Towards Language-Based Cryptographic Proofs

Gilles Barthe Benjamin Grégoire Romain Janvier
Santiago Zanella Béguelin

INRIA Sophia Antipolis
INRIA-Microsoft Research Joint Centre

2007.10.04

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, EuroCrypt 2006

What's wrong with cryptographic proofs

Increasing complexity in cryptographic proofs

+

Unmanageable numbers of them appearing in articles

+

No one willing to verify boring, repetitive, handmade proofs

Subtle errors in supposedly peer-reviewed cryptographic proofs

What's wrong with cryptographic proofs

Increasing complexity in cryptographic proofs

+

Unmanageable numbers of them appearing in articles

+

No one willing to verify boring, repetitive, handmade proofs

Subtle errors in supposedly peer-reviewed cryptographic proofs

What's wrong with cryptographic proofs

Increasing complexity in cryptographic proofs

$+$

Unmanageable numbers of them appearing in articles

$+$

No one willing to verify boring, repetitive, handmade proofs

Subtle errors in supposedly peer-reviewed cryptographic proofs

What's wrong with cryptographic proofs

Increasing complexity in cryptographic proofs

$+$

Unmanageable numbers of them appearing in articles

$+$

No one willing to verify boring, repetitive, handmade proofs

Subtle errors in supposedly peer-reviewed cryptographic proofs

Three authoritative opinions

- *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

The game-playing technique

The general idea

- Describe security using a game played between a challenger and an adversary. May be encoded as a program in a probabilistic programming language,
- Pick an initial game, transform it stepwise preserving (up to a negligible factor) or increasing the winning probability of the adversary,
- Bound this probability in the final game.
- Argue that the bound also holds for the initial game
- For all this, rely on a well-defined set of hypotheses (e.g. Decisional Diffie-Hellman) and properties of primitives (Ideal-cipher, one-way function)

The game-playing technique

The general idea

- Describe security using a game played between a challenger and an adversary. May be encoded as a program in a probabilistic programming language,
- Pick an initial game, transform it stepwise preserving (up to a negligible factor) or increasing the winning probability of the adversary,
- Bound this probability in the final game.
- Argue that the bound also holds for the initial game
- For all this, rely on a well-defined set of hypotheses (e.g. Decisional Diffie-Hellman) and properties of primitives (Ideal-cipher, one-way function)

Caveat: Game-playing doesn't substitute probabilistic reasoning but supplements it.

Goals and rationale

Our objective is to build a certified tool for checking game-playing proofs, on top of a general purpose proof assistant (Coq)

- The tool provides independently checkable certificates that justify transitions between games
- Security goals, properties and hypotheses are explicit. The latter can be taken from a standard library.
- The "mundane" and "innovative" parts of the proofs can be justified formally in a unified formalism.

Disclaimer: we are (currently) not interested in

- Discovering the sequence of games,
- user interface

A probabilistic WHILE programming language

$$
\begin{aligned}
\mathcal{C} \ni c \quad ::= \ & \mathsf{skip} \\
\mid \ & x \leftarrow e \\
\mid \ & x \xleftarrow{\$} \Delta \\
\mid \ & \mathsf{while}\ e\ \mathsf{do}\ c \\
\mid \ & \mathsf{if}\ e\ \mathsf{then}\ c_1\ \mathsf{else}\ c_2 \\
\mid \ & c1\ ;\ c2 \\
\mid \ & x \leftarrow p(e_1, e_2, \ldots)
\end{aligned}
$$

Denotational semantics

Our semantics maps a command c and an initial state σ to the expected value operator over the distribution of states where the execution c halts starting from σ

$$[\![\cdot]\!] : \mathcal{C} \to \mathcal{S} \to (\mathcal{S} \to [0,1]) \to [0,1]$$

Intuitevely,

$$[\![c]\!]\ \sigma\ f = \sum_{\sigma' \in \mathcal{S}} f(\sigma')\ \mathrm{Pr}[\langle c, \sigma \rangle \downarrow \sigma']$$

Instead of defining the semantic function directly, we rely on a frame-based small-step semantics.

Small-step semantics

We define $\mathcal{D}_A = (A \to [0,1]) \to [0,1]$.
$[\![\cdot]\!]_1 : \mathcal{C} \to \mathcal{S} \to \mathcal{D}_\mathcal{S}$ is the frame-based small-step semantics
$[\![\cdot]\!]_n : \mathcal{C} \to \mathcal{S} \to \mathcal{D}_\mathcal{S}$ is the n-unfold of $[\![\cdot]\!]_1$
$[\![\cdot]\!] : \mathcal{C} \to \mathcal{M} \to \mathcal{D}_\mathcal{M}$ is defined as the LUB of $[\![\cdot]\!]_n$, measuring the function on memories of final configurations reachable in at most n steps.

$$[\![c]\!]\ \mu\ f = \text{lub}\ (\lambda n \cdot [\![c]\!]_n\ \mu\ f!)$$

Where $f!\ \sigma'$ takes the value of f on the memory of σ' if it is a final configuration and 0 otherwise.

The least upper bound is guaranteed to exist and corresponds to the limit when restricted to monotone sequences.

Since $[\![c]\!]_n\ \mu\ !f$ is increasing, the semantics is well defined.

- Game-playing cryptographic proofs try to bound the winning probability of an adversary often by proving indistinguishability between a scheme and an ideal version of it. Program equivalence is key for this kind of proofs
- Our definition of program equivalence satisfies congruence properties that allow to relate two programs under different contexts.
- Although our definition is semantical, we derive syntactic criteria for deciding program equivalence and prove them correct wrt the semantical definition.

Observational equivalence

Definition (Indistinguishable functions)

We say that two functions $f, g : \mathcal{M} \to A$ are indistinguishable wrt a relation $R \subseteq \mathcal{M} \times \mathcal{M}$ and denote it as $f \simeq_R g$ iff

$$\forall (\mu_1, \mu_2) \in R \cdot f \ \mu_1 = g \ \mu_2$$

Definition (Observational equivalence)

Let $P, Q \subseteq \mathcal{M} \times \mathcal{M}$ be a PER over memories, we say that c_1 is observational equivalent to c_2 wrt to the input relation P and the output relation Q and denote it $c_1 \simeq_P^Q c_2$ iff,

$$\begin{array}{c} \forall (\mu_1, \mu_2) \in P; f, g \in \mathcal{M} \to [0,1] \cdot \\ f \simeq_Q g \Rightarrow [\![c_1]\!] \ \mu_1 \ f = [\![c_2]\!] \ \mu_2 \ g \end{array}$$

INRIA

Observational equivalence properties

$$\frac{c_1 \simeq_P^Q c_2}{c_2 \simeq_P^Q c_1}\ \text{sym} \qquad \frac{c_1 \simeq_P^Q c_2 \quad c_2 \simeq_P^Q c_3}{c_1 \simeq_P^Q c_3}\ \text{trans}$$

$$\frac{c_1 \simeq_P^Q c_2 \quad P' \subseteq P}{c_1 \simeq_{P'}^Q c_2}\ \text{str} \qquad \frac{c_1 \simeq_P^Q c_2 \quad Q \subseteq Q'}{c_1 \simeq_P^{Q'} c_2}\ \text{weak}$$

$$\frac{c_1 \simeq_P^Q c_1' \quad c_2 \simeq_Q^R c_2'}{c_1; c_2 \simeq_P^R c_1'; c_2'}\ \text{seq}$$

$$\frac{c_1 \simeq_{P|_e}^Q c_1' \quad c_2 \simeq_{P|_{\neg e}}^Q c_2' \quad [\![e]\!] \simeq_P [\![e']\!]}{\text{if } e \text{ then } c_1 \text{ else } c_2 \simeq_P^Q \text{if } e' \text{ then } c_1' \text{ else } c_2'}\ \text{cond}$$

...

INRIA

Provable game transformations

- Algebraic manipulations
  - substitute $s_1 \xleftarrow{\$} \{0,1\}^n; s_2 \leftarrow s_1 \oplus t$ for $s_2 \xleftarrow{\$} \{0,1\}^n; s_1 \leftarrow s_2 \oplus t$
  - substitute $h_1 \leftarrow g^{u_1}; h_2 \leftarrow h_1^{u_2}$ for $h_1 \leftarrow g^{u_1}; h_2 \leftarrow g^{u_1 u_2}$
- Code motion
- Constant propagation
- Dead-code elimination
- Inlining of procedure calls
- Equivalent-until-failure games
- Derandomization
  - replace $x \xleftarrow{\$} t; c$ with $x \leftarrow v; c$ where v maximizes over t the probability of a failure event

INRIA

IND-CPA security of an asymmetric encryption scheme

Definition (Assymetric encryption scheme)

A triple of algorithms $(\mathcal{K}, \mathcal{E}, \mathcal{D})$

$$
\begin{array}{lcll}
\mathcal{K}_\eta & : & \text{Coins} \to \text{Key} \times \text{Key} & \text{Key generation} \\
\mathcal{E} & : & \text{Key} \times \text{Plaintext} \times \text{Coins} \to \text{Ciphertext} & \text{Encryption} \\
\mathcal{D} & : & \text{Key} \times \text{Ciphertext} \to \text{Plaintext} & \text{Decryption}
\end{array}
$$

where $\forall (pk, sk) = \mathcal{K}_\eta(r), m, \phi = \mathcal{E}(pk, m) \Rightarrow m = \mathcal{D}(sk, \phi)$

A game-playing proof of IND-CPA for an asymmetric encryption scheme begins with a game like

$$
\begin{array}{l}
(pk, sk) \leftarrow \mathcal{K}_\eta(); (m_0, m_1) \leftarrow A_1(pk); \\
b \stackrel{\$}{\leftarrow} \{0, 1\}; \phi \leftarrow \mathcal{E}(pk, m_b); \\
\hat{b} \leftarrow A_2(m_0, m_1, pk, \phi)
\end{array}
$$

If the probability of the event $\hat{b} = b$ after the execution can be bound by a *negligible* function of η, the game is IND-CPA secure.

Toy example: semantic security of ElGamal encryption

Theorem: $\forall$ polynomial adversaries $\mathcal{A}$, $\mathcal{A}'$ (sharing state), if the DDH problem is hard for the chosen group family, then

$\mathsf{ElGamal}_0 \stackrel{\mathrm{def}}{=}$		$\mathsf{ElGamal}_1 \stackrel{\mathrm{def}}{=}$
$x \stackrel{\$}{\leftarrow} [0..\eta];\ \alpha \leftarrow \gamma^x;$		$x \stackrel{\$}{\leftarrow} [0..\eta];\ \alpha \leftarrow \gamma^x;$
$(m_0, m_1) \leftarrow \mathcal{A}(\alpha);$		$(m_0, m_1) \leftarrow \mathcal{A}(\alpha);$
$y \stackrel{\$}{\leftarrow} [0..\eta];\ \beta \leftarrow \gamma^y;$	$\approx^{\eta}_{[d=1]}$	$y \stackrel{\$}{\leftarrow} [0..\eta];\ \beta \leftarrow \gamma^y;$
$\delta \leftarrow \alpha^y;$		$\delta \leftarrow \alpha^y;$
$\zeta \leftarrow \delta \times m_0;$		$\zeta \leftarrow \delta \times m_1;$
$d \leftarrow \mathcal{A}'(\alpha, \beta, \zeta);$		$d \leftarrow \mathcal{A}'(\alpha, \beta, \zeta)$

DDH assumption: it's hard to distinguish $(\gamma^x, \gamma^y, \gamma^{xy})$ from $(\gamma^x, \gamma^y, \gamma^z)$ (x, y, z uniformly sampled in $[0..\eta]$).

Proof. (as a sequence of games)

$$\begin{array}{l}
\mathsf{ElGamal}_0 \stackrel{\text{def}}{=} \\
x \stackrel{\$}{\leftarrow} [0..\eta];\ \alpha \leftarrow \gamma^x \\
(m_0, m_1) \leftarrow \mathcal{A}(\alpha); \\
y \stackrel{\$}{\leftarrow} [0..\eta];\ \beta \leftarrow \gamma^y; \\
\delta \leftarrow \alpha^y; \\
\zeta \leftarrow \delta \times m_0; \\
d \leftarrow \mathcal{A}'(\alpha, \beta, \zeta);
\end{array}
\qquad \simeq \qquad
\begin{array}{l|l}
\mathsf{DDH}_l \stackrel{\text{def}}{=} & \mathcal{B}(\alpha, beta, \delta) \stackrel{\text{def}}{=} \\
x \stackrel{\$}{\leftarrow} [0..\eta];\ \alpha \leftarrow \gamma^x & (m_0, m_1) \leftarrow \mathcal{A}(\alpha); \\
y \stackrel{\$}{\leftarrow} [0..\eta];\ \beta \leftarrow \gamma^y; & \zeta \leftarrow \delta \times m_0; \\
\delta \leftarrow \alpha^y; & d \leftarrow \mathcal{A}'(\alpha, \beta, \zeta); \\
d \leftarrow \mathcal{B}(\alpha, \beta, \delta) & \text{return } d
\end{array}$$

```
simplify
code_motion
simplify
inline B; simplify
```

Semantic security of ElGamal encryption

DDH assumption: it's hard to distinguish $(\gamma^x, \gamma^y, \gamma^{xy})$ from $(\gamma^x, \gamma^y, \gamma^z)$ (x, y, z uniformly sampled in $[0..\eta]$).

$$\begin{array}{l} \mathsf{DDH}_l \stackrel{\text{def}}{=} \\ x \stackrel{\$}{\leftarrow} [0..\eta];\ \alpha \leftarrow \gamma^x \\ y \stackrel{\$}{\leftarrow} [0..\eta];\ \beta \leftarrow \gamma^y; \\ \delta \leftarrow \alpha^y; \\ d \leftarrow \mathcal{B}(\alpha, \beta, \delta) \end{array} \quad \approx^{\eta}_{[d=1]} \quad \begin{array}{l} \mathsf{DDH}_r \stackrel{\text{def}}{=} \\ x \stackrel{\$}{\leftarrow} [0..\eta];\ \alpha \leftarrow \gamma^x \\ y \stackrel{\$}{\leftarrow} [0..\eta];\ \beta \leftarrow \gamma^y; \\ z \stackrel{\$}{\leftarrow} [0..\eta]; \delta \leftarrow \gamma^z \\ d \leftarrow \mathcal{B}(\alpha, \beta, \delta) \end{array}$$

`apply DDH_assumption`
Proof that $\mathcal{B}$ is polynomial if $\mathcal{A}$ and *$\mathcal{A}'$ are so*

Semantic security of ElGamal encryption

$\mathsf{DDH}_r \stackrel{\mathrm{def}}{=}$
$x \stackrel{\$}{\leftarrow} [0..\eta];\ \alpha \leftarrow \gamma^x;$
$y \stackrel{\$}{\leftarrow} [0..\eta];\ \beta \leftarrow \gamma^y;$
$z \stackrel{\$}{\leftarrow} [0..\eta];\ \delta \leftarrow \gamma^z;$
$d \leftarrow \mathcal{B}(\alpha, \beta, \delta)$

$\simeq$

$\mathsf{ElGamal}_0^1 \stackrel{\mathrm{def}}{=}$
$x \stackrel{\$}{\leftarrow} [0..\eta];\ \alpha \leftarrow \gamma^x;$
$(m_0, m_1) \leftarrow \mathcal{A}(\alpha);$
$y \stackrel{\$}{\leftarrow} [0..\eta];\ \beta \leftarrow \gamma^y;$
$z \stackrel{\$}{\leftarrow} [0..\eta];\ \delta \leftarrow \gamma^z;$
$\zeta \leftarrow \delta \times m_0;$
$d \leftarrow \mathcal{A}'(\alpha, \beta, \zeta)$

```
simplify
code_motion
simplify
inline B; simplify
```

Semantic security of ElGamal encryption

$$\begin{array}{l} \mathsf{ElGamal}_0^1 \stackrel{\mathrm{def}}{=} \\ x \stackrel{\$}{\leftarrow} [0..\eta];\ \alpha \leftarrow \gamma^x; \\ (m_0, m_1) \leftarrow \mathcal{A}(\alpha); \\ y \stackrel{\$}{\leftarrow} [0..\eta];\ \beta \leftarrow \gamma^y; \\ z \stackrel{\$}{\leftarrow} [0..\eta];\ \delta \leftarrow \gamma^z; \\ \zeta \leftarrow \delta \times m_0; \\ d \leftarrow \mathcal{A}'(\alpha, \beta, \zeta) \end{array} \quad \simeq \quad \begin{array}{l} \mathsf{ElGamal}^2 \stackrel{\mathrm{def}}{=} \\ x \stackrel{\$}{\leftarrow} [0..\eta];\ \alpha \leftarrow \gamma^x; \\ (m_0, m_1) \leftarrow \mathcal{A}(\alpha); \\ y \stackrel{\$}{\leftarrow} [0..\eta];\ \beta \leftarrow \gamma^y; \\ z \stackrel{\$}{\leftarrow} [0..\eta];\ \zeta \leftarrow \gamma^z; \\ d \leftarrow \mathcal{A}'(\alpha, \beta, \zeta) \end{array}$$

```
simplify_head 6
simplify_tail
apply mult_pad
```

$\texttt{mult\_pad} : \forall\ a, b, c, d\ \cdot\ (a \stackrel{\$}{\leftarrow} [0..\eta]; b \leftarrow \gamma^a; c \leftarrow b \times d) \simeq_c (a \stackrel{\$}{\leftarrow} [0..\eta]; c \leftarrow \gamma^a)$

Semantic security of ElGamal encryption

Thus, we have

$$\mathsf{ElGamal}_0 \simeq \mathsf{DDH}_l \approx^{\eta}_{[d=1]} \mathsf{DDH}_r \simeq \mathsf{ElGamal}^1_0 \simeq \mathsf{ElGamal}^2$$

which implies that

$$\mathsf{ElGamal}_0 \approx^{\eta}_{[d=1]} \mathsf{ElGamal}^2$$

Symetrically, $\mathsf{ElGamal}_1 \approx^{\eta}_{[d=1]} \mathsf{ElGamal}^2$ and therefore

$$\mathsf{ElGamal}_0 \approx^{\eta}_{[d=1]} \mathsf{ElGamal}_1$$

Q.E.D.

Summary

So far, formalized in Coq (20k lines)

- Semantics of a probabilistic programming language
- Theory of program equivalence
- Reflective tactics for performing common transformations
- A proof of ElGamal IND-CPA security
- A significant part of the proof of OAEP IND-CPA security
- Preliminary asymptotic version of the PRP/PRF switching lemma

Disclaimer

- Semantics of groups and bitstrings is axiomatized
- For the time being, we (almost) avoid complexity issues
- We do not have a complete proof of OAEP semantic security

Prospective applications

- computational soundness of an information flow type system.
- verification of randomized algorithms in general

Equivalent IND-CPA definitions

$$
\begin{aligned}
\mathsf{Adv}_G^A &\stackrel{\text{def}}{=} \left|\Pr[G_A^b \to b] - \tfrac{1}{2}\right| \\
&= \left|\Pr[G_A^b \to b \wedge b = 0] + \Pr[G_A^b \to b \wedge b = 1] - \tfrac{1}{2}\right| \\
&= \Big|\Pr[G_A^b \to b | b = 0]\Pr[b = 0] + \\
&\qquad\quad \Pr[G_A^b \to b | b = 1] Pr[b = 1] - \tfrac{1}{2}\Big| \\
&= \left|\Pr[G_A^0 \to 0]\tfrac{1}{2} + \Pr[G_A^1 \to 1]\tfrac{1}{2} - \tfrac{1}{2}\right| \\
&= \left|(1 - \Pr[G_A^0 \to 1])\tfrac{1}{2} + \Pr[G_A^1 \to 1]\tfrac{1}{2} - \tfrac{1}{2}\right| \\
&= \tfrac{1}{2}\left|\Pr[G_A^0 \to 1] - \Pr[G_A^1 \to 1]\right|
\end{aligned}
$$