CertiCrypt

Formal certification of code-based cryptographic proofs

Santiago Zanella[1,2]

Gilles Barthe[3] Benjamin Grégoire[1,2] Sylvain Heraud[2]

[1]Microsoft Research - INRIA Joint Centre, France

CENTRE DE RECHERCHE COMMUN INRIA MICROSOFT RESEARCH

[2]INRIA Sophia Antipolis - Méditerranée, France

INRIA SOPHIA ANTIPOLIS

[3]IMDEA Software, Madrid, Spain

imdea instituto madrileño de estudios avanzados

CosyProofs '09

2009.04.07

What's wrong with cryptographic proofs?

- *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.
- *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
- *Security proofs in cryptography may be organized as sequences of games [...] this can be a useful tool in taming the complexity of security proofs that might otherwise become so messy, complicated, and subtle as to be nearly impossible to verify*
  V. Shoup

Our goal

To mechanize the construction and verification of direct computational proofs, structured as sequences of games

Why direct computational proofs matter?

- More convincing and general
- Results easily interpretable
- Exact security bounds
- Give hints as to how to choose practical parameters
- Reductionist proofs are much more informative than a Yes/No answer

Game-based cryptographic proofs

Attack Game

Game G_0^η :
$\cdots$
$\mathcal{A}$
$\cdots$

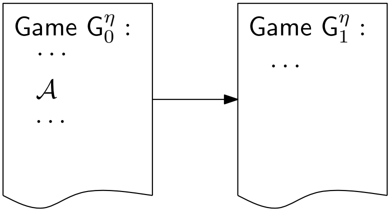
$\mathsf{Pr}_{\mathsf{G}_0^\eta}[A_0]$

$$\boxed{\mathsf{Pr}_{\mathsf{G}_0^\eta}[A_0] \leq \epsilon(\eta)}$$

Security property

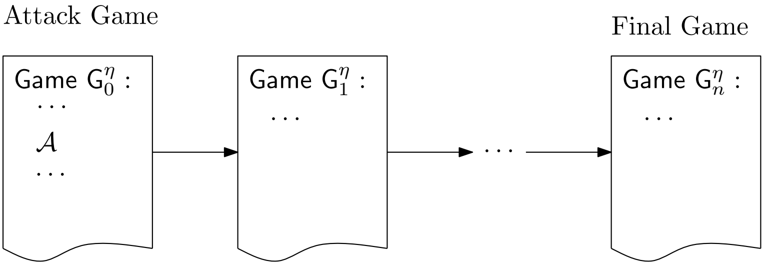
Game-based cryptographic proofs

Attack Game



$$\Pr_{\mathsf{G}_0^\eta}[A_0] \quad \leq \quad h_1(\Pr_{\mathsf{G}_1^\eta}[A_1])$$

Game-based cryptographic proofs



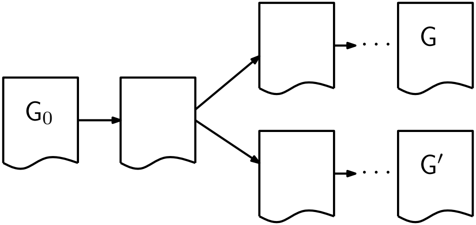
$$\mathsf{Pr}_{G_0^\eta}[A_0] \quad \leq \quad h_1(\mathsf{Pr}_{G_1^\eta}[A_1]) \quad \leq \quad \cdots \quad \leq \quad h_n(\mathsf{Pr}_{G_n^\eta}[A_n])$$

$$\mathsf{Pr}_{G_0^\eta}[A_0] \leq h(\mathsf{Pr}_{G_n^\eta}[A_n]) \leq \epsilon(\eta)$$

For any *computationally feasible* adversary $\mathcal{A}$

Game-based proofs: essence and problems

Independent events



$$\Pr_{\mathsf{G}_0}[A \wedge A'] \leq \Pr_{\mathsf{G}}[A] \times \Pr_{\mathsf{G}'}[A'])$$

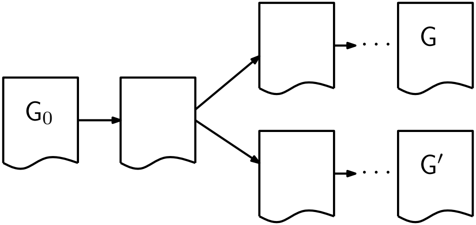
Essence: relate the probability of events in consecutive games

But,

- How do we represent games?
- What adversaries are *feasible*?
- How do we make a proof hold for any feasible adversary?

Game-based proofs: essence and problems

Independent events



$$\Pr_{G_0}[A \wedge A'] \leq \Pr_{G}[A] \times \Pr_{G'}[A'])$$

Essence: relate the probability of events in consecutive games

But,

- How do we represent games?
- What adversaries are *feasible*?
- How do we make a proof hold for any feasible adversary?

Language-based proofs

What if we represent games as programs?

Games	$\Longrightarrow$	programs
Probability space	$\Longrightarrow$	program denotation
Game transformations	$\Longrightarrow$	program transformations
Generic adversary	$\Longrightarrow$	unspecified procedure
Feasibility	$\Longrightarrow$	Probabilistic Polynomial-Time

pWHILE: a probabilistic programming language

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

- $x \xleftarrow{\$} d$: sample the value of x according to distribution d. d may depend on program variables.

Semantics of programs

Measure monad: $M(X) \stackrel{\text{def}}{=} (X \to [0,1]) \to [0,1]$

$$[\![\mathsf{G}]\!] : \forall\, \eta, \mathcal{M} \to M(\mathcal{M})$$

- Interpret $[\![\mathsf{G}]\!]^\eta m$ as the expectation operator of the probability distribution induced by the game.
- Probability: $\Pr_{\mathsf{G}^\eta, m}[A] \stackrel{\text{def}}{=} [\![\mathsf{G}]\!]^\eta\ m\ \mathbb{1}_A$

Example.

$$\text{Let } \mathsf{G} \stackrel{\text{def}}{=} x \stackrel{\$}{\leftarrow} \{0,1\};\ y \stackrel{\$}{\leftarrow} \{0,1\}$$

$\Pr_{\mathsf{G}^\eta, m}[x \neq y] = [\![\mathsf{G}]\!]^\eta\ m\ \mathbb{1}_{x \neq y} =$

Semantics of programs

Measure monad: $M(X) \stackrel{\text{def}}{=} (X \to [0,1]) \to [0,1]$

$$[\![\mathrm{G}]\!] : \forall\, \eta, \mathcal{M} \to M(\mathcal{M})$$

- Interpret $[\![\mathrm{G}]\!]^\eta m$ as the expectation operator of the probability distribution induced by the game.
- Probability: $\Pr_{\mathrm{G}^\eta,m}[A] \stackrel{\text{def}}{=} [\![\mathrm{G}]\!]^\eta\ m\ \mathbb{1}_A$

Example.

$$\text{Let } \mathrm{G} \stackrel{\text{def}}{=} x \stackrel{\$}{\leftarrow} \{0,1\};\ y \stackrel{\$}{\leftarrow} \{0,1\}$$

$$\begin{aligned}
&\Pr_{\mathrm{G}^\eta,m}[x \neq y] = [\![\mathrm{G}]\!]^\eta\ m\ \mathbb{1}_{x\neq y} = \\
&\tfrac{1}{4}\ \mathbb{1}_{x\neq y}(m[x \mapsto 0, y \mapsto 0]) \quad + \quad \tfrac{1}{4}\ \mathbb{1}_{x\neq y}(m[x \mapsto 0, y \mapsto 1]) \quad + \\
&\tfrac{1}{4}\ \mathbb{1}_{x\neq y}(m[x \mapsto 1, y \mapsto 0]) \quad + \quad \tfrac{1}{4}\ \mathbb{1}_{x\neq y}(m[x \mapsto 1, y \mapsto 1])
\end{aligned}$$

Semantics of programs

Measure monad: $M(X) \stackrel{\text{def}}{=} (X \to [0,1]) \to [0,1]$

$$[\![G]\!] : \forall\, \eta, \mathcal{M} \to M(\mathcal{M})$$

- Interpret $[\![G]\!]^\eta m$ as the expectation operator of the probability distribution induced by the game.
- Probability: $\Pr_{G^\eta, m}[A] \stackrel{\text{def}}{=} [\![G]\!]^\eta \; m \; \mathbb{1}_A$

Example.

$$\text{Let } G \stackrel{\text{def}}{=} x \stackrel{\$}{\leftarrow} \{0,1\};\; y \stackrel{\$}{\leftarrow} \{0,1\}$$

$$\Pr_{G^\eta, m}[x \neq y] = [\![G]\!]^\eta \; m \; \mathbb{1}_{x \neq y} =$$

$$0 \quad + \quad \frac{1}{4} \quad +$$

$$\frac{1}{4} \quad + \quad 0$$

Semantics of programs

Measure monad: $M(X) \stackrel{\text{def}}{=} (X \to [0,1]) \to [0,1]$

$$[\![\mathsf{G}]\!] : \forall\, \eta, \mathcal{M} \to M(\mathcal{M})$$

- Interpret $[\![\mathsf{G}]\!]^\eta m$ as the expectation operator of the probability distribution induced by the game.
- Probability: $\Pr_{\mathsf{G}^\eta, m}[A] \stackrel{\text{def}}{=} [\![\mathsf{G}]\!]^\eta\ m\ \mathbb{1}_A$

Example.

$$\text{Let } \mathsf{G} \stackrel{\text{def}}{=} x \stackrel{\$}{\leftarrow} \{0,1\};\ y \stackrel{\$}{\leftarrow} \{0,1\}$$

$$\Pr_{\mathsf{G}^\eta, m}[x \neq y] = [\![\mathsf{G}]\!]^\eta\ m\ \mathbb{1}_{x \neq y} = \frac{1}{2}$$

Characterizing feasible adversaries

A non-intrusive cost model for reasoning about program complexity

$$[\![\mathrm{G}]\!]' : \forall \eta,\ (\mathcal{M} \times \mathbb{N}) \to M(\mathcal{M} \times \mathbb{N})$$

A program G runs in probabilistic polynomial time if:

- It terminates with probability 1 (i.e. $\forall m,\ \Pr_{\mathrm{G},m}[\mathrm{true}] = 1$)
- There exists a polynomial $p(\cdot)$ s.t. if (m', n) is reachable with positive probability, then $n \leq p(\eta)$

Representing Random Oracles

Random oracles can be represented as stateful procedures

$$\begin{array}{l}\textbf{Oracle } \mathcal{O}(x) : \\ \ \text{if } x \notin \mathsf{dom}(\mathbf{L}) \text{ then} \\ \quad y \xleftarrow{\$} \{0,1\}^{\eta};\ \mathbf{L} \leftarrow (x,y) :: \mathbf{L} \\ \text{return } \mathbf{L}(x)\end{array}$$

- Variable **L** is global

Program equivalence

Definition (Observational equivalence)

$$
\begin{array}{rcl}
f =_X g & \stackrel{\text{def}}{=} & \forall m_1\ m_2,\ m_1(X) = m_2(X) \implies f\ m_1 = g\ m_2 \\
\vDash \mathrm{G}_1 \simeq^I_O \mathrm{G}_2 & \stackrel{\text{def}}{=} & \forall m_1\ m_2\ f\ g,\ m_1(I) = m_2(I)\ \wedge\ f =_O g \implies \\
& & [\![\mathrm{G}_1]\!]\ m_1\ f = [\![\mathrm{G}_2]\!]\ m_2\ g
\end{array}
$$

Generalizes information flow security (take $I = O = \mathcal{V}_{\text{low}}$)

But is not general enough...

$$
\frac{???}{\vDash \text{if } x = 0 \text{ then } y \leftarrow x \text{ else } y \leftarrow 1 \simeq^{\{x\}}_{\{x,y\}} \text{if } x = 0 \text{ then } y \leftarrow 0 \text{ else } y \leftarrow 1}
$$

Program equivalence

Definition (Observational equivalence)

$$f =_X g \quad \stackrel{\text{def}}{=} \quad \forall m_1\ m_2,\ m_1(X) = m_2(X) \implies f\ m_1 = g\ m_2$$

$$\models \mathsf{G}_1 \simeq^I_O \mathsf{G}_2 \quad \stackrel{\text{def}}{=} \quad \forall m_1\ m_2\ f\ g,\ m_1(I) = m_2(I)\ \wedge\ f =_O g \implies [\![\mathsf{G}_1]\!]\ m_1\ f = [\![\mathsf{G}_2]\!]\ m_2\ g$$

Generalizes information flow security (take $I = O = \mathcal{V}_{\text{low}}$)
But is not general enough...

$$\frac{???}{\models \text{if } x = 0 \text{ then } y \leftarrow x \text{ else } y \leftarrow 1 \simeq^{\{x\}}_{\{x,y\}} \text{if } x = 0 \text{ then } y \leftarrow 0 \text{ else } y \leftarrow 1}$$

Program equivalence

Definition (Observational equivalence, generalization)

$$\models G_1 \sim G_2 : \Psi \Rightarrow \Phi \stackrel{\text{def}}{=} \forall m_1\ m_2.\ m_1\ \Psi\ m_2 \Rightarrow [\![G_1]\!]\ m_1 \sim_\Phi [\![G_2]\!]\ m_2$$

Where $\sim_\Phi$ lifts Φ from memories to distributions.

$$(x = 0) \sim_{\{x\}} (x = 0)$$

$$\frac{\models y \leftarrow x \sim y \leftarrow 0 : =_{\{x\}} \wedge (x = 0)\langle 1 \rangle \Rightarrow =_{\{x,y\}} \qquad \models y \leftarrow 1 \sim y \leftarrow 1 : =_{\{x\}} \wedge (x \neq 0)\langle 1 \rangle \Rightarrow =_{\{x,y\}}}{\text{if } x = 0 \text{ then } y \leftarrow x \text{ else } y \leftarrow 1 \sim \text{if } x = 0 \text{ then } y \leftarrow 0 \text{ else } y \leftarrow 1 : =_{\{x\}} \Rightarrow =_{\{x,y\}}}$$

Program equivalence

Definition (Observational equivalence, generalization)

$$\models G_1 \sim G_2 : \Psi \Rightarrow \Phi \stackrel{\text{def}}{=} \forall m_1\ m_2.\ m_1\ \Psi\ m_2 \Rightarrow [\![G_1]\!]\ m_1 \sim_\Phi [\![G_2]\!]\ m_2$$

Where $\sim_\Phi$ lifts Φ from memories to distributions.

$$\frac{\begin{array}{c}(x = 0) \sim_{\{x\}} (x = 0) \\ \models y \leftarrow x \sim y \leftarrow 0 : =_{\{x\}} \wedge (x = 0)\langle 1\rangle \Rightarrow =_{\{x,y\}} \\ \models y \leftarrow 1 \sim y \leftarrow 1 : =_{\{x\}} \wedge (x \neq 0)\langle 1\rangle \Rightarrow =_{\{x,y\}}\end{array}}{\begin{array}{l}\text{if } x = 0 \text{ then } y \leftarrow x \text{ else } y \leftarrow 1 \sim \\ \text{if } x = 0 \text{ then } y \leftarrow 0 \text{ else } y \leftarrow 1 : =_{\{x\}} \Rightarrow =_{\{x,y\}}\end{array}}$$

From program equivalence to probability

Let A be an event that depends only on variables in O

To prove $\Pr_{\mathsf{G}_1,m_1}[A] = \Pr_{\mathsf{G}_2,m_2}[A]$ it suffices to show

- $\models \mathsf{G}_1 \simeq^I_O \mathsf{G}_2$
- $m_1 =_I m_2$

Proving program equivalence

Goal

$$\models \mathsf{G}_1 \simeq^I_O \mathsf{G}_2$$

A Relational Hoare Logic

$$\frac{\models c_1 \sim c_2 : \Phi \Rightarrow \Phi' \qquad \models c_1' \sim c_2' : \Phi' \Rightarrow \Phi''}{\models c_1; c_1' \sim c_2; c_2' : \Phi \Rightarrow \Phi''}\text{[R-Seq]}$$

$\ldots$

Proving program equivalence

Goal

$$\vDash \mathsf{G}_1 \simeq^{I}_{O} \mathsf{G}_2$$

Mechanized program transformations

- Transformation: $T(\mathsf{G}_1, \mathsf{G}_2, I, O) = (\mathsf{G}'_1, \mathsf{G}'_2, I', O')$
- Soundness theorem

$$\frac{T(\mathsf{G}_1, \mathsf{G}_2, I, O) = (\mathsf{G}'_1, \mathsf{G}'_2, I', O') \qquad \vDash \mathsf{G}'_1 \simeq^{I'}_{O'} \mathsf{G}'_2}{\vDash \mathsf{G}_1 \simeq^{I}_{O} \mathsf{G}_2}$$

- Reflection-based Coq tactic

Proving program equivalence

Goal

$$\models G_1 \simeq^I_O G_2$$

Mechanized program transformations

- Dead code elimination (`deadcode`)
- Constant folding and propagation (`ep`)
- Procedure call inlining (`inline`)
- Code movement (`swap`)
- Common suffix/prefix elimination (`eqobs_hd`, `eqobs_tl`)

Proving program equivalence

Goal

$\models G \simeq^{I}_{O} G$

A semi-decision procedure for self-equivalence
(`eqobs_in`)

- Does $\models G \simeq^{I}_{O} G$ hold?
- Analyze dependencies to compute I' s.t. $\models G \simeq^{I'}_{O} G$
- Check that $I' \subseteq I$
- Think about information flow security...

Proving program equivalence

Goal

$$\models G_1 \sim G_2 : \Psi \Rightarrow \Phi$$

A mechanized Weakest Precondition calculus
(sound, but incomplete)

- Compute wp Ψ' s.t. $\models G_1 \sim G_2 : \Psi' \Rightarrow \Phi$
- Generate proof obligation $\Psi \implies \Psi'$

The Fundamental Lemma of Game-Playing

Fundamental lemma

If two games G_1 and G_2 behave identically in an initial memory m unless a failure event F fires, then

$$|\Pr_{\mathsf{G}_1,m}[A] - \Pr_{\mathsf{G}_2,m}[A]| \leq \Pr_{\mathsf{G}_{1,2}}[F]$$

The Fundamental Lemma of Game-Playing

Syntactic criterion

Game G_1 :
. . .
bad $\leftarrow$ true; c_1
. . .

Game G_2 :
. . .
bad $\leftarrow$ true; c_2
. . .

- $\Pr_{\mathsf{G}_1,m}[A \mid \neg \mathsf{bad}] = \Pr_{\mathsf{G}_2,m}[A \mid \neg \mathsf{bad}]$
- $\Pr_{\mathsf{G}_1,m}[\mathsf{bad}] = \Pr_{\mathsf{G}_2,m}[\mathsf{bad}]$

Corollary

$$|\Pr_{\mathsf{G}_1,m}[A] - \Pr_{\mathsf{G}_2,m}[A]| \leq \Pr_{\mathsf{G}_{1,2}}[\mathsf{bad}]$$

Example: ElGamal encryption

Game $\mathsf{ElGamal}$:
$(x, \alpha) \leftarrow \mathsf{KG}()$;
$(m_0, m_1) \leftarrow \mathcal{A}(\alpha)$;
$b \xleftarrow{\$} \{0, 1\}$;
$(\beta, \zeta) \leftarrow \mathsf{Enc}(\alpha, m_b)$;
$b' \leftarrow \mathcal{A}'(\alpha, \beta, \zeta)$;
$d \leftarrow b = b'$

(1)

```
inline_l KG.
inline_l Enc.
ep.
deadcode.
swap.
eqobs_in.
```

$\simeq_d$

Game $\mathsf{ElGamal}_0$:
$x \xleftarrow{\$} \mathbb{Z}_q$; $y \xleftarrow{\$} \mathbb{Z}_q$;
$(m_0, m_1) \leftarrow \mathcal{A}(g^x)$;
$b \xleftarrow{\$} \{0, 1\}$;
$\zeta \leftarrow g^{xy} \times m_b$;
$b' \leftarrow \mathcal{A}'(g^x, g^y, \zeta)$;
$d \leftarrow b = b'$

(2)

```
inline_r B.
ep.
deadcode.
eqobs_in.
```

$\simeq_d$

Game DDH_0 :
$x \xleftarrow{\$} \mathbb{Z}_q$;
$y \xleftarrow{\$} \mathbb{Z}_q$;
$d \leftarrow \mathcal{B}(g^x, g^y, g^{xy})$

Adversary $\mathcal{B}(\alpha, \beta, \gamma)$:
$(m_0, m_1) \leftarrow \mathcal{A}(\alpha)$;
$b \xleftarrow{\$} \{0, 1\}$;
$b' \leftarrow \mathcal{A}'(\alpha, \beta, \gamma \times m_b)$;
return $b = b'$

Game DDH_1 :
$x \xleftarrow{\$} \mathbb{Z}_q$;
$y \xleftarrow{\$} \mathbb{Z}_q$;
$z \xleftarrow{\$} \mathbb{Z}_q$;
$d \leftarrow \mathcal{B}(g^x, g^y, g^z)$

```
Lemma B_PPT : PPT B.
Proof. PPT_tac. Qed.
```

```
Lemma B_wf : WFAdv B.
Proof. ... Qed.
```

Game $\mathsf{ElGamal}_2$:
$x \xleftarrow{\$} \mathbb{Z}_q$; $y \xleftarrow{\$} \mathbb{Z}_q$;
$(m_0, m_1) \leftarrow \mathcal{A}(g^x)$;
$z \xleftarrow{\$} \mathbb{Z}_q$; $\zeta \leftarrow g^z$;
$b' \leftarrow \mathcal{A}'(g^x, g^y, \zeta)$;
$b \xleftarrow{\$} \{0, 1\}$;
$d \leftarrow b = b'$

(4)

```
swap.
eqobs_hd 4.
eqobs_tl 2.
apply mult_pad.
```

$\simeq_d$

Game $\mathsf{ElGamal}_1$:
$x \xleftarrow{\$} \mathbb{Z}_q$; $y \xleftarrow{\$} \mathbb{Z}_q$;
$(m_0, m_1) \leftarrow \mathcal{A}(g^x)$;
$b \xleftarrow{\$} \{0, 1\}$;
$z \xleftarrow{\$} \mathbb{Z}_q$; $\zeta \leftarrow g^z \times m_b$;
$b' \leftarrow \mathcal{A}'(g^x, g^y, \zeta)$;
$d \leftarrow b = b'$

(5)

```
inline_r B.
ep.
deadcode.
swap.
eqobs_in.
```

$\simeq_d$

$$\left|\Pr_{\mathsf{ElGamal}}[b = b'] - \frac{1}{2}\right| = \left|\Pr_{\mathsf{DDH}_0}[d] - \Pr_{\mathsf{DDH}_1}[d]\right| \leq \epsilon_{\mathsf{DDH}}$$

Example: ElGamal encryption

Game $\mathsf{ElGamal}_0$:
$x \xleftarrow{\$} \mathbb{Z}_q;\ y \xleftarrow{\$} \mathbb{Z}_q;$
$(m_0, m_1) \leftarrow \mathcal{A}(g^x);$
$b \xleftarrow{\$} \{0,1\};$
$\zeta \leftarrow g^{xy} \times m_b;$
$b' \leftarrow \mathcal{A}'(g^x, g^y, \zeta);$
$d \leftarrow b = b'$

```
inline_r B.
ep.
deadcode.
eqobs_in.
```

$\simeq_d$

Game DDH_0 :
$x \xleftarrow{\$} \mathbb{Z}_q;$
$y \xleftarrow{\$} \mathbb{Z}_q;$
$d \leftarrow \mathcal{B}(g^x, g^y, g^{xy})$

Adversary $\mathcal{B}(\alpha, \beta, \gamma)$:
$(m_0, m_1) \leftarrow \mathcal{A}(\alpha);$
$b \xleftarrow{\$} \{0,1\};$
$b' \leftarrow \mathcal{A}'(\alpha, \beta, \gamma \times m_b);$
return $b = b'$

Example: ElGamal encryption

$$
\begin{array}{|l|}
\hline
x \xleftarrow{\$} \mathbb{Z}_q;\ y \xleftarrow{\$} \mathbb{Z}_q; \\
(m_0, m_1) \leftarrow \mathcal{A}(g^x); \\
b \xleftarrow{\$} \{0,1\}; \\
\zeta \leftarrow g^{xy} \times m_b; \\
b' \leftarrow \mathcal{A}'(g^x, g^y, \zeta); \\
d \leftarrow b = b' \\
\\
\hline
\end{array}
$$

$$\simeq^{\emptyset}_{\{d\}}$$

$$
\begin{array}{|l|}
\hline
x \xleftarrow{\$} \mathbb{Z}_q; \\
y \xleftarrow{\$} \mathbb{Z}_q; \\
d \leftarrow \mathcal{B}(g^x, g^y, g^{xy}) \\
\\
\hline
\end{array}
$$

`Lemma foo:` $\vDash \mathsf{ElGamal}_0 \simeq^{\emptyset}_{\{d\}} \mathsf{DDH}_0$

`Proof.`

Example: ElGamal encryption

$$
\begin{array}{l}
x \xleftarrow{\$} \mathbb{Z}_q;\ y \xleftarrow{\$} \mathbb{Z}_q; \\
(m_0, m_1) \leftarrow \mathcal{A}(g^x); \\
b \xleftarrow{\$} \{0,1\}; \\
\zeta \leftarrow g^{xy} \times m_b; \\
b' \leftarrow \mathcal{A}'(g^x, g^y, \zeta); \\
d \leftarrow b = b'
\end{array}
$$

$$\simeq^{\emptyset}_{\{d\}}$$

$$
\begin{array}{l}
x \xleftarrow{\$} \mathbb{Z}_q; \\
y \xleftarrow{\$} \mathbb{Z}_q; \\
\alpha \leftarrow g^x;\ \beta \leftarrow g^y;\ \gamma \leftarrow g^{xy}; \\
(m_0, m_1) \leftarrow \mathcal{A}(\alpha); \\
b \xleftarrow{\$} \{0,1\}; \\
b' \leftarrow \mathcal{A}'(\alpha, \beta, \gamma \times m_b); \\
d \leftarrow b = b'
\end{array}
$$

```
Lemma foo: ⊨ ElGamal₀ ≃^∅_{d} DDH₀
Proof.
  inline_r B.
```

Example: ElGamal encryption

$$
\begin{array}{|l|}
\hline
x \xleftarrow{\$} \mathbb{Z}_q;\ y \xleftarrow{\$} \mathbb{Z}_q; \\
(m_0, m_1) \leftarrow \mathcal{A}(g^x); \\
b \xleftarrow{\$} \{0,1\}; \\
\zeta \leftarrow g^{xy} \times m_b; \\
b' \leftarrow \mathcal{A}'(g^x, g^y, g^{xy} \times m_b); \\
d \leftarrow b = b' \\
\hline
\end{array}
$$

$$\simeq^{\emptyset}_{\{d\}}$$

$$
\begin{array}{|l|}
\hline
x \xleftarrow{\$} \mathbb{Z}_q; \\
y \xleftarrow{\$} \mathbb{Z}_q; \\
\alpha \leftarrow g^x;\ \beta \leftarrow g^y;\ \gamma \leftarrow g^{xy}; \\
(m_0, m_1) \leftarrow \mathcal{A}(g^x); \\
b \xleftarrow{\$} \{0,1\}; \\
b' \leftarrow \mathcal{A}'(g^x, g^y, g^{xy} \times m_b); \\
d \leftarrow b = b' \\
\hline
\end{array}
$$

`Lemma foo:` $\models \mathsf{ElGamal}_0 \simeq^{\emptyset}_{\{d\}} \mathsf{DDH}_0$

```
Proof.
  inline_r B.
  ep.
```

Example: ElGamal encryption

$$
\begin{array}{l}
x \xleftarrow{\$} \mathbb{Z}_q;\ y \xleftarrow{\$} \mathbb{Z}_q; \\
(m_0, m_1) \leftarrow \mathcal{A}(g^x); \\
b \xleftarrow{\$} \{0,1\}; \\
\zeta \leftarrow g^{xy} \times m_b; \\
b' \leftarrow \mathcal{A}'(g^x, g^y, g^{xy} \times m_b); \\
d \leftarrow b = b'
\end{array}
$$

$$\simeq^{\emptyset}_{\{d\}}$$

$$
\begin{array}{l}
x \xleftarrow{\$} \mathbb{Z}_q; \\
y \xleftarrow{\$} \mathbb{Z}_q; \\
\alpha \leftarrow g^x;\ \beta \leftarrow g^y;\ \gamma \leftarrow g^{xy}; \\
(m_0, m_1) \leftarrow \mathcal{A}(g^x); \\
b \xleftarrow{\$} \{0,1\}; \\
b' \leftarrow \mathcal{A}'(g^x, g^y, g^{xy} \times m_b); \\
d \leftarrow b = b'
\end{array}
$$

`Lemma foo:` $\models \mathsf{ElGamal}_0 \simeq^{\emptyset}_{\{d\}} \mathsf{DDH}_0$

```
Proof.
  inline_r B.
  ep.
  deadcode.
```

Example: ElGamal encryption

$$
\begin{array}{|l|}
\hline
x \xleftarrow{\$} \mathbb{Z}_q;\ y \xleftarrow{\$} \mathbb{Z}_q; \\
(m_0, m_1) \leftarrow \mathcal{A}(g^x); \\
b \xleftarrow{\$} \{0,1\}; \\
b' \leftarrow \mathcal{A}'(g^x, g^y, g^{xy} \times m_b); \\
d \leftarrow b = b' \\
\hline
\end{array}
$$

$$\simeq^{\emptyset}_{\{d\}}$$

$$
\begin{array}{|l|}
\hline
x \xleftarrow{\$} \mathbb{Z}_q;\ y \xleftarrow{\$} \mathbb{Z}_q; \\
(m_0, m_1) \leftarrow \mathcal{A}(g^x); \\
b \xleftarrow{\$} \{0,1\}; \\
b' \leftarrow \mathcal{A}'(g^x, g^y, g^{xy} \times m_b); \\
d \leftarrow b = b' \\
\hline
\end{array}
$$

`Lemma foo:` $\models \mathsf{ElGamal}_0 \simeq^{\emptyset}_{\{d\}} \mathsf{DDH}_0$

```
Proof.
  inline_r B.
  ep.
  deadcode.
```

Example: ElGamal encryption

$$\begin{array}{l} x \xleftarrow{\$} \mathbb{Z}_q;\ y \xleftarrow{\$} \mathbb{Z}_q; \\ (m_0, m_1) \leftarrow \mathcal{A}(g^x); \\ b \xleftarrow{\$} \{0,1\}; \\ b' \leftarrow \mathcal{A}'(g^x, g^y, g^{xy} \times m_b); \\ d \leftarrow b = b' \end{array}$$

$$\simeq^{\emptyset}_{\{d\}}$$

$$\begin{array}{l} x \xleftarrow{\$} \mathbb{Z}_q;\ y \xleftarrow{\$} \mathbb{Z}_q; \\ (m_0, m_1) \leftarrow \mathcal{A}(g^x); \\ b \xleftarrow{\$} \{0,1\}; \\ b' \leftarrow \mathcal{A}'(g^x, g^y, g^{xy} \times m_b); \\ d \leftarrow b = b' \end{array}$$

`Lemma foo:` $\models \mathsf{ElGamal}_0 \simeq^{\emptyset}_{\{d\}} \mathsf{DDH}_0$

```
Proof.
  inline_r B.
  ep.
  deadcode.
  eqobs_in.
Qed.
```

$$\Pr_{\mathsf{ElGamal}_0, m}[b = b'] = \Pr_{\mathsf{DDH}_0, m}[b = b']$$

What does it take to trust a proof in CertiCrypt

Proof verification is fully-automated!
(but proof construction is still time-consuming)

- You need to..
  - trust the type checker of Coq
  - trust the definition of the semantics
  - make sure the final security statement ($\approx$ 1 line in Coq) is what you expect it to be
- You don't need to..
  - understand or even read the proof
  - trust proof tactics, program transformations
  - trust program logics, wp-calculus
  - be an expert in Coq

Wrapping up

Contributions

- Formal semantics of games
- Characterization of probabilistic polynomial-time programs
- Mechanization of common proof techniques
- Formalized emblematic proofs
  - PRP/PRF switching lemma
  - ElGamal
  - Hashed ElGamal (Random Oracle and standard model)
  - FDH (original and improved bound)
  - OAEP (IND-CPA)

To learn more about CertiCrypt

- *Formally certifying the security of digital signature schemes*
  IEEE Symposium on Security & Privacy, S&P 2009
- *Formal certification of code-based cryptographic proofs*
  ACM Symposium on Principles of Programming Languages, POPL 2009
- *Formal certification of ElGamal encryption. A gentle introduction to CertiCrypt*
  International Workshop on Formal Aspects in Security and Trust, FAST 2008

`www-sop.inria.fr/members/Santiago.Zanella/`

Questions

Future work

What's next?

- Overwhelming number of applications
  - OAEP (IND-CCA2)
  - Identity-based cryptography
  - Zero-knowledge protocols
  - 3DES, RSA-PSS, ...
- Computational soundness of symbolic proof methods
- Computational soundness of information flow type systems
- Beyond cryptography:
  Verification of randomized algorithms

Some statistics

- 7 persons involved. In chronological order:
  - Gilles Barthe (researcher)
  - Santiago Zanella (PhD student)
  - Benjamin Grégoire (researcher)
  - Romain Janvier (PostDoc)
  - Federico Olmedo (Intern)
  - Sylvain Heraud (PhD student)
  - Daniel Hedin (PostDoc)
- CertiCrypt: 30,000 lines of Coq / 50 man-months
- Full Domain Hash: 2,200 lines of Coq / 3 man-months (for a person without previous experience in CertiCrypt and unfamiliar with cryptography, let alone cryptographic proofs)

Characterizing well-formed adversaries

$$I \vdash \mathrm{nil} : I \qquad \frac{I \vdash i : I' \quad I' \vdash c : O}{I \vdash i;\ c : O}$$

$$\frac{\mathrm{Writable}(x) \quad \mathrm{fv}(e) \subseteq I}{I \vdash x \leftarrow e : I \cup \{x\}} \qquad \frac{\mathrm{Writable}(x) \quad \mathrm{fv}(d) \subseteq I}{I \vdash x \overset{\$}{\leftarrow} d : I \cup \{x\}}$$

$$\frac{\mathrm{fv}(e) \subseteq I \quad I \vdash c_i : O_i \quad i = 1,2}{I \vdash \mathrm{if}\ e\ \mathrm{then}\ c_1\ \mathrm{else}\ c_2 : O_1 \cap O_2} \qquad \frac{\mathrm{fv}(e) \subseteq I \quad I \vdash c : I}{I \vdash \mathrm{while}\ e\ \mathrm{do}\ c : I}$$

$$\frac{\mathrm{fv}(\vec{e}) \subseteq I \quad \mathrm{Writable}(x) \quad p \in \mathcal{O}}{I \vdash x \leftarrow p(\vec{e}) : I \cup \{x\}}$$

$$\frac{\mathrm{fv}(\vec{e}) \subseteq I \quad \mathrm{Writable}(x) \quad p \notin \mathcal{O} \quad \vdash_{\mathrm{wf}} p}{I \vdash x \leftarrow p(\vec{e}) : I \cup \{x\}}$$

$$\frac{\mathcal{V}_{\mathrm{rw}} \cup \mathcal{V}_{\mathrm{ro}} \cup \mathcal{A}.\mathrm{params} \vdash \mathcal{A}.\mathrm{body} : O \quad \mathrm{fv}(\mathcal{A}.\mathrm{re}) \subseteq O}{\vdash_{\mathrm{wf}} \mathcal{A}}$$

$$\mathrm{Writable}(x) \stackrel{\mathrm{def}}{=} \mathrm{Local}(x) \vee x \in \mathcal{V}_{\mathrm{rw}}$$

Characterizing well-formed adversaries

A type system for adversaries

If $\vdash_{\mathrm{wf}} \mathcal{A}$, then adversary $\mathcal{A}$...

- always initializes local variables before using them
- only writes global variables in $\mathcal{V}_{\mathrm{rw}}$
- only reads global variables in $\mathcal{V}_{\mathrm{rw}} \cup \mathcal{V}_{\mathrm{ro}}$
- may call oracles in $\mathcal{O}$
- may call a procedure not in $\mathcal{O}$, as long as it is itself a well-formed adversary

Observational equivalence

$$\models G_1 \sim G_2 : \Psi \Rightarrow \Phi \stackrel{\text{def}}{=} m_1\ \Psi\ m_2 \Rightarrow [\![G_1]\!]\ m_1 \sim_\Phi [\![G_2]\!]\ m_2$$

Lifting

$$\text{range } P\ \mu \stackrel{\text{def}}{=} \forall f,\ (\forall a,\ P\ a \Rightarrow f\ a = 0) \Rightarrow \mu\ f = 0$$

$$\mu_1 \sim_\Phi \mu_2 \stackrel{\text{def}}{=} \exists \mu,\ \pi_1(\mu) = \mu_1 \wedge \pi_2(\mu) = \mu_2 \wedge \text{range } \Phi\ \mu$$

Examples

- $\models x \xleftarrow{\$} \mathbb{Z}_q; \alpha \leftarrow g^x \times \beta \simeq_{\{\alpha\}} y \xleftarrow{\$} \mathbb{Z}_q; \alpha \leftarrow g^y$
- $\models x \xleftarrow{\$} \{0,1\}^k; y \leftarrow x \oplus z \simeq_{\{x,y,z\}}^{\{z\}} y \xleftarrow{\$} \{0,1\}^k; x \leftarrow y \oplus z$
- If f is a permutation,
  $\models x \xleftarrow{\$} \{0,1\}^{k-\rho}; y \xleftarrow{\$} \{0,1\}^{\rho}; z \leftarrow f(x \| y) \simeq_{\{z\}} z \xleftarrow{\$} \{0,1\}^k$

Small-step semantics

$$
\begin{aligned}
(\mathrm{nil}, m, [\,]) &\leadsto \mathrm{unit}\,(\mathrm{nil}, m, [\,]) \\
(\mathrm{nil}, m, (x, e, c, l) :: F) &\leadsto \mathrm{unit}\,(c, (l, m.\mathrm{glob})\{[\![e]\!]\ m/x\}, F) \\
(x \leftarrow p(\vec{e});\ c, m, F) &\leadsto \mathrm{unit}\,(E(p).\mathrm{body}, (\emptyset\{[\![\vec{e}]\!]\ m/E(p).\mathrm{params}\}, \\
(\mathrm{if}\ e\ \mathrm{then}\ c_1\ \mathrm{else}\ c_2;\ c, m, F) &\leadsto \mathrm{unit}\,(c_1;\ c, m, F) \\
&\qquad\qquad\qquad \text{if } [\![e]\!]\ m = \mathrm{true} \\
(\mathrm{if}\ e\ \mathrm{then}\ c_1\ \mathrm{else}\ c_2;\ c, m, F) &\leadsto \mathrm{unit}\,(c_2;\ c, m, F) \\
&\qquad\qquad\qquad \text{if } [\![e]\!]\ m = \mathrm{false} \\
(\mathrm{while}\ e\ \mathrm{do}\ c;\ c', m, F) &\leadsto \mathrm{unit}\,(c;\ \mathrm{while}\ e\ \mathrm{do}\ c;\ c', m, F) \\
&\qquad\qquad\qquad \text{if } [\![e]\!]\ m = \mathrm{true} \\
(\mathrm{while}\ e\ \mathrm{do}\ c;\ c', m, F) &\leadsto \mathrm{unit}\,(c', m, F) \\
&\qquad\qquad\qquad \text{if } [\![e]\!]\ m = \mathrm{false}
\end{aligned}
$$

$$
\begin{aligned}
(x \leftarrow e;\ c, m, F) &\leadsto \mathrm{unit}\,(c, m\{[\![e]\!]\ m/x\}, F) \\
(x \xleftarrow{\$} d;\ c, m, F) &\leadsto \mathrm{bind}\,([\![d]\!]\ m)(\lambda v.\ \mathrm{unit}\,(c, m\{v/x\}, F))
\end{aligned}
$$

Denotation

$$[\![S]\!]_0 \stackrel{\text{def}}{=} \text{unit } S \qquad [\![S]\!]_{n+1} \stackrel{\text{def}}{=} \text{bind } [\![S]\!]_n \ [\![\cdot]\!]^1$$

$$[\![c]\!] \ m : M(\mathcal{M}) \stackrel{\text{def}}{=} \lambda f.\ \sup \{[\![(c, m, [\,])]\!]_n \ f|_{\text{final}} \mid n \in \mathbb{N}\}$$

Existential unforgeability of FDH

Game G_{EF} : $\boldsymbol{L}, \boldsymbol{S} \leftarrow [\ ];$ $(m, \sigma) \leftarrow \mathcal{A}();$ $h \leftarrow H(m)$	$H(m) \stackrel{\text{def}}{=}$ if $m \notin \mathrm{dom}(\boldsymbol{L})$ then $h \stackrel{\$}{\leftarrow} \mathcal{G}; \boldsymbol{L} \leftarrow (m, h) :: \boldsymbol{L}$ return $\boldsymbol{L}(m)$ $\mathsf{Sign}(m) \stackrel{\text{def}}{=}$ $\boldsymbol{S} \leftarrow m :: \boldsymbol{S}; h \leftarrow H(m);$ return $f^{-1}(h)$

Consider an adversary $\mathcal{A}$ s.t.

- $\mathcal{A}$ makes at most $q_{\mathsf{H}}(k)$ hash queries
- $\mathcal{A}$ makes at most $q_{\mathsf{S}}(k)$ signature queries

Suppose

- $\mathcal{A}$ runs within time $t(k)$
- $\mathcal{A}$ forges a signature with probability $\epsilon(k)$
  i.e. $\epsilon(k) = \Pr_{\mathsf{G}_{\mathsf{EF}}}[h = f(\sigma)]$

Existential unforgeability of FDH

Theorem (Original bound)

There exists an $\mathcal{I}$ that inverts f with probability $\epsilon'(k)$ within time $t'(k)$, where

$$\epsilon'(k) \geq (q_{\mathsf{H}}(k) + q_{\mathsf{S}}(k) + 1)^{-1}\, \epsilon(k)$$

$$t'(k) \leq t(k) + (q_{\mathsf{H}}(k) + q_{\mathsf{S}}(k))\, \Theta(T_f)$$

Existential unforgeability of FDH

Theorem (Coron's optimal bound)

There exists an $\mathcal{I}$ that inverts f with probability $\epsilon'(k)$ within time $t'(k)$, where

$$\begin{aligned}
\epsilon'(k) &\geq \frac{1}{q_{\mathrm{S}}(k)+1}\left(1-\frac{1}{q_{\mathrm{S}}(k)+1}\right)^{q_{\mathrm{S}}(k)} \epsilon(k) \\
&\approx \exp(-1) q_{\mathrm{S}}(k)^{-1} \epsilon(k) \\
t'(k) &\leq t(k) + (q_{\mathrm{H}}(k) + q_{\mathrm{S}}(k))\, \Theta(T_f)
\end{aligned}$$