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
The main predicate:

attacker($M$)  \textit{means} “the attacker may have $M$”.

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

\textbf{Example: Shared-key encryption and decryption}

\begin{align*}
\text{attacker}(m) \land \text{attacker}(k) \rightarrow & \quad \text{attacker(encrypt}(m, k)) \\
\text{attacker(encrypt}(m, k)) \land \text{attacker}(k) \rightarrow & \quad \text{attacker}(m)
\end{align*}

and of the protocol participants:

\textbf{Example: A receives $M$ and replies with $M'$}

attacker($M$) \rightarrow attacker($M'$)
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
- Protocol Code
- OCaml Compiler
- Implementation
- Proof in the computational model

Caption: Tool Input Result
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
Lead to the discovery of many attacks in TLS implementations

SMACK: State Machine AttaCKs

Implementations of the Transport Layer Security (TLS) protocol have been vulnerable to various attacks due to implementation errors. Our research team, including Bruno Blanchet, has been working on developing a symbolic model to identify these vulnerabilities. We have implemented this model to detect attacks in existing TLS implementations.

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, forcing them to use weak encryption. We have been tracking the impact of this attack and have identified several weaknesses in how Diffie-Hellman key exchange has been deployed.

The BEAST Wins Again: W

The BEAST attack is a zero-day vulnerability that allows attackers to decrypt HTTPS traffic. Our team has been working on mitigations and countermeasures to address this issue. We have developed a computational model to simulate the attack and test our countermeasures.

Conclusion

By improving our symbolic and computational models, we are able to identify and address vulnerabilities in TLS implementations. Our work has led to the discovery of many attacks and has contributed to the development of more secure TLS implementations.
Conclusion

Verified security protocols at several levels:

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

- **Implementations:**
  - **Generation** of implementations from specifications: CryptoVerif
  - **Direct verification** of implementations: F*.
    Available at https://www.fstar-lang.org/

See http://prosecco.inria.fr/