Verifier for Electronic Election

Proven correct verifier to guarantee election integrity

Ratmir Mugattarov Thomas Haines Michael Norrish

Australian National University

What

- Electronic election is a software with cryptography
- Software is always **buggy**
- **Cryptography** causes more bugs
- Errors in election **harm** democracy
- We want guaranteed **zero** errors in election
- Fixing errors brings **more errors**
- Do not fix
- Instead - **verify** afterwards
- **Verify** computations of flawed election afterwards

but

- Again: Verifier is a **software**
- Software always buggy
- Errors ...
- **Answer:** Proof-based Development

Why verifier

- Verifier is **simpler** than election
  deals with numbers, not with human voter via browser over the internet like most of the election,
  separate smaller verifiers for security, privacy, integrity properties of election
- Does **not require** election itself to be correct
  Software Independence concept [1]:
  If Verifier is guaranteed correct, it suffices to guarantee election correctness
- Can guarantee election correctness

How to do

- Build a **verifier**
- **Prove** its code is correct
- **Compile** executable
- Run verifier on the transcript of **conducted election**

Technique

- Election uses sigma protocols
  Sigma protocol produces transcripts that evidence the stages of the election process, such as vote encryption, ballot submission, vote counting, result decryption etc.
  This data is used to verify conducted election properties.
- Construct the same sigma protocol
  We construct sigma protocol equivalent to the election sigma protocol.
  We construct elementary components so they can be reused for different sigma protocols.
  We use HOL Theorem Prover for construction so we can develop proofs in the same language.
- Prove sigma protocol
  Sigma protocol have formalised [2] definition of correctness.
  We prove that our equivalent sigma protocol is correct by definition.
  We develop proofs in HOL Theorem Prover so it is connected with the code.
- Verifier is in sigma protocol
  Sigma protocol contains verifier that can tell if the produced transcript is correct.
  We instantiate verifier from our equivalent sigma protocol and compile.
  We use CakeML compiler because it is proven to preserve the code properties [3].
- Verifier is correct
  Verifier operation is correct because we compile proven correct code with verified compiler CakeML [3].
  The code of verifier is correct because it is a part of proven correct sigma protocol.
  Sigma protocol is correct because we proved its properties required by definition.
  Proof is valid because HOL cannot accept false proof [4,5].
  Proof relates to code of verifier without a gap because they are written in the same language
- Verify election
  We use published election transcript - data available for audit of election and feed it as a transcript to our executable verifier. If verifier accepts, then this suffices to guarantee correctness of election by Software Independence concept [1].

Results

Done

We built election verifier, proved it correct, compiled and used it to **verify real election** property (integrity). Building components of verifier are proven correct and allow to be used for composing correct verifiers for other elections.

Key thing

Our verifier guarantees election is **100% correct** in this particular property (by Software Independence concept)

Value

Our **technique** and components can be used for developing verifiers for other elections and other properties, which can give us election guaranteed to have zero errors

References

[1] Ronald L. Rivest. On the notion of ‘software independence’ in voting systems. Philosophical Transactions of the Royal Society A: Mathematical, Physical and Engineering Sciences, 366:3759 – 3767, 2008.
[2] Gilles Barthe, Daniel Hedin, Santiago Zanella B´eguelin, Benjamin Gr´egoire, and Sylvain Heraud. A machine-checked formalization of sigma-protocols. 2010 23rd IEEE Computer Security Foundations Symposium, pages 246–260, 2010.
[3] Yong Kiam Tan, Magnus O. Myreen, Ramana Kumar, Anthony C. J. Fox, Scott Owens, and Michael Norrish. The verified cakeml compiler backend. Journal of Functional Programming, 29, 2019.
[4] Ramana Kumar, Rob Arthan, Magnus O. Myreen, and Scott Owens. Self- formalisation of higher-order logic. Journal of Automated Reasoning, 56:221–259, 2016.
[5] Ramana Kumar, Rob Arthan, Magnus O. Myreen, and Scott Owens. Hol with definitions: Semantics, soundness, and a verified implementation. In International Conference on Interactive Theorem Proving, 2014.

"Correct" - by correct we mean comply with properties according to its specification/definition, whichever applies. Actual correctness property can vary from one election to another, but should exist.