Automated Proof Techniques for Cryptographic Assurance

Bruno Blanchet

EPI Prosecco

INRIA Paris

September 2016
Models of protocols

Two models of security protocols:

- **Symbolic model (Dolev-Yao):**
  - Primitives are black boxes.
  - Messages are terms on these primitives.
  - The adversary is restricted to apply only those primitives.
  
  This model **facilitates the automation** of proofs.

- **Computational model:**
  - Messages are bitstrings.
  - Primitives are functions on bitstrings.
  - The adversary is any (probabilistic polynomial-time) Turing machine.

  This model is **more realistic**: the adversary can apply any algorithm.
Models of protocols

Two models of security protocols:

- **Symbolic model (Dolev-Yao):**
  - Primitives are black boxes.
  - Messages are terms on these primitives.
  - The adversary is restricted to apply only those primitives.

  This model **facilitates the automation** of proofs.

- **Computational model:**
  - Messages are bitstrings.
  - Primitives are functions on bitstrings.
  - The adversary is any (probabilistic polynomial-time) Turing machine.

  This model is **more realistic**: the adversary can apply any algorithm.
Protocol verification in the symbolic model

Security protocols are infinite state:
- The attacker can create messages of unbounded size.
- Unbounded number of sessions of the protocol.

Solutions:
- Bound the state space arbitrarily:
  exhaustive exploration (model-checking, ...);
  find attacks but not prove security.
- Bound the number of sessions:
  the insecurity is NP-complete (with reasonable assumptions).
- Unbounded case:
  the problem is undecidable.
Solutions to undecidability

To solve an undecidable problem, we can

- Use approximations, abstraction.
- Terminate on a restricted class.
- Rely on user interaction or annotations.

In ProVerif, we do the first two, using a very precise abstraction by Horn clauses.
Features of ProVerif

- Fully automatic.
- Works for unbounded number of sessions and message space.
- Handles a wide range of cryptographic primitives, defined by rewrite rules or equations.
- Handles various security properties: secrecy, authentication, some equivalences.
- Limitations:
  - Does not always terminate. However, efficient in practice: small examples verified in less than 0.1 s; complex ones in minutes.
  - May answer “I don’t know” (false attack). However, very precise in practice: no false attack in our tests for secrecy and authentication.
ProVerif

Protocol: Pi calculus + cryptography
Primitives: rewrite rules, equations

Properties to prove: Secrecy, authentication, process equivalences

Automatic translator

Horn clauses

Derivability queries

Resolution with selection

Non-derivable: the property is true

Derivation

Attack: the property is false

False attack: I don’t know

Bruno Blanchet (INRIA)
Automated Crypto Proofs
September 2016
The Horn clause representation

The main predicate:

\[
\text{attacker}(M) \quad \text{means} \quad \text{“the attacker may have } M \text{”}
\]

Thanks to this predicate, we can model actions of the adversary:

**Example: Shared-key encryption and decryption**

\[
\text{attacker}(m) \land \text{attacker}(k) \rightarrow \text{attacker(encrypt}(m, k)) \\
\text{attacker(encrypt}(m, k)) \land \text{attacker}(k) \rightarrow \text{attacker}(m)
\]

and of the protocol participants:

**Example: A receives } M \text{ and replies with } M'**

\[
\text{attacker}(M) \rightarrow \text{attacker}(M')
\]
CryptoVerif

The protocol verifier CryptoVerif:

- works directly in the computational model.
- proves secrecy and correspondence (authentication) properties.
- provides a generic method for specifying properties of cryptographic primitives which handles MACs (message authentication codes), symmetric encryption, public-key encryption, signatures, hash functions, Diffie-Hellman key agreements, . . .
- works for $N$ sessions (polynomial in the security parameter).
- gives a bound on the probability of an attack (exact security).
- has automatic and manual modes.
Produced proofs

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

- The first game is the real protocol.
- One goes from one game to the next by syntactic transformations or by applying the definition of security of a cryptographic primitive. The difference of probability between consecutive games is negligible.
- The last game is “ideal”: the security property is obvious from the form of the game. (The advantage of the adversary is usually 0 for this game.)
Models vs. implementations

- ProVerif and CryptoVerif automatically analyze protocol models.
- Just that models are very abstract:
  - Protocol models may miss implementation attacks.
- Verified models are good
  - ...but verified implementations are much better!
Generation of implementations (David Cadé)

- **CryptoVerif specification** → **Our Compiler** → **OCaml Compiler** → **Implementation** → **Network Code** → **Proof in the computational model**
Verified implementations with F*, https://www.fstar-lang.org/

- F* is a new programming language
- ...putting together:
  - impure functional programming in ML
    - extracts to OCaml and F#, interoperates
  - the automation of SMT-based verification systems
    - like in Why3, Frama-C, Boogie, VCC, Dafny
  - the expressive power of interactive proof assistants based on dependent types
    - like in Coq, Agda, or Lean
miTLS, http://www.mitls.org/

- Formally verified reference implementation of TLS 1.2 in F7/F* (working towards TLS 1.3)
- Written from scratch focusing on verification

This page is served using the miTLS demo HTTPS server. (Go back to production server)
- ciphersuite: TLS_RSA_WITH_AES_128_CBC_SHA,
- compression: NullCompression,
- version: TLS_1p2
Lead to the discovery of many attacks in TLS implementations

**Triple Handshakes Consid Breaking and Fixing Authent**

**March 4, 2014**

<table>
<thead>
<tr>
<th>Introduction</th>
<th>TLS Weaknesses</th>
<th>Triple Handshakes</th>
</tr>
</thead>
<tbody>
<tr>
<td>Countermeasures</td>
<td>Disclosure</td>
<td>Other Attack</td>
</tr>
</tbody>
</table>

**SMACK: State Machine AttaCKs**

Implementations of the Transport Layer Security (TLS) protocol are vulnerable to several attacks.

**Tracking the FREAK Attack**

On Tuesday, March 3, 2015, researchers announced a new SSL/TLS vulnerability called the FREAK attack. It allows an attacker to intercept HTTPS connections between vulnerable clients and servers and force them to use weak encryption channels.

The FREAK attack disclosure was coordinated by the University of Michigan, including four team members.

For additional details, see this Washington Post article.

**The BEAST Wins Again:**

Documents

- PDF of slides
- Summary of briefing for non-experts
- Paper: Virtual Host Fare
- Paper: Triple Handshakes

**Exploit videos**

Disclaimer: The goal of these videos is to have a strong impact. The attack is not intended to be implemented by the discoverers.

**FREAK Attack Threatens SSL Clients**

Posted by Soulskill on Tuesday March 03, 2015 @04:29PM from the another-day-another-vun dept.

**msm1267** writes:

For the nth time in the last couple of years, security experts are warning about Internet-scale vulnerability, this time in some popular SSL clients. The flaw allows attackers to force clients to downgrade to weakened ciphers and break their supposedly encrypted communications through a man-in-the-middle attack.

**Bruno Blanchet (INRIA)**
Verified security protocols at several levels:

- **Specifications:**
  - In the **symbolic model**: ProVerif
    Available at [http://proverif.inria.fr/](http://proverif.inria.fr/)
  - In the **computational model**: CryptoVerif
    Available at [http://cryptooverif.inria.fr/](http://cryptooverif.inria.fr/)

- **Implementations:**
  - **Generation** of implementations from specifications: CryptoVerif
  - **Direct verification** of implementations: \( F^* \).
    Available at [https://www.fstar-lang.org/](https://www.fstar-lang.org/)

See [http://prosecco.inria.fr/](http://prosecco.inria.fr/)