A Verification Toolbox for Cryptographic Primitives

David NOWAK

Research Center for Information Security, AIST

JSIAM annual meeting
FAIS organized session
September 18, 2008

Computer-aided mathematics and security proofs

- Scientific computing
  - Computing derivatives
  - Numerical simulations
  - ⋮
- Proving
  - **Automated theorem prover**
    - Secure cryptographic primitives are assumed.
    - Some protocols and generic schemes can be proved secure automatically.
    - For example, CryptoVerif, ProVerif and Scyther are such tools.
  - **Proof assistant**
    - Security of cryptographic primitives cannot be proved automatically.
    - Those proofs are error-prone and difficult to check.
    - A proof assistant can check such proofs.

On top of the proof assistant Coq,
I am making a verification toolbox for cryptographic primitives.

Game-based security proofs

- (Bellare and Rogaway, 2004) and (Shoup, 2004) advocate sequences of games as a way to get better proofs.

  > *" Many proofs in cryptography have become essentially unverifiable. Our field may be approaching a crisis of rigor." (Bellare and Rogaway, 2004).*

- (Halevi, 2005) advocates the need for a software which can deal with the mundane part of writing and checking game-based proofs.
- Security properties are modeled as probabilistic programs:
  such a program implements a game that is to be solved by the attacker.
- The attacker is modeled as an external procedure which is interfaced with the game.
- The goal is then to prove that any efficient attacker has at most a negligible advantage over a random player.

Related work

- Game-based proof of the PRP/PRF switching lemma in Coq (Affeldt, Tanaka and Marti, 2007)

- Certicrypt (Gilles Barthe et al.)
  - A project of the *Microsoft Research INRIA Joint Centre*
    `http://www.msr-inria.inria.fr/projects/sec/certicrypt/`
  - On top of the proof assistant Coq
  - It involves 2 full-time researchers and 3 PhD students.

- Talk at FCC 2008 on "a formal language for cryptographic pseudocode" (Michael Backes, Matthias Berg, and Dominique Unruh)
  - On top of the proof assistant Isabelle
  - They do not yet have examples of applications.

Deep vs. Shallow embedding

- Deep embedding
  - This is the approach by previously mentioned related work.
  - Games are syntactic objects.
  - Game transformations are syntactic manipulations which can be automated.
  - Advantages:
    - Possibility of proving completeness of decision procedures
    - Smaller proof terms
  - But it requires a huge machinery.
- Shallow embedding
  - This is the approach I chose.
  - Games are probability distributions (as advocated by Shoup).
  - They can still be subject to formal manipulation and automated, through the metalanguage of the proof assistant.

The proof assistant Coq

- Developed at INRIA since 1984
- Based on a kernel which checks that:
  a given proof term p is really a proof of a given statement H.
- A tactic language (metalanguage) for building proofs incrementically
- Decision procedures for decidable fragments
- The kernel is the only critical part:
  it will reject wrong proof terms.
- To be closer to mathematical practice: notations, implicit parameters and subset coercions are available.
- A standard library:
  arithemetic, analysis, polymorphic lists...

Overview of the toolbox

- First layer: various extensions of the standard library of Coq
  - Probability distributions
  - Cyclic groups
  - Integers modulo n (Jacobi symbol, Blum primes...)
  - Bitstrings
- Second layer: Security proofs
  - definitions of hard problems:
    - DDH problem
    - quadratic residuosity problem
    - entropy smoothing
  - Security definitions:
    - semantic security, for public-key cryptographic schemes
    - unpredictability, for pseudo-random bit generators
  - A collection of game transformations
    - stemming from properties proven in the first layer
    - Example: for all game G, for all predicate P,

$$\begin{array}{l}\text{let } x \xleftarrow{R} \mathbb{Z}_n^*(+1) \text{ in} \\ G(x^2)\end{array} \quad \equiv_0^P \quad \begin{array}{l}\text{let } y \xleftarrow{R} \mathrm{QR}_n \text{ in} \\ G(y)\end{array}$$

Applications

- Semantic security of the ElGamal public-key cryptographic scheme (and also its hashed version)
  Based on the intractability of the DDH problem
- Semantic security of the Goldwasser-Micali public-key cryptographic scheme
  Based on the intractability of the quadratic residuosity problem
- Unpredictability of the Blum-Blum-Shub pseudorandom bit generator
  Based on the intractability of the quadratic residuosity problem
- Next application: security of your cryptographic primitive ?

Limitations

- The following kinds of proofs are not supported :-)
  - proof by intimidation such as:
    "*trivial*" or "*The reader may easily supply the details*"
  - proof by never-ending revisions
  - proof by hidden assumption
- No random oracle model:

  Although it is nowadays fashionable to end the title of a paper by "without random oracles".
- Neither exact or asymptotic running time are computed.

Demo