FormaCrypt: Formal Computational Cryptography

Bruno Blanchet[1], David Pointcheval[1]
Jean Goubault-Larrecq[2], Stéphanie Delaune[2], Steve Kremer[2]
Véronique Cortier[3], Heinrich Hördegen[3], Mathieu Turuani[3]
Eugen Zalinescu[3], Martín Abadi[4]

[1]LIENS [2]LSV [3]LORIA [4]UCSC & Microsoft Research

November 2007

ARA SSIA FormaCrypt: participants

- Laboratoire d'Informatique de l'Ecole Normale Supérieure (LIENS)
  - Bruno Blanchet
  - David Monniaux (at VERIMAG since Sept. 1st, 2007)
  - David Pointcheval
- Laboratoire Spécification et Vérification (LSV), ENS Cachan
  - Jean Goubault-Larrecq
  - Mathieu Baudet (at DCSSI since July 1st, 2006)
  - Stéphanie Delaune (from October 2007)
  - Steve Kremer
  - Laurent Mazaré (October 2006–April 2007)
- Laboratoire Lorrain de Recherche en Informatique et ses Applications (LORIA)
  - Véronique Cortier
  - Stéphanie Delaune (January 2007–September 2007)
  - Heinrich Hördegen (Phd thesis to be defended soon)
  - Mathieu Turuani
  - Bogdan Warinschi (in Bristol since January 2007)
  - Eugen Zalinescu (Phd thesis to be defended soon)
- Scientific advisor: Martín Abadi

Proofs of cryptographic protocols

There are two main frameworks for analyzing security protocols:

- The Dolev-Yao model: a formal, abstract model.

  The cryptographic primitives are ideal blackboxes.
  The adversary uses only those primitives.

  Proofs can be done automatically.

- The computational model: a realistic model.

  The cryptographic primitives are functions on bit-strings.
  The adversary is a polynomial-time Turing machine.

  Proofs are done manually.

Our goal: bridge the gap between these two frameworks.

Three approaches

We have considered three approaches:

- Direct approach: build an automatic computationally sound prover.
- Intermediate approach: design a computationally sound logic, for reasoning symbolically on protocols.
- Modular approach: obtain computational soundness results, that is, show that security in the formal model implies security in the computational model.

We will obviously compare these approaches on examples ranging from protocols of the literature to more complex, realistic protocols.

An automatic computationally sound prover

We have implemented an automatic prover sound in the computational model:

- proves secrecy and correspondence properties.
  Correspondence = if some event has been executed, then other events have been executed before (except in cases of negligible probability).
- handles various cryptographic primitives: MACs (message authentication codes), symmetric encryption, public-key encryption, signatures, hash functions, ...
- works for a parametric number of sessions with an active adversary.
- gives a bound on the probability of an attack (exact security).

Produced proofs

As in Shoup's or Bellare and Rogaway's method, the proof is a sequence of games:

- The prover is given the first game, which represents real protocol, in interaction with an adversary.
- The prover transforms each game into the next one by syntactic transformations or by applying security assumptions on cryptographic primitives.

  The difference of probability between consecutive games is bounded.
- The last game is "ideal": the desired security properties can be read directly on it.

Games are formalized in a process calculus.

The user is allowed (but does not have) to interact with the prover to make it follow a specific sequence of games.

Experimental results

The prover is available at http://www.cryptoverif.ens.fr

We have tested it successfully on many examples:

- protocols of the literature: incorrect and corrected versions of Otway-Rees, Yahalom, Needham-Schroeder shared-key and public-key, and Denning-Sacco public key;
- the Full Domain Hash signature scheme;
- encryption schemes of [Bellare and Rogaway, CCS'93].
- Kerberos (joint work with A. D. Jaggard, A. Scedrov, and J.-K. Tsay)

Planed extensions

1. Handle other primitives, such as Diffie-Hellman key agreements.
2. Improve the proof strategy, for more automation.
3. Test on more examples.

A computationally sound logic

We have studied the following approach for proving protocols:

1. start from an existing protocol logic, designed in the formal model (here, the Protocol Composition Logic),
2. and adapt it to the computational model.

Advantage of this approach:
proofs that use the logic in the formal model can be adapted to the computational model with only minor corrections.

The Protocol Composition Logic (PCL)

The Protocol Composition Logic allows symbolic reasoning on protocols:

- The protocol is specified in a simple "protocol programming language".
- The logic consists of both
  - logical formulas (including predicates that specify knowledge and actions of participants)
  - and modal formulas, similarly to the Floyd-Hoare logic (if a formula is true at some point and certain actions are executed, then another formula holds afterwards).
- The logic allows compositional reasoning.

Adapting PCL to the computational model

We have adapted the Protocol Composition Logic to the computational model:

- We have given a new probabilistic, polynomial-time semantics to the logic: a formula is true if it holds with asymptotically overwhelming probability.
- We have given a meaning to logical connectives in this semantics.
- We had to extend the logic with new predicates that make sense in the computational model but not in the formal one, for example to express indistinguishability.

A soundness proof has been done for a subset of PCL with positive indistinguishability tests.

Proving more complex properties

We have extended this logic to more complex properties, in particular secrecy of keys.

This result allows the following compositional reasoning:

- we first prove the security of keys established using a key exchange protocol;
- then, we infer the security of a secure channel application that uses these keys.

Planed extensions

1. Soundness for any proof in PCL extended with computational tests.
2. Make the semantics more direct and natural.

Formal model: several abstractions

Messages are modeled by terms

- $\{m\}_k$: message m encrypted by k
- $\langle m_1, m_2 \rangle$: pair of m_1 and m_2
- ...

$\longrightarrow$ no collisions:

$$\forall m, m', k, k' \quad \{m\}_k \neq \{m'\}_{k'}, \{\{m\}_k\}_k \neq m, \langle m, m' \rangle \neq \{m\}_k, \ldots$$

Perfect encryption assumption:

Nothing can be learned from $\{m\}_k$ except if k is known.

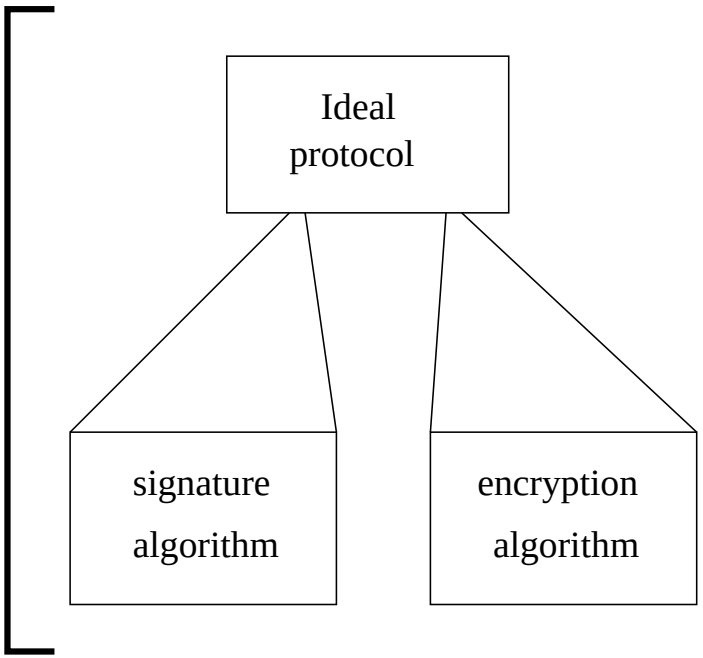
Much easier to analyze automatically

- numerous decidability results
- numerous automatic tools

Goal: soundness of the formal model

Composition of two approaches

Implemented protocol



Formal approach: verification of idealized protocols

Cryptographers: verification of the cryptographic primitives

Combination result in presence of hash functions

Example

$A \to B : h(s)$

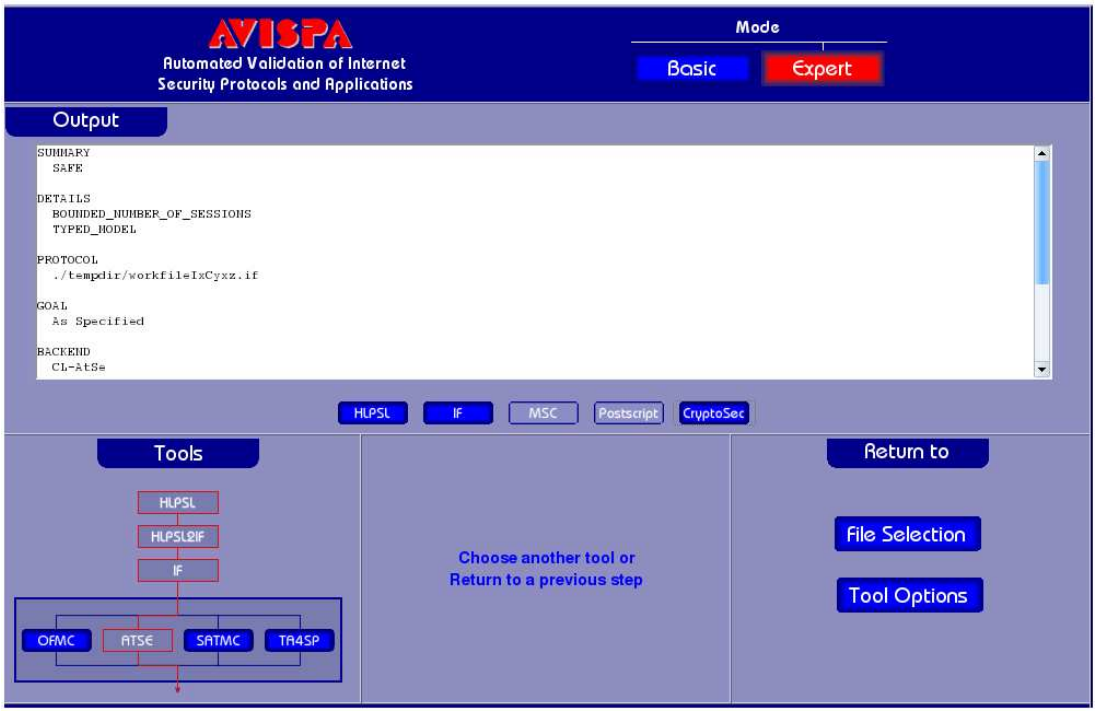
s is symbolically secret but not indistinguishable to an attacker: $h(n_b), n_0, n_1 \to b$

Results:

1. Design of a new formal secrecy property
2. Proof of its soundness and its faithfulness w.r.t. indistinguishability in our new setting:
   - pairing
   - asymmetric encryption
   - hashes (random oracle model)
3. NP-completeness of the secrecy property

Application: CryptoSec

Automatic proof at the cryptographic level
using Avispa, a symbolic theorem prover.



→ 9 protocols proved secure (for authentication or secrecy properties) at the cryptographic level.

Guessing attacks

Guessing attacks: The adversary can guess low entropy values such as passwords and verify them off-line.

Results:

1. Design of a new security property for the interaction between normal and password-based encryption
2. Proof of soundness of static-equivalence (passive case) w.r.t. indistinguishability for:
   - asymmetric encryption
   - symmetric encryption
   - password-based encryption

Equational theories

Equational theories: Cryptographic primitives can be modelled as equational theories, which allows to specify algebraic properties

Results:

1. Generalization of the DDH assumption and soundness result for modular exponentiation in the presence of a passive adversary
2. Soundness result for protocols based on bilinear pairings in the presence of a passive adversary
3. A general framework for reasoning about the soundness of equational theories in the presence of an adaptive adversary: equational theories include modular exponentiation and xor; application to dynamic group key exchange protocols

Planed extensions

1. Study branching properties, such as fairness.
   We have designed a model of branching properties for contract signing in the computational model; computational soundness results are still needed.
2. Prove the secrecy of keys (not only of nonces).
3. Soundness results for equational theories in the presence of active adversaries

Conclusion

- We have investigated three different approaches for bridging the gap between the computational and the formal models of cryptography.
- Up to now, the project has produced 15 papers and 2 tools that provide proofs of protocols in the computational model.
- These three approaches will be extended and compared in the next years.

More details, publications, and software available at:
http://www.di.ens.fr/~blanchet/formacrypt/