# EasyCrypt - Lecture 1 - Introduction

Gilles Barthe

Monday November 24th

# Lecture 1: Introduction

- Motivation
- Background
- Approach
- Example

# Problems with cryptographic proofs

Proofs are error-prone and flawed

- *In our opinion, many proofs in cryptography have become essentially unverifiable. Our field may be approaching a crisis of rigor*. Bellare and 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)*. Halevi, 2005

Gap between algorithms, implementations and machine code

- *Omitting one fine-grained detail from a formal analysis can have a large effect on how that analysis applies in practice.* Degabriele, Paterson, and Watson, 2011
- *Real-world crypto is breakable; is in fact being broken; is one ongoing disaster area in security*. Bernstein, 2013

# The code-based game-playing approach

- Provable security
  - mathematical approach to cryptography (inspired from complexity theory)
  - reductionist proofs: construction is asymptotically secure if some computational problem is asymptotically hard
- Concrete provable security
  - for every adversary that breaks construction with probability $p$ in time $t$, there exists an adversary that breaks computational assumption with probability $p'$ in time $t'$
- Code-based approach: probabilistic programs everywhere!
- Game-playing approach: sequences of simple steps
- Really? Many steps are not so simple
  - uniformly distributed and independent from adversary's view
  - compiler optimizations
  - implicit invariants

# Computer-aided cryptographic proofs

Halevi (2005) outlines the design of a tool for crypto proofs

- code-based approach
- reasoning principles

Many good ideas, but

- no implementation
- limited generality (example driven)
- weak guarantees (no foundations, large TCB)

Ideally:

- solid foundations
- full and independently verifiable proofs
- practical and accessible to cryptographers

# Cryptographic proofs as program verification

Program verification:

- achieves practicality and verifiability
- is supported by solid foundations and tools

But:

- programs are probabilistic
- verification goals over multiple programs
  - verify multiple programs
  - programs are often verified pairwise
    (crypto: reductionist argument, PL: program equivalence)
- program verification is only part of the story:
  - mathematical reasoning: algebra, arithmetic, ...
  - (discrete) probabilities
  - proofs by induction

# Quest

- define a verification paradigm for provable security

  provable security
  =
  relational verification of parameterized probabilistic programs
- compact proofs that adhere to cryptographic practice
  - mundane steps fully automated
  - users drive proofs by applying high-level principles
  - support for mathematical reasoning
- narrow the gap between provable sec. and “real-world”
  - reference implementations
  - executable code

  covering a.o.t. side-channels
- leverage existing verification techniques and tools
  - SMT solvers, theorem provers, computer algebra systems, etc
  - program logics, VC generation, invariant generation, etc
  - verified compilers, certifying compilers, etc

# EasyCrypt

- Verification framework that integrates functionalities of
  - an interactive proof assistant
  - program verification environment

  and achieves automation via SMT and CAS back-ends
- Focused on cryptography
  - discrete probabilities
  - structured proofs
    - composition/instantiation
    - universal and existential quantification over modules
  - ultimately: complexity analysis
- Generic
  - primitives and protocols
  - game-based and simulation-based
  - random oracle, standard model, etc.
- Nexus of a larger system

# Foundations

- Program logics (some overlaps, many interplays)
  - probabilistic relational Hoare logic (pRHL)

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

    Assertions are written in predicate logic (no probabilities)
  - probabilistic Hoare logic (pHL)

    $$\{P\}\ c\ \{Q\} \bowtie \delta$$

- Program transformations
  - constant folding, loop optimizations, etc
- Soundness w.r.t. set-theoretical semantics
- Program logics are embedded in ambient logic
  - hybrid arguments, generic arguments

# Programs

- User-extensible expression language
  - sub-distributions, higher-order functions, inductive types, etc.
- Imperative language

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

  - procedures can be left abstract

# Semantics

- The set distr $A$ of discrete sub-distributions over $A$ is the set of functions $\mu : A \to [0,1]$ such that:
  - $\mathsf{supp}(\mu) = \{a \in A \mid \mu\ a > 0\}$ is discrete
  - $\sum_{a \in A} \mu\ a \leq 1$
- The probability of $E \subseteq A$ in $\mu \in \mathsf{distr}\ A$ is defined as

$$\sum_{a \in E} \mu\ a$$

- Commands are interpreted as maps from memories to sub-distribution on memories

$$[\![c]\!] : \mathcal{M} \to \mathsf{distr}\ \mathcal{M}$$

- Kleisli operator

$$\cdot^{\sharp} : (A \to \mathsf{distr}\ B) \to (\mathsf{distr}\ A \to \mathsf{distr}\ B)$$

## pRHL: intuition and preview

- $\{P\}\ c_1 \sim c_2\ \{Q\}$ is valid iff for all $m_1, m_2 \in \mathcal{M}$, $P\ m_1\ m_2$ implies

$$\mathcal{L}(Q)\ [\![c_1]\!]_{m_1}\ [\![c_2]\!]_{m_2}$$

- If $\{P\}\ c_1 \sim c_2\ \{A_{\langle 1\rangle} \Leftrightarrow B_{\langle 2\rangle}\}$ is valid then for all $m_1, m_2 \in \mathcal{M}$, $P\ m_1\ m_2$ implies

$$\Pr[c_1, m_1 : A] = \Pr[c_2, m_2 : B]$$

- If $\{P\}\ c_1 \sim c_2\ \{A_{\langle 1\rangle} \Rightarrow B_{\langle 2\rangle}\}$ is valid then for all $m_1, m_2 \in \mathcal{M}$, $P\ m_1\ m_2$ implies

$$\Pr[c_1, m_1 : A] \leq \Pr[c_2, m_2 : B]$$

- If $\{P\}\ c_1 \sim c_2\ \{\neg F_{\langle 2\rangle} \Rightarrow A_{\langle 1\rangle} \Rightarrow B_{\langle 2\rangle}\}$ is valid then for all $m_1, m_2 \in \mathcal{M}$, $P\ m_1\ m_2$ implies

$$\Pr[c_1, m_1 : A] - \Pr[c_2, m_2 : B] \leq \Pr[c_2, m_2 : F]$$

# Public-key encryption

Algorithms $(\mathcal{K}, \mathcal{E}, \mathcal{D})$, s.t.:

- $\mathcal{E}$ takes as inputs a public key and a message, and outputs a ciphertext
- $\mathcal{D}$ takes as inputs a secret key and a ciphertext, and outputs a plaintext
- if $(sk, pk)$ is a valid key pair, $\mathcal{D}_{sk}(\mathcal{E}_{pk}(m)) = m$

---

```
module type Scheme = {
  proc kg() : pkey * skey
  proc enc(pk:pkey, m:plain)  : cipher
  proc dec(sk:skey, c:cipher) : plain
}.
```

---

(A bit of cheating here: $\mathcal{D}$ may be partial)

# Correctness

```
module Correct (S:Scheme) = {
  proc main(m:plain) : bool = {
    var pk : pkey;
    var sk : skey;
    var c  : cipher;
    var m' : plain;
    (pk, sk) = S.kg();
    c        = S.enc(pk, m);
    m'       = S.dec(sk, c);
    return (m' = m);
  }
}.
```

Correctness states that m'=m holds with probability 1.

# Adversary

```
module type Adv = {
  proc choose (pk:pkey) : msg * msg
  proc guess (c:cipher) : bool
}.
```

Many reasonings require that adversaries are lossless
(i.e. terminate with probability 1)

# Indistinguishability

```
module CPA (S:Scheme, A:Adversary) = {
  proc main() : bool = {
    var pk, sk, m0, m1, c, b, b';

    (pk, sk)  = S.kg();
    (m0, m1) = A.choose(pk);
    b         = ${0,1};
    c         = S.enc(pk, b ? m1 : m0);
    b'        = A.guess(c);
    return (b' = b);
  }
}.
```

Want that probability of b'=b is close to $\frac{1}{2}$

# One-way trapdoor permutations

```
module type Inverter = {
 proc i(pk : pkey, y : rand) : rand
}.

module OW(F:TP, I :Inverter) ={
 proc main() : bool ={
  var x, x', pk, sk;

  x = $uniform_rand;
  (pk,sk) = KG.();
  x'  = I.i(pk,(f pk x));
  return (x' = x);
 }
}.
```

Assume that probability of x'=x is small

# Random oracles (excerpts)

```
module type Oracle = {
  proc init():unit
  proc o(x:from):to
}.

theory ROM.
  module RO:Oracle = {
    var m : (from, to) map

    proc o(x:from) : to = {
      var y : to;
      y = $uniform_to;
      if (!mem x (dom m)) m.[x] = y;
      return (m.[x]);
    }
  }.
```

# Example: Bellare and Rogaway 1993 encryption

- plain $= \{0,1\}^n$ (bitstrings of length $n$)
- rand $= \{0,1\}^k$ (bitstrings of length $k$)
- cipher $= \{0,1\}^{n+k}$ (bitstrings of length $n+k$)

```
proc enc(pk:pkey, m:plain): cipher = {
  var h, s : plain;
  var r : rand;

  r  = ${0,1}^k;
  h  = H.o(r);
  s  = m ⊕ h;
  return ((f pk r) || s);
}
```

# Security

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

$$\Pr[\mathsf{CPA}(\mathsf{BR}, \mathcal{A}) : b' = b] - \frac{1}{2} \leq \Pr[\mathsf{OW}(\mathsf{F}, \mathcal{I}) : x' = x]$$

In the next slides, we will adopt some shorthands:

- CPA for $\mathsf{CPA}(\mathsf{BR}, \mathcal{A})$
- OW for $\mathsf{OW}(\mathsf{F}, \mathcal{I})$

# Proof

## Game hopping technique

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

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

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

**Game** OW :
$(sk, pk) = \mathcal{K}()$;
$y = \$\{0,1\}^\ell$;
$y' = \mathcal{I}(pk, f_{pk}(y))$;
return $(y' = y)$;
**Adversary** $\mathcal{I}(p, x)$ :
$(m_0, m_1) = \mathcal{A}_1(p)$;
$s = \$\{0,1\}^k$;
$c^\star = x \parallel s$;
$b' = \mathcal{A}_2(c^\star)$;
$y' = [z \in \mathbf{L}_H^{\mathcal{A}} \mid f_p(z) = x]$;
return $y'$

1. For each hop
   - prove validity of pRHL judgment
   - derive probability claim(s)
2. Obtain security bound by combining claims
3. Check execution time of constructed adversary

# Conditional equivalence

$$\begin{array}{|l|}\hline \mathcal{E}_{pk}(m): \\ \quad r = \$\{0,1\}^{\ell}; \\ \quad h = H(r); \\ \quad s = h \oplus m; \\ \quad c = f_{pk}(r) \parallel s; \\ \quad \text{return } c; \\ \hline \end{array} \qquad \rhd \qquad \begin{array}{|l|}\hline \mathcal{E}_{pk}(m): \\ \quad r = \$\{0,1\}^{\ell}; \\ \quad h = \$\{0,1\}^{k}; \\ \quad s = h \oplus m; \\ \quad c = f_{pk}(r) \parallel s; \\ \quad \text{return } c; \\ \hline \end{array}$$

$$\{\top\}\ \mathrm{CPA} \sim \mathbf{G}\ \{(\neg r \in \mathbf{L}_H^{\mathcal{A}})_{\langle 2\rangle} \ \Rightarrow\ =_{\{b,b'\}}\}$$

Hence:

$$\Pr[\mathrm{CPA} : b' = b] - \Pr[\mathbf{G} : b' = b] \leq \Pr[\mathbf{G} : r \in \mathbf{L}_H^{\mathcal{A}}]$$

# Equivalence

$$\boxed{\begin{array}{l} \mathcal{E}_{pk}(m): \\ \quad r = \$\{0,1\}^{\ell}; \\ \quad h = \$\{0,1\}^{k}; \\ \quad s = h \oplus m; \\ \quad c = f_{pk}(r) \parallel s; \\ \quad \text{return } c; \end{array}} \quad \rhd \quad \boxed{\begin{array}{l} \mathcal{E}_{pk}(m): \\ \quad r = \$\{0,1\}^{\ell}; \\ \quad s = \$\{0,1\}^{k}; \\ \quad h = s \oplus m; \\ \quad c = f_{pk}(r) \parallel s; \\ \quad \text{return } c; \end{array}}$$

$$\{\top\}\ \mathbf{G} \sim \mathbf{G}'\ \{=_{\{b,b',\mathbf{L}_H^{\mathcal{A}}\}}\}$$

Hence:

$$\Pr[\mathbf{G} : r \in \mathbf{L}_H^{\mathcal{A}}] = \Pr[\mathbf{G}' : r \in \mathbf{L}_H^{\mathcal{A}}]$$

$$\Pr[\mathbf{G} : b' = b] = \Pr[\mathbf{G}' : b' = b] = \frac{1}{2}$$

# Equivalence

$$\boxed{\begin{array}{l}\mathcal{E}_{pk}(m):\\ \quad r = \$\{0,1\}^{\ell};\\ \quad h = \$\{0,1\}^{k};\\ \quad s = h \oplus m;\\ \quad c = f_{pk}(r) \parallel s;\\ \quad \text{return } c;\end{array}} \quad \rhd \quad \boxed{\begin{array}{l}\mathcal{E}_{pk}(m):\\ \quad r = \$\{0,1\}^{\ell};\\ \quad s = \$\{0,1\}^{k};\\ \quad h = s \oplus m;\\ \quad c = f_{pk}(r) \parallel s;\\ \quad \text{return } c;\end{array}}$$

$$\{\top\}\ \mathbf{G} \sim \mathbf{G}'\ \{=_{\{b,b',\mathbf{L}_H^{\mathcal{A}}\}}\}$$

Hence:

$$\Pr[\text{CPA} : b' = b] - \frac{1}{2} \leq \Pr[\mathbf{G}' : r \in \mathbf{L}_H^{\mathcal{A}}]$$

# Reduction

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

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

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

**Adversary** $\mathcal{I}(p, x)$ :
$(m_0, m_1) = \mathcal{A}_1(p);$
$b = \$\{0,1\};$
$s = \$\{0,1\}^k;$
$c^\star = x \parallel s;$
$b' = \mathcal{A}_2(c^\star);$
$y' = [z \in \mathbf{L}_H^{\mathcal{A}} \mid f_p(z) = x];$
return $y'$;

$$\{\top\}\ \mathbf{G}' \sim \mathrm{OW}\ \{(r \in \mathbf{L}_H^{\mathcal{A}})_{\langle 1\rangle} \Rightarrow (y' = y)_{\langle 2\rangle}\}$$

Hence:

$$\Pr[\mathbf{G}' : r \in \mathbf{L}_H^{\mathcal{A}}] \leq \Pr[\mathrm{OW} : y' = y]$$

# Reduction

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

**Game** OW :
$(sk, pk) = \mathcal{K}();$
$y = \$\{0,1\}^\ell;$
$y' = \mathcal{I}(pk, f_{pk}(y));$
return $(y' = y);$
**Adversary** $\mathcal{I}(p, x)$ :
$(m_0, m_1) = \mathcal{A}_1(p);$
$b = \$\{0,1\};$
$s = \$\{0,1\}^k;$
$c^\star = x \parallel s;$
$b' = \mathcal{A}_2(c^\star);$
$y' = [z \in \mathbf{L}_H^{\mathcal{A}} \mid f_p(z) = x];$
return $y';$

$$\{\top\}\ \mathbf{G}' \sim \mathrm{OW}\ \{(r \in \mathbf{L}_H^{\mathcal{A}})_{\langle 1\rangle} \Rightarrow (y' = y)_{\langle 2\rangle}\}$$

Hence:

$$\Pr[\mathrm{CPA} : b' = b] - \frac{1}{2} \leq \Pr[\mathrm{OW} : y' = y]$$

# Variation on indistinguishability

For every adversary $\mathcal{A}$, there exists an adversary $\mathcal{B}$ st

$$\left|\mathrm{Pr}[\mathrm{CPA}(\mathcal{A}) : b' = b] - \frac{1}{2}\right| = \mathrm{Pr}[\mathrm{CPA}(\mathcal{B}) : b' = b] - \frac{1}{2}$$

By case analysis on $\mathrm{Pr}[\mathrm{CPA}(\mathcal{A}) : b' = b] \leq \frac{1}{2}$

- If true, then $\mathcal{B}$ returns the result of $\mathcal{A}$
- If false, then $\mathcal{B}$ returns the negation of the result of $\mathcal{A}$