Models and Proofs of Protocol Security: A Progress Report

Martín Abadi — MSR, UCSC
Bruno Blanchet — CNRS, ENS, INRIA
Hubert Comon-Lundh — ENS Cachan, INRIA, RCIS-AIST

(Martín Abadi's invited paper at CAV'09, to appear)

June 2009

Introduction

Two approaches for automating computational proofs of protocols:

- Indirect approach:
  1. prove a security property in the formal model
  2. use a computational soundness theorem
- Direct approach:
  prove the security property directly in the computational model

Our goal: study the state of the art in these two approaches and compare them.

Introduction

Two approaches for automating computational proofs of protocols:

- Indirect approach:
  1. prove a security property in the formal model
     prove an observational equivalence using ProVerif
  2. use a computational soundness theorem
     use the Comon-Lundh, Cortier, CCS'08 soundness theorem for observational equivalence
- Direct approach:
  prove the security property directly in the computational model using CryptoVerif

Our goal: study the state of the art in these two approaches and compare them.

Our case study: a variant of the WMF protocol

We consider a variant of the Wide-Mouth Frog protocol, without timestamps, but with tags to distinguish the first two messages:

$$\begin{array}{lll} \text{Message 1.} & A \rightarrow S: & \{c_0, B, k\}_{k_{AS}} \\ \text{Message 2.} & S \rightarrow B: & \{c_1, A, k\}_{k_{BS}} \\ \text{Message 3.} & B \rightarrow A: & \{m\}_k \end{array}$$

We prove the (strong) secrecy of the payload m.
(The adversary cannot distinguish two different values of the payload.)

Proof in the formal model

We have modeled this protocol in the input language of ProVerif, an extension of the pi calculus with cryptography.

- We model a symmetric encryption scheme that is:
  - probabilistic, using an additional argument containing random coins: $(\nu r)\ \mathrm{encrypt}(m, k, r)$.
  - key-revealing: an adversary can test whether two ciphertexts use the same key.
    reduc $\mathrm{keyeq}(\mathrm{encrypt}(x, y, r), \mathrm{encrypt}(x', y, r')) = \mathrm{true}$.
  - length-concealing.

Proof in the formal model

- Representation of tables of keys:
  - We avoid functions that link keys to principals (computationally unsound).
  - We avoid private channels (not supported by Comon, Cortier, CCS'08, but may be permitted in future work).

  These points lead to some code duplication in the model.
- Limitation: the payload is the same for all sessions. (ProVerif does not terminate in the general case.)

ProVerif automatically proves the strong secrecy of the payload m: $P(m) \sim P(m')$.

The computational soundness theorem

If $P \sim_s P'$ in the formal model, then P is computationally indistinguishable from P'.

Assumptions:

- The encryption scheme is IND-CPA and INT-CTXT.
- The attacker can create a key only using the key-generation algorithm.
- There are no encryption cycles.
    - There is an ordering $<$ on private keys such that, if k appears in the plaintext of a ciphertext encrypted under k', then $k < k'$.
    - We have proved this manually for WMF.
- It is possible to compute a symbolic representation of any bitstring.
    - This "parsing assumption" is probably not necessary but eases the proofs.

Applying the computational soundness theorem

A priori, $\sim$ (proved by ProVerif; encryption is length-concealing) is different from $\sim_s$ (required by the computational soundness theorem; encryption may be length-revealing).

They can be reconciled by

- adding in the ProVerif model functions that reveal the length or structure of plaintexts (requires some extensions of ProVerif), or
- requiring encryption to be length-concealing in the computational model.

  With this hypothesis, we obtain the desired computational indistinguishability of the payload.

Direct proof in the computational model

We have also modeled our protocol in CryptoVerif.

Assumptions:

- The encryption scheme is IND-CPA and INT-CTXT.
- The function *concat*, which builds the first two messages $concat(c_i, h, k)$, returns bitstrings of constant length.
- One can compute x, y, z from $concat(x, y, z)$ in PTIME.
- All payloads have the same length.

We manually introduce a case distinction between honest and dishonest interlocutors of A.

Then, CryptoVerif proves automatically the desired computational indistinguishability.

Comparison with the indirect approach

- We do not assume that the attacker can create a key only using the key-generation algorithm.
- We do not assume the absence of encryption cycles.
    - The success of the game transformation sequence shows that there is a key hierarchy.
- We do not have any parsing assumption.

Lessons learned

- There has been much progress in recent years on the verification of security protocols.
- Soundness theorems often require more hypotheses.
- When the hypotheses are met, a symbolic proof suffices and is generally easier to obtain.
- Both the indirect and the direct approaches still present challenges.
    - Adapt symbolic tools to the hypotheses of computational soundness theorems: prove the absence of key cycles, allow length-revealing encryption, ...
    - Extend computational soundness theorems: allow private channels, nested replications, . . .
    - Develop further computationally-sound provers: more automation and/or more fine-grained user guidance, more primitives and game transformations, . . .