state-of-art against DPA attacks
- Crypto offers many nice opportunities for PL and PV

`http://www.easycrypt.info`