Computer-aided cryptographic proofs

Gilles Barthe
IMDEA Software Institute, Madrid, Spain

July 18, 2014

Motivation

- Cryptography is a small but important part of security
- Proofs are a small but important part of cryptography
- Hard to get right
- Often iterate over extended period ($\geq$10 years)

- *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

Computer-aided cryptographic proofs

provable security
=
deductive verification of parametrized probabilistic programs

- adhere to cryptographic practice
  - ☞ same proof techniques
  - ☞ same guarantees
  - ☞ same level of abstraction
- leverage existing verification techniques and tools
  - ☞ program logics, VC generation, invariant generation
  - ☞ SMT solvers, theorem provers, proof assistants, CAS
  - ☞ certified compilers

EasyCrypt

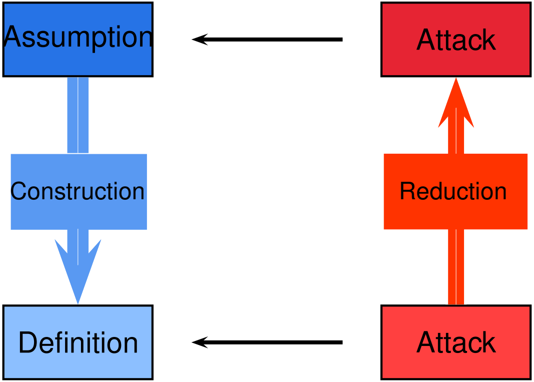
(B. Grégoire, P.-Y. Strub, F. Dupressoir, B. Schmidt, C. Kunz)

- Initially a weakest precondition calculus for pRHL
- Now a full-fledged proof assistant
  - ☞ Proof engine inspired from SSReflect
  - ☞ Calls to SMT and CAS
  - ☞ Embedding of rich probabilistic language w/ modules (neither shallow nor deep)
  - ☞ Support for different program logics
  - ☞ Reasoning in the large

Applications

- PKCS encryption
- Verification of cryptographic systems
- Key-exchange protocols under weaker assumptions

Reductionist proofs



Reductionist statement

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

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

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

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

A language for cryptographic games

$\mathcal{C}$	$::=$	skip	skip
	$\mid$	$\mathcal{V} \leftarrow \mathcal{E}$	assignment
	$\mid$	$\mathcal{V} \overset{\$}{\leftarrow} \mathcal{D}$	random sampling
	$\mid$	$\mathcal{C};\ \mathcal{C}$	sequence
	$\mid$	if $\mathcal{E}$ then $\mathcal{C}$ else $\mathcal{C}$	conditional
	$\mid$	while $\mathcal{E}$ do $\mathcal{C}$	while loop
	$\mid$	$\mathcal{V} \leftarrow \mathcal{P}(\mathcal{E}, \ldots, \mathcal{E})$	procedure call

- $\mathcal{E}$: (higher-order) expressions } user extensible
- $\mathcal{D}$: discrete sub-distributions } user extensible
- $\mathcal{P}$: procedures
  - . oracles: concrete procedures
  - . adversaries: constrained abstract procedures

Reasoning about programs

- Probabilistic Hoare Logic

$$\models \{P\} c \{Q\} \diamond \delta$$

- Probabilistic Relational Hoare logic

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

- Ambient logic

Applications

Allows deriving judgments of the form

$$\Pr_{c_1, m_1}[A_1] \diamond \delta$$

or

$$\Pr_{c_1, m_1}[A_1] \diamond \Pr_{c_2, m_2}[A_2]$$

or

$$|\Pr_{c_1, m_1}[A_1] - \Pr_{c_2, m_2}[A_2]| \leq \Pr_{c_2, m_2}[F]$$

pRHL: probabilistic relational Hoare logic

- Judgment
$$\vDash \{P\}\ c_1 \ \sim\ c_2\ \{Q\}$$
where P and Q denote relations on memories
- Validity
$$\forall m_1, m_2.\ (m_1, m_2) \vDash P \implies (\llbracket c_1 \rrbracket\ m_1, \llbracket c_2 \rrbracket\ m_2) \vDash Q^\sharp$$
- Definition of $\cdot^\sharp$ drawn from probabilistic process algebra

Application

Assume $\vDash \{P\}\ c_1\ \sim\ c_2\ \{Q\}$ and $(m_1, m_2) \models P$

If $Q \triangleq \bigwedge_{x \in X} x\langle 1 \rangle = x\langle 2 \rangle$ and $\mathrm{FV}(A) \subseteq X$ then

$$\Pr_{c_1, m_1}[A] = \Pr_{c_2, m_2}[A]$$

Proof rule: assignments and conditionals

Assignments

$$\frac{}{\models \{Q\{e\langle 1\rangle/x\langle 1\rangle\}\{e'\langle 2\rangle/x'\langle 2\rangle\}\}\ x \leftarrow e\ \sim\ x' \leftarrow e'\ \{Q\}}$$

$$\frac{}{\models \{Q[x\langle 1\rangle := e\langle 1\rangle]\}\ x \leftarrow e\ \sim\ \text{skip}\ \{Q\}}$$

Conditionals

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

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

Proof rules: random assignment

Intuition

Let A be a finite set and let $f, g : A \to B$. Define

- $c = x \xleftarrow{\$} \mu; y \leftarrow f\ x$
- $c' = x \xleftarrow{\$} \mu'; y \leftarrow g\ x$

Then $[\![c]\!] = [\![c']\!]$ (extensionally) iff there exists $h : A \overset{1-1}{\to} A$ st

- $f = g \circ h$
- for all a, $\mu(a) = \mu'(h(a))$

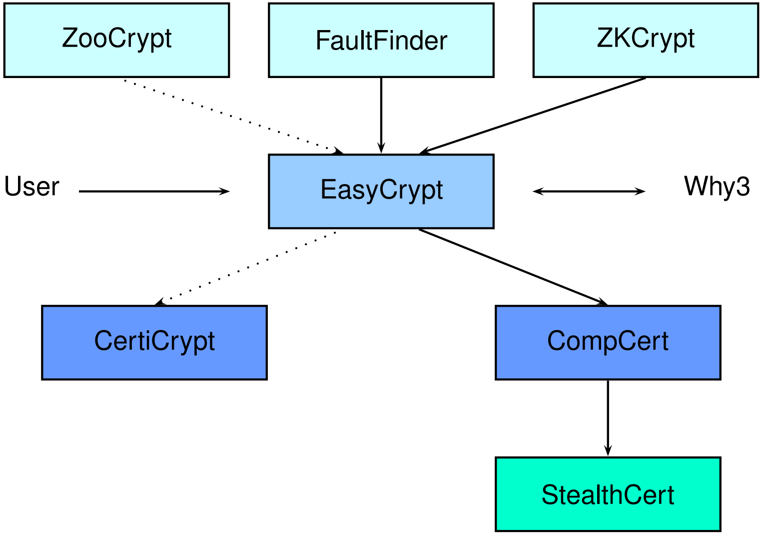
$$\frac{h \text{ is 1-1 and } \forall a,\ \mu(a) = \mu'(h(a))}{\models \{\forall v, Q\{h\ v/x\langle 1\rangle\}\{v/x\langle 2\rangle\}\}\ x \xleftarrow{\$} \mu\ \sim\ x \xleftarrow{\$} \mu'\ \{Q\}}$$

Adversaries

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

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

EasyCrypt toolchain



ZooCrypt

Aautomated analysis of padding-based encryption schemes

- Attack finding tool
- Proof search for domain-specific logics
- Interactive tutor
- Generation of EasyCrypt proofs (ongoing)

- Generated $\geq 10^6$ padding-based encryption schemes
- Proved chosen-plaintext security for 11%
- Found attacks for 88%
- About .5% unknowns
- Interactive tutor

Generic Group Analyzer

- Profusion of (non-standard) cryptographic assumptions
  - ☞ for efficiency reasons
  - ☞ for achieving a construction
- Some assumptions are broken
- Heuristics: prove absence of algebraic attacks
  - ☞ Master theorem: security from symbolic condition
  - ☞ Use CAS or SMT to discharge symbolic condition

Example: DDH

- Cannot distinguish between (g^x, g^y, g^{xy}) and (g^x, g^y, g^z)
- Symbolic condition: (x, y, xy) and (x, y, z) satisfy the same linear equalities

FaultFinder

- Goal: find physical attacks on implementations
- Isolate post-conditions ϕ that enable attacks
- Given an implementation *c*, find faulted implemtation $\hat{c}$ st

$$\{\psi\}\hat{c}\{\phi\}$$

- Use SMT-based synthesis
- New attacks for RSA and ECDSA signatures

Conclusion

- Solid foundation for cryptographic proofs
- Formal verification of emblematic case studies

Different styles of proofs

- EasyCrypt: proof objects
- ZooCrypt: proof trees
- GGA: traces
- FaultFinder: proofs for attack finding

Further directions

- Proof Theory of Cryptographic Proofs
- Synthesis of “classical” cryptography

`http://www.easycrypt.info`