Formal Verification of an Internet Voting Protocol

Kristjan Krips

Cybernetica AS, University of Tartu

October 24, 2017

Why we need formal verification?

It is difficult to design secure cryptographic protocols. Manual overview does not find all of the vulnerabilities. Handwritten proofs about the security properties may contain errors.

Latest examples of problematic protocols:

- TLS - BEAST, Lucky13
- Bluetooth - BlueBorne
- WPA2 - KRACK

Why we need formal verification?

What were the problems of the previously mentioned protocols?

- cryptographic issues
- backward compatibility
- specification contains over 2800 pages
- specification is not public

Formal verification

Using formal methods it is possible to prove that the claimed properties hold. Computer-aided formalization can lead to computer-aided verification. This would remove some of the human errors. Computer-aided formalization & verification:

- which abstraction level to use for formalization?
- are all subprotocols specified?
- is it possible to verify all properties with a computer?

Formal verification of an online voting protocol

Online voting won't be accepted if it is not trusted. Blindly trusting voting systems or voting protocols is not smart. Thus, it should be possible to verify that the voting protocol behaves correctly.

It should be possible to verify that:

- protocol follows the security requirements
- protocol is implemented according to the specification
- implementation of the protocol does not contain bugs

I focused on the first property.

Formal verification of an online voting protocol

Most of the computer-aided security proof of cryptographic protocols have been done in the symbolic model by using ProVerif. However, symbolic model does not reflect real world.

Computational model allows to claim properties about probabilities. Formalizations in the computational model are more realistic but also more complex to prove. Computer-aided formalization & verification can be done with the proof assistant EasyCrypt.

Overview of EasyCrypt

- computer-aided verification
- works in computational model
- work in progress, developed by IMDEA (previously also by Inria)
- rather steep learning curve

Overview of EasyCrypt

EasyCrypt allows to specify primitives and protocols by using a probabilistic while language. Steps for formalization of the primitive or protocol:

- specify types, operations, axioms
- write security games and modules
- create lemmas to state the security claims
- create auxiliary lemmas that are required in the proofs
- prove the lemmas by using either proof tactics or SMT solvers

Overview of EasyCrypt

Proving process consists of proving subgoals

- goals can be resolved by using tactics
- lemmas, axioms might be required to apply a tactic
- lemmas, operations, axioms have to be found from the (undocumented) theory files
- trivial subgoals may have to be resolved manually by writing new lemmas and then proving these lemmas

Doing proofs in EasyCrypt is non-trivial. Simple goals may take much time to solve.

Formal verification of an online voting protocol

We chose to start by proving the integrity properties of the voting protocol that was used in Utah Republican caucus in 2016. We used EasyCrypt for the formalization and proving.

- integrity properties are easier to formalize & prove than privacy properties
- integrity properties were essential for the given protocol

Sample of the requirements for the analysed protocol

We focused on the following properties:

- correctness of the protocol
- tally integrity - submitted ballot has to be included in the final tally
- post-election individual verifiability (by using a receipt generated during the voting phase)

Verification with EasyCrypt

We used EasyCrypt to show:

- proof of correctness
- malicious voting client replacing the vote of the voter
- malicious voting servers not delivering the vote

Current & future work

Currently i am working on the verification of a new online voting protocol. This time the focus is on the privacy properties.

Questions?

Supported by University of Tartu ASTRA project PER ASPERA
Doctoral School of Information and Communication Technologies.

European Union
European Regional
Development Fund

Investing
in your future