Computer-aided cryptographic proofs and designs

Gilles Barthe (IMDEA, Spain)

Benjamin Grégoire (INRIA Sophia Antipolis, France)
Juan Manuel Crespo (IMDEA, Spain)
Francois Dupressoir (IMDEA, Spain)
César Kunz (IMDEA/U. Politecnica Madrid, Spain)
Yassine Lakhnech (U. de Grenoble/CNRS, France)
Benedikt Schmidt (IMDEA, Spain)
Pierre-Yves Strub (IMDEA, Spain)
Santiago Zanella Béguelin (MSR Cambridge, UK)

The CertiCrypt project (2006-)

Cryptographic proofs as program verification

- Formalize key notions and techniques using

  programming language semantics
  deductive program verification

- Provide machine support using off-the-shelf tools

  proof assistants
  SMT solvers

- Automation

  domain-specific logics; proof search
  systematic exploration of design space

- Modularity

What's wrong with provable security?

- *In our opinion, many proofs in cryptography have become essentially unverifiable. Our field may be approaching a crisis of rigor*. M. Bellare and P. Rogaway, 2004-2006
- *Do we have a problem with cryptographic proofs? Yes, we do [...] We generate more proofs than we carefully verify (and as a consequence some of our published proofs are incorrect)*. S. Halevi, 2005

Code-based cryptography

(Bellare & Rogaway 2004, Halevi 2005)

Everything is a probabilistic program

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

Code-based cryptography

(Bellare & Rogaway 2004, Halevi 2005)

Everything is a probabilistic program

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

- For cryptographers: rigorous notation
- In our work: rigorous justification of proofs

Code-based cryptography

Everything is a probabilistic program

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

Today:

$$
\begin{array}{rcll}
\mathcal{E} & ::= & \mathcal{E} \oplus \mathcal{E} & \text{xor} \\
& | & \mathcal{E} \parallel \mathcal{E} & \text{concatenation}
\end{array}
$$

- Uniform sampling on bitstrings of fixed length
- Memories map variables to bitstrings of fixed length
- Programs map memories to sub-distributions on memories

Code-based cryptography

Everything is a probabilistic program

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

Reductionist proofs:

For every feasible adversary $\mathcal{A}$ against scheme **S** (wrt goal **G**) **there exists** a feasible adversary $\mathcal{B}$ against assumption **H** st

$$\Pr_{\mathsf{G}_a}[\mathcal{A} \text{ breaks } \mathbf{S}] \leq h(\Pr_{\mathsf{G}_h}[\mathcal{B} \text{ breaks } \mathbf{H}])$$

A famous example: RSA-OAEP

Oracle $\mathrm{Enc}_{pk}(m)$:
$r \xleftarrow{\$} \{0,1\}^{k_0}$;
$s \leftarrow G(r) \oplus (m \| 0^{k_1})$;
$t \leftarrow H(s) \oplus r$;
return $f_{pk}(s \| t)$

Oracle $\mathrm{Dec}_{sk}(c)$:
$(s, t) \leftarrow f_{sk}^{-1}(c)$;
$r \leftarrow t \oplus H(s)$;
if $[s \oplus G(r)]_{k_1} = 0^{k_1}$ then return $[s \oplus G(r)]^n$
else return $\perp$

A famous example: RSA-OAEP

Oracle $\mathrm{Enc}_{pk}(m)$:

$r \xleftarrow{\$} \{0,1\}^{k_0}$;
$s \leftarrow G(r) \oplus (m \| 0^{k_1})$;
$t \leftarrow H(s) \oplus r$;
return $f_{pk}(s \| t)$

Oracle $\mathrm{Dec}_{sk}(c)$:

$(s, t) \leftarrow f^{-1}_{sk}(c)$;
$r \leftarrow t \oplus H(s)$;
if $[s \oplus G(r)]_{k_1} = 0^{k_1}$ then return $[s \oplus G(r)]^n$
else return $\perp$

Game IND-CCA2 :

$(sk, pk) \leftarrow \mathcal{KG}(\,)$;
$(m_0, m_1, \sigma) \leftarrow \mathcal{A}_1(pk)$;
$b \xleftarrow{\$} \{0,1\}$;
$c^* \leftarrow \mathrm{Enc}(pk, m_b)$;
$b' \leftarrow \mathcal{A}_2(pk, c^*, \sigma)$;
return $b = b'$

A famous example: RSA-OAEP

Oracle $\mathrm{Enc}_{pk}(m)$:
$r \xleftarrow{\$} \{0,1\}^{k_0}$;
$s \leftarrow G(r) \oplus (m \| 0^{k_1})$;
$t \leftarrow H(s) \oplus r$;
return $f_{pk}(s \| t)$

Oracle $\mathrm{Dec}_{sk}(c)$:
$(s, t) \leftarrow f_{sk}^{-1}(c)$;
$r \leftarrow t \oplus H(s)$;
if $[s \oplus G(r)]_{k_1} = 0^{k_1}$ then return $[s \oplus G(r)]^n$
else return $\perp$

Oracle $G(x)$:
if $x \notin \mathrm{dom}(L_G)$ then $L_G[x] \xleftarrow{\$} \{0,1\}^{n+k_1}$;
return $L_G[x]$

Oracle $H(x)$:
if $x \notin \mathrm{dom}(L_H)$ then $L_H[x] \xleftarrow{\$} \{0,1\}^{k_0}$;
return $L_H[x]$

Game IND-CCA2 :
$(sk, pk) \leftarrow \mathcal{KG}(\,)$;
$(m_0, m_1, \sigma) \leftarrow \mathcal{A}_1(pk)$;
$b \xleftarrow{\$} \{0,1\}$;
$c^* \leftarrow \mathrm{Enc}(pk, m_b)$;
$b' \leftarrow \mathcal{A}_2(pk, c^*, \sigma)$;
return $b = b'$

Game POW :
$(sk, pk) \leftarrow \mathcal{KG}()$;
$y \xleftarrow{\$} \{0,1\}^{n+k_1}$;
$z \xleftarrow{\$} \{0,1\}^{k_0}$;
$y' \leftarrow \mathcal{I}(f_{pk}(y \| z))$;
return $y = y'$

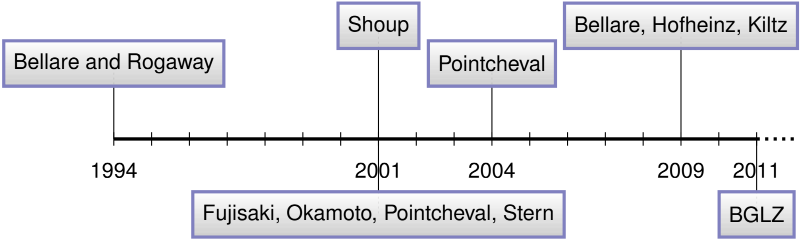
A famous example: RSA-OAEP

For every IND-CCA2 adversary $\mathcal{A}$ executing in time $t_{\mathcal{A}}$ there exists an inverter $\mathcal{I}$ executing in time $t_{\mathcal{I}}$ s.t.

$$\begin{aligned}\mathsf{Adv}_{\mathsf{IND\text{-}CCA2}(\mathcal{A})} &= \left|\Pr_{\mathsf{IND\text{-}CCA2}}[b = b'] - \tfrac{1}{2}\right| \\ &\leq \Pr_{\mathsf{POW}(\mathcal{I})}[y = y'] + \tfrac{2q_D q_G + q_D + q_G}{2^{k_0}} + \tfrac{q_D}{2^{k_1}}\end{aligned}$$

$$t_{\mathcal{I}} \simeq t_{\mathcal{A}} + q_D\ q_G\ q_H$$

A famous example: RSA-OAEP



1994 Purported proof of chosen-ciphertext security

2001 1994 proof gives weaker security; desired security holds

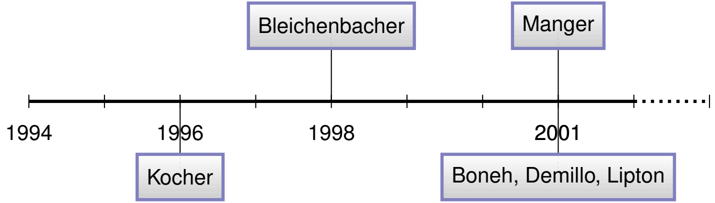
- for a modified scheme
- under stronger assumptions

2004 Filled gaps in Fujisaki et al. 2001 proof

2009 Security definition needs to be clarified

2011 Machine-checked proof

A famous example: RSA-OAEP



Attacks and countermeasures against implementations

1996 Timing attack

1998 Padding (million messages) attack

2001 Fault injection attack

2012 Machine-checked proof[1] for pseudo-implementation

201? Machine-checked proof[1] for implementation

[1] Interpret with care

The game-playing approach

(Shoup 2004, Bellare & Rogaway 2004, Halevi, 2005)

For every feasible adversary $\mathcal{A}$ against scheme **S** (wrt goal **G**) **there exists** a feasible adversary $\mathcal{B}$ against assumption **H** st

$$\Pr_{\mathsf{G}_a}[\mathcal{A} \text{ breaks } \mathbf{S}] \leq h(\Pr_{\mathsf{G}_h}[\mathcal{B} \text{ breaks } \mathbf{H}])$$

Game G_a : $\ldots$ $\ldots \leftarrow \mathcal{A}(\ldots)$; $\ldots$	**Game** G_1 : $\ldots$ $\ldots$ $\ldots$	$\ldots$	**Game** G_h : $\ldots$ $\ldots \leftarrow \mathcal{B}(\ldots)$; $\ldots$

$$\Pr_{\mathsf{G}_a}[\mathcal{A} \text{ breaks } \mathbf{S}] \quad \leq \quad h_1(\Pr_{\mathsf{G}_1}[E_1]) \leq \ldots \leq h(\Pr_{\mathsf{G}_h}[\mathcal{B} \text{ breaks } \mathbf{H}])$$

Example: IND-CPA security of BR93

Game IND-CPA :	$\mathsf{Enc}_{pk}(m)$:
$(sk, pk) \leftarrow \mathcal{KG}(\,);$ $(m_0, m_1, \sigma) \leftarrow \mathcal{A}_1(pk);$ $b \xleftarrow{\$} \{0,1\};$ $c^* \leftarrow \mathsf{Enc}(pk, m_b);$ $b' \leftarrow \mathcal{A}_2(pk, c^*, \sigma);$ return $b = b'$	$r \xleftarrow{\$} \{0,1\}^\ell;$ $s \leftarrow G(r) \oplus m;$ $y \leftarrow f_{pk}(r) \parallel s;$ return y

$G(x)$:
if $x \notin \mathrm{dom}(L_G)$ then $L_G[x] \xleftarrow{\$} \{0,1\}^k;$
return $L_G[x]$

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

For every IND-CPA adversary $\mathcal{A}$ making at most q_G queries to G, there exists an inverter $\mathcal{I}$ against OW such that

$$\left|\Pr_{\text{IND-CPA}}\left[b = b'\right] - \frac{1}{2}\right| \leq q_G\, \mathbf{Succ}_f^{\mathsf{OW}}(\mathcal{I})$$

Step 1: failure event

Game G_0 :	**Game** G_1 :
$L_G \leftarrow \emptyset;\ L_G^{\mathcal{A}} \leftarrow [\,];$	$L_G \leftarrow \emptyset;\ L_G^{\mathcal{A}} \leftarrow [\,];$
$(sk, pk) \leftarrow \mathcal{KG}();$	$(sk, pk) \leftarrow \mathcal{KG}();$
$(m_0, m_1, \sigma) \leftarrow \mathcal{A}_1(pk);$	$(m_0, m_1, \sigma) \leftarrow \mathcal{A}_1(pk);$
$b \stackrel{\$}{\leftarrow} \{0,1\};$	$b \stackrel{\$}{\leftarrow} \{0,1\};$
$r \stackrel{\$}{\leftarrow} \{0,1\}^{\ell};$	$r \stackrel{\$}{\leftarrow} \{0,1\}^{\ell};$
$g \leftarrow G(r);$	$g \stackrel{\$}{\leftarrow} \{0,1\}^{k};$
$s \leftarrow g \oplus m_b;$	$s \leftarrow g \oplus m_b;$
$c^* \leftarrow f_{pk}(r) \,\|\, s;$	$c^* \leftarrow f_{pk}(r) \,\|\, s;$
$b' \leftarrow \mathcal{A}_2(pk, c^*, \sigma);$	$b' \leftarrow \mathcal{A}_2(pk, c^*, \sigma);$

Step 1: failure event

Game G_0 :	**Game** G_1 :
$L_G \leftarrow \emptyset;\ L_G^{\mathcal{A}} \leftarrow [\,];$	$L_G \leftarrow \emptyset;\ L_G^{\mathcal{A}} \leftarrow [\,];$
$(sk, pk) \leftarrow \mathcal{KG}();$	$(sk, pk) \leftarrow \mathcal{KG}();$
$(m_0, m_1, \sigma) \leftarrow \mathcal{A}_1(pk);$	$(m_0, m_1, \sigma) \leftarrow \mathcal{A}_1(pk);$
$b \xleftarrow{\$} \{0,1\};$	$b \xleftarrow{\$} \{0,1\};$
$r \xleftarrow{\$} \{0,1\}^{\ell};$	$r \xleftarrow{\$} \{0,1\}^{\ell};$
$g \leftarrow G(r);$	$g \xleftarrow{\$} \{0,1\}^{k};$
$s \leftarrow g \oplus m_b;$	$s \leftarrow g \oplus m_b;$
$c^* \leftarrow f_{pk}(r) \,\|\, s;$	$c^* \leftarrow f_{pk}(r) \,\|\, s;$
$b' \leftarrow \mathcal{A}_2(pk, c^*, \sigma);$	$b' \leftarrow \mathcal{A}_2(pk, c^*, \sigma);$

The games are equivalent until the adversary queries G with r

$$\left|\Pr_{\mathsf{G}_0}\left[b = b'\right] - \Pr_{\mathsf{G}_1}\left[b = b'\right]\right| \leq \Pr_{\mathsf{G}_1}\left[r \in L_G^{\mathcal{A}}\right]$$

Step 2: optimistic sampling

Game G_1 :	**Game** G_2 :
$L_G \leftarrow \emptyset;\ L_G^{\mathcal{A}} \leftarrow [\,];$	$L_G \leftarrow \emptyset;\ L_G^{\mathcal{A}} \leftarrow [\,];$
$(sk, pk) \leftarrow \mathcal{KG}();$	$(sk, pk) \leftarrow \mathcal{KG}();$
$(m_0, m_1, \sigma) \leftarrow \mathcal{A}_1(pk);$	$(m_0, m_1, \sigma) \leftarrow \mathcal{A}_1(pk);$
$b \xleftarrow{\$} \{0,1\};$	$b \xleftarrow{\$} \{0,1\};$
$r \xleftarrow{\$} \{0,1\}^{\ell};$	$r \xleftarrow{\$} \{0,1\}^{\ell};$
$g \leftarrow \{0,1\}^{k};$	$s \xleftarrow{\$} \{0,1\}^{k};$
$s \leftarrow g \oplus m_b;$	$g \leftarrow s \oplus m_b;$
$c^* \leftarrow f_{pk}(r) \,\|\, s;$	$c^* \leftarrow f_{pk}(r) \,\|\, s;$
$b' \leftarrow \mathcal{A}_2(pk, c^*, \sigma);$	$b' \leftarrow \mathcal{A}_2(pk, c^*, \sigma);$

Step 2: optimistic sampling

Game G_1 :
$L_G \leftarrow \emptyset;\ L_G^{\mathcal{A}} \leftarrow [\,];$
$(sk, pk) \leftarrow \mathcal{KG}();$
$(m_0, m_1, \sigma) \leftarrow \mathcal{A}_1(pk);$
$b \xleftarrow{\$} \{0,1\};$
$r \xleftarrow{\$} \{0,1\}^{\ell};$
$g \leftarrow \{0,1\}^{k};$
$s \leftarrow g \oplus m_b;$
$c^* \leftarrow f_{pk}(r) \,\|\, s;$
$b' \leftarrow \mathcal{A}_2(pk, c^*, \sigma);$

Game G_2 :
$L_G \leftarrow \emptyset;\ L_G^{\mathcal{A}} \leftarrow [\,];$
$(sk, pk) \leftarrow \mathcal{KG}();$
$(m_0, m_1, \sigma) \leftarrow \mathcal{A}_1(pk);$
$b \xleftarrow{\$} \{0,1\};$
$r \xleftarrow{\$} \{0,1\}^{\ell};$
$s \xleftarrow{\$} \{0,1\}^{k};$
$g \leftarrow s \oplus m_b;$
$c^* \leftarrow f_{pk}(r) \,\|\, s;$
$b' \leftarrow \mathcal{A}_2(pk, c^*, \sigma);$

Games are equivalent and c^* is independent from b, hence

$$\left| \Pr_{\mathsf{IND\text{-}CPA}}\left[b = b'\right] - \frac{1}{2} \right| \leq \Pr_{\mathsf{G}_2}\left[r \in L_G^{\mathcal{A}}\right]$$

Step 3: reduction

Game G_2 :

$L_G \leftarrow \emptyset;\ L_G^{\mathcal{A}} \leftarrow [\,];$
$(sk, pk) \leftarrow \mathcal{KG}();$
$(m_0, m_1, \sigma) \leftarrow \mathcal{A}_1(pk);$
$r \xleftarrow{\$} \{0,1\}^{\ell};$
$s \xleftarrow{\$} \{0,1\}^{k};$
$c^* \leftarrow f_{pk}(r) \,\|\, s;$
$b' \leftarrow \mathcal{A}_2(pk, c^*, \sigma);$

Game OW :

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

Adversary $\mathcal{I}(x)$:

$L_G \leftarrow \emptyset; L_G^{\mathcal{A}} \leftarrow [\,];$
$(m_0, m_1, \sigma) \leftarrow \mathcal{A}_1(pk);$
$s \xleftarrow{\$} \{0,1\}^{k};\ y \leftarrow x \,\|\, s;$
$b' \leftarrow \mathcal{A}_2(pk, y, \sigma);$
$i \xleftarrow{\$} [1, |L_G^{\mathcal{A}}|];$
return $L_G^{\mathcal{A}}[i];$

Step 3: reduction

Game G_2 :
$L_G \leftarrow \emptyset;\ L_G^{\mathcal{A}} \leftarrow [\,];$
$(sk, pk) \leftarrow \mathcal{KG}();$
$(m_0, m_1, \sigma) \leftarrow \mathcal{A}_1(pk);$
$r \xleftarrow{\$} \{0,1\}^{\ell};$
$s \xleftarrow{\$} \{0,1\}^{k};$
$c^* \leftarrow f_{pk}(r) \,\|\, s;$
$b' \leftarrow \mathcal{A}_2(pk, c^*, \sigma);$

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

Adversary $\mathcal{I}(x)$:
$L_G \leftarrow \emptyset; L_G^{\mathcal{A}} \leftarrow [\,];$
$(m_0, m_1, \sigma) \leftarrow \mathcal{A}_1(pk);$
$s \xleftarrow{\$} \{0,1\}^{k};\ y \leftarrow x \,\|\, s;$
$b' \leftarrow \mathcal{A}_2(pk, y, \sigma);$
$i \xleftarrow{\$} [1, |L_G^{\mathcal{A}}|];$
return $L_G^{\mathcal{A}}[i];$

Inverter wins with probability $\frac{1}{q_G}$ if $r \in L_G^{\mathcal{A}}$, and 0 otherwise

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

pRHL: Relational Hoare Logic for pWHILE

- Judgment:
$$c_1 \sim c_2 : P \Rightarrow Q$$
where P and Q are relations on memories
- Validity:
$$\models c_1 \sim c_2 : P \Rightarrow Q$$
iff for all memories m_1 and m_2
$$(m_1, m_2) \models P \to ([\![c_1]\!]_{m_1}, [\![c_2]\!]_{m_2}) \models Q^\sharp$$
- $\cdot^\sharp$ lifts relations on $A \times B$ to relations on $\mathcal{D}(A) \times \mathcal{D}(B)$

Recommended reading

Yuxin Deng and Wenjie Du. *Logical, Metric, and Algorithmic Characterisations of Probabilistic Bisimulation*.
TR CMU-CS-11-110, Carnegie Mellon University, March 2011

Lifting Relations to Distributions

$\Psi^\sharp \subseteq \mathcal{D}(A) \times \mathcal{D}(B)$ is the smallest relation that satisfies:

- If $(s, t) \models \Psi$ then $(\mathrm{unit}(s), \mathrm{unit}(t)) \models \Psi^\sharp$
- If $(\mu_i, \nu_i) \models \Psi^\sharp$ and $\sum_i p_i = 1$, then

$$\left(\sum_i p_i \, \mu_i, \sum_i p_i \, \nu_i \right) \models \Psi^\sharp$$

Lifting Relations to Distributions

Alternatively, $(\mu_1, \mu_2) \models \Psi^\sharp$ iff there exists $\mu \in \mathcal{D}(\mathcal{A} \times B)$ s.t.

- The 1[st] projection of μ coincides with μ_1
- The 2[nd] projection of μ coincides with μ_2
- The support of μ is a subset of Ψ

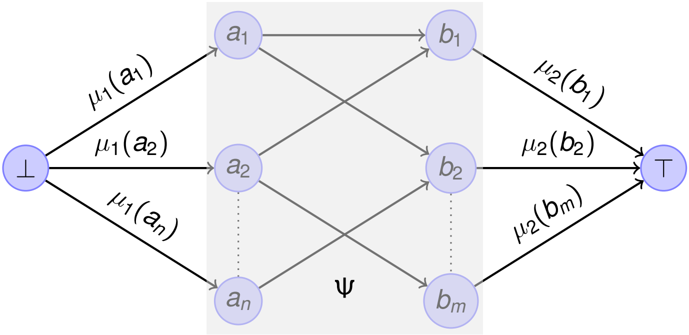
Extensions

This characterization is convenient to consider extensions to

- statistical distance
- α-distance (for differential privacy)
- f-divergence (for Kullback-Leibler or Hellinger distance)

Lifting Relations to Distributions

Alternatively, $(\mu_1, \mu_2) \models \Psi^\sharp$ iff the maximum flow in the following network is 1:

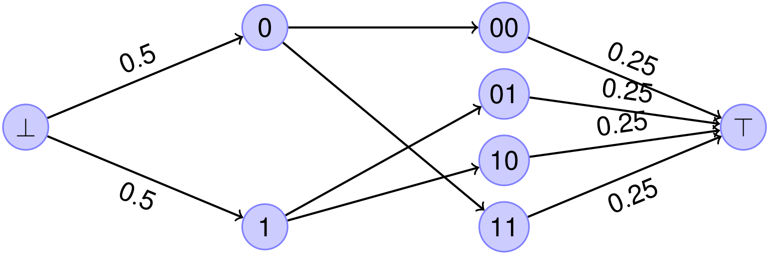


Example

$$c_1 \stackrel{\text{def}}{=} x \xleftarrow{\$} \{0,1\} \qquad c_2 \stackrel{\text{def}}{=} x \xleftarrow{\$} \{0,1\};\; y \xleftarrow{\$} \{0,1\}$$

- c_1 generates a distribution μ_1 over $\{0,1\}$
- c_2 generates a distribution μ_2 over $\{0,1\}^2$
- Consider $\Psi \stackrel{\text{def}}{=} x\langle 1\rangle = x\langle 2\rangle \oplus y\langle 2\rangle$

Q: Does $(\mu_1, \mu_2) \models \Psi$ hold?

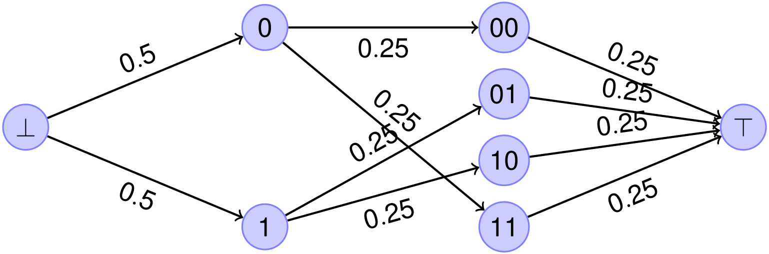


Example

$$c_1 \stackrel{\text{def}}{=} x \xleftarrow{\$} \{0,1\} \qquad c_2 \stackrel{\text{def}}{=} x \xleftarrow{\$} \{0,1\};\ y \xleftarrow{\$} \{0,1\}$$

- c_1 generates a distribution μ_1 over $\{0,1\}$
- c_2 generates a distribution μ_2 over $\{0,1\}^2$
- Consider $\Psi \stackrel{\text{def}}{=} x\langle 1\rangle = x\langle 2\rangle \oplus y\langle 2\rangle$

Q: Does $(\mu_1, \mu_2) \models \Psi$ hold?
A: Yes, because we can construct a flow of value 1 in the corresponding network



Rules for assignments

Random assignment

$$\frac{f \text{ is 1-1 and } Q' \stackrel{\text{def}}{=} \forall v, Q\{x\langle 1\rangle := f\ v, x\langle 2\rangle := v\}}{\models x \xleftarrow{\$} A \sim x \xleftarrow{\$} A : Q' \Rightarrow Q}$$

- Captures a special case of liftings
- More general rules exist, but are not implemented
- Would still be incomplete

Assignment

$$\frac{}{\models x \leftarrow e \sim x' \leftarrow e' : Q\{x\langle 1\rangle := e\langle 1\rangle, x'\langle 2\rangle := e'\langle 2\rangle\} \Rightarrow Q}$$

$$\frac{}{\models x \leftarrow e \sim \text{nil} : Q\{x\langle 1\rangle := e\langle 1\rangle\} \Rightarrow Q}$$

Rules for conditionals

Conditionals

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

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

Loops

- Two-sided rule: loops make the same number of iterations
- One-sided rules: loop unrolling (left or right)
- Advanced loop optimizations through product construction (not integrated in EasyCrypt)

Adversaries

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

- Adversaries are sequences of oracle calls
- No functional specification
- Given the same inputs, provide the same outputs

Cryptographic reasoning with pRHL

pRHL captures common patterns in cryptographic proofs

- Failure events: if $\vDash c_1 \sim c_2 : P \Rightarrow \neg F\langle 2\rangle \rightarrow Q_1\langle 1\rangle \leftrightarrow Q_2\langle 2\rangle$ then

$$(m_1, m_2) \vDash P \Longrightarrow |\Pr_{c_1,m_1}[Q_1] - \Pr_{c_2,m_2}[Q_2]| \leq \Pr_{c_2,m_2}[F]$$

- Bridging steps: if $\vDash c_1 \sim c_2 : P \Rightarrow =$ then for all events Q

$$(m_1, m_2) \vDash P \Longrightarrow \Pr_{c_1,m_1}[Q] = \Pr_{c_2,m_2}[Q]$$

  (pRHL subsumes obs. equiv./prob. non-interference and validates many compiler optimizations)

- Reductions: if $\vDash c_1 \sim c_2 : P \Rightarrow Q_1\langle 1\rangle \rightarrow Q_2\langle 2\rangle$, then

$$(m_1, m_2) \vDash P \Longrightarrow \Pr_{c_1,m_1}[Q_1] \leq \Pr_{c_2,m_2}[Q_2]$$

- Eager/lazy sampling

Tool support and examples

CertiCrypt: formally verified Coq libraries

- Optimizations and probabilistic relational Hoare logic
- Verified against operational semantics based on ALEA

EasyCrypt: SMT-based verification tool

- Probabilistic relational Hoare logic
- Verification condition generation + why3 back-end
- Accessible to cryptographers

Examples

- Crypto: public-key encryption, block ciphers, signatures, hash designs, zero-knowledge proofs of knowledge, authenticated key exchange protocols
- Differential privacy: continuous statistics, approximation algorithms, synthetic databases, 2-party computation

The story so far

A unifying formalism to justify cryptographic proofs, but:

- no instantiation mechanism
- no proof-theoretical analysis of cryptographic reasoning
- no proof discovery mechanism
- no systematic analysis of classes of cryptographic systems

Challenges and opportunities

- mechanisms for modular proofs
- domain-specific logics and proof search algorithms
- decision procedures
- exhaustive exploration and practical interpretation

Modular proofs

- OAEP is a generic conversion: it transforms a one-way function into an IND-CCA2 scheme
- Many cryptographic constructions are generic conversions

Careful with instantiation/indifferentiability

- Instantiated schemes often expose more state than assumed in generic proofs. Probabilistic encapsulation quantifies the amount of information leaked by instantiation
- Generic conversions hinge on negative hypotheses. Our modules integrate negative constraints that can be verified during instantiation

Applications (ongoing):

- Authenticated key exchange
- Modes of operation

Automated proofs and exploration

The next 700 cryptosystems (after Landin, 1966)

Do the cryptosystems reflect [...] the situations that are being catered for? Or are they accidents of history and personal background that may be obscuring fruitful developments?
[...]
We must think in terms, not of cryptosystems, but of families of cryptosystems. That is to say we must systematize their design so that a new system is a point chosen from a well-mapped space, rather than a laboriously devised construction.

An algebraic view of padding-based schemes

Encryption algorithms are modelled as algebraic expressions

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

Decryption algorithms are modelled using list comprehension

$$\vec{x} \stackrel{c}{\leftarrow} \vec{L}_H^{\mathcal{A}} : T \triangleright e$$

where $\quad T ::= e = e \mid e \in L_H \mid e \in L_H^{\mathcal{A}} \mid T \wedge T$

Semantics

Left-to-right evaluation with sharing, yields a pWHILE procedure

Example

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

interpreted as:

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

Deducibility

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

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

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

Convertibility

- Based on equational theory of bitstrings
- Decidable for probabilistic expressions without H, f, f^{-1}

Useful for

- Expressing proof rules
- Discovering decryption algorithm and finding attacks

Proof principles

Chosen-plaintext security

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

Chosen-ciphertext security

Extensionality	Replace e or d by equivalent ones
Plaintext extractor	“Public” decryption oracle can be eliminated

Completeness:

- Holds empirically for IND-CPA
- Fails for IND-CCA2: RO not programmable

Proof system for IND-CPA

Side-conditions apply

$$\frac{m \notin \mathcal{V}(c^*)}{c^* :_{\frac{1}{2}} \text{Guess}}[\text{Indep}] \qquad \frac{e \vdash \vec{r} \qquad \vec{r} \cap \mathcal{R}(c^*) = \emptyset}{c^* :_{q_\mathsf{H}\, 2^{-|\vec{r}|}} e \in L_H^{\mathcal{A}}}[\text{Indom}]$$

$$\frac{e \vdash_t^{\mathcal{A}} [\vec{r}]_0^{\ell} \qquad f(\vec{r}) \,\|\, \mathcal{R}(c^*\{0/f(\vec{r})\}) \,\|\, \mathcal{V}(c^*) \vdash_{t'}^{\mathcal{A}} c^*}{c^* :_{\mathbf{Succ}_{\Theta}^{\mathsf{OW}_\ell^{q_\mathsf{H}}}(t'')} e \in L_H^{\mathcal{A}}}[\text{OW}]$$

Soundness

- Once and for all
  - \+ "global" guarantee; relatively simple and intuitive
  - \+ avoids resorting to intermediate framework
- By generating a pRHL/EasyCrypt proof for each scheme
  - \+ limits Trusted Computing Base
  - \+ proofs can be combined and reused
  - \- currently restricted to IND-CPA

Systematic exploration

- Generate well-typed terms up to user-defined constraints
- Check for decryption algorithm and attacks
- Launch proof search (strategy + backtracking)
- Compile successful runs to EasyCrypt (for IND-CPA)
- Practical interpretation

Searching attacks

- IND-CPA

  is decryption possible without a key? $m \parallel f(r)$
  is encryption randomized? $f(m)$
  is randomness extractable without a key? $r \parallel f(m \oplus r)$

- IND-CCA2

  is encryption malleable? $f(r) \parallel m \oplus G(r)$

Experiments

- Analyzed over 100 variants of OAEP
- Applied crawler with many different size constraints; filters discard over 99% of candidates
- Incomplete, but experimentally ok.

Some numbers

Op	GEN	$\neg$ CPA	CPA	$\neg$ CCA	CCA
4	2	0	2	2	0
5	44	27	12	9	0
6	419	244	104	68	1
7	4131	2392	883	537	39
8	41860	24166	7850	4424	436
9	275318	155669	54884	27697	3750

ZAEP

Two minimal schemes

$$\mathrm{BR93}: \; f(r) \,\|\, (G(r) \oplus m) \qquad \mathrm{ZAEP}: \; f(r \,\|\, G(r) \oplus m)$$

ZAEP is redundant-free

$$\mathrm{Dec}(c) : r \,\|\, t \leftarrow f_{sk}^{-1}(c); g \leftarrow G(r); \text{return } t \oplus g$$

INDCCA Security of ZAEP for RSA exponent 2 and 3

$$\left| \Pr_{\text{IND-CCA2}}[b = b'] - \frac{1}{2} \right| \leq \mathbf{Succ}_f^{\mathrm{OW}}(\mathcal{I}) + \frac{q_D}{2^n}$$

Based on existence of two efficient algorithms:

- CIE: given $f(r, s_1)$, $f(r, s_2)$ with $s_1 \neq s_2$, returns s_1, s_2 and r
- SIE: given $f(r, s)$ and r returns s

Conclusion

High-assurance cryptographic proofs

- Rigorous proofs using PL techniques (pRHL)
- Independent verification

New directions

- Modularity
- Atlas of cryptographic constructions
- Real-world cryptography; verifying implementations
- EasyCrypt components:

  relational invariant inference
  logics and decision procedures
  automated complexity analysis
  resurrect certification in Coq

Further information and tools

`http://easycrypt.gforge.inria.fr`