A formal security analysis of Blockchain voting

Nikolaj Sidorenco Laura Brædder **Lasse Letager Hansen**
Eske Hoy Nielsen Bas Spitters

AARHUS UNIVERSITY
Department of Computer Science, Aarhus University, Denmark

TYPES-2024

Can we trust online voting?

How do we ensure the security of online voting implemented by smart contracts?

EU DATA ACT SMART CONTRACTS

(30a) robustness and access control:

> *ensure that the smart contract has been designed to offer [. . .] a very high degree of robustness to avoid functional errors and to withstand manipulation by third parties.*

SWISS E-VOTING REGULATION

> Requires formal security proofs in the computational model

Why is voting relevant for blockchains

- Voting is used in blockchains for
  - consensus, governance, and decentralized organizations (DAOs).
- Moreover, for blockchains, the adversarial model is complex:
  - The adversary has complete knowledge of the system and full access to the network.
- The stakes are high, both financial and societal.

There can be bugs in

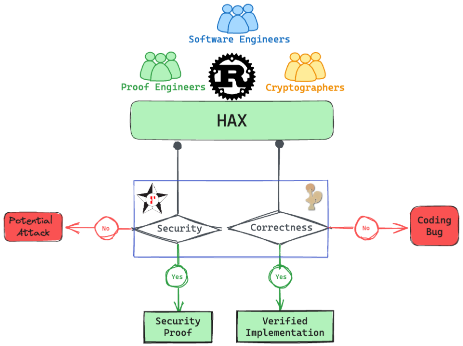
- the specification, the cryptographic proofs and/or the implementation.

Process

- Implement an executable specification of a protocol in safe Rust (Hax)
- Translate it into a proof assistant (Coq)
- Prove security properties (SSProve)
- Prove functional correctness and trace properties (ConCert)

Hax

- a subset of safe Rust with translations to proof assistants
- makes internet standards (e.g. IETF and NIST) machine-readable.
- executable specification in safe Rust
    - efficient implementation when building on the libcrux library of verified cryptographic primitives



SSProve

- a foundational framework for modular cryptographic proofs in Coq
- a language with monadic state and probability
- a program logic derived from the categorical Dijkstra monad framework

SSProve

ConCert

- A smart contract certification framework in Coq
- Models an abstract account-based blockchain with pure smart contracts
- Verified extraction to $\lambda_\square$
    - connects to CertiCoq which has verified extraction to WebAssembly (WASM)

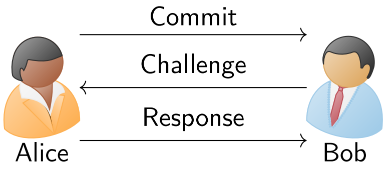
Online voting

Most online voting protocols use

- Commitment schemes
  - Allows one to commit to a chosen value while keeping hidden to others
- Zero Knowledge Proofs
  - A proof of knowledge about something without revealing anything extra

Zero-Knowledge Proof (ZKP):

- Schnorr
  - Proof that I know the exponents of an expression, without revealing them
- OR proof
  - Proof that I know one of two statements is correct, without revealing which.
  - e.g. vote is 0 or 1
- These examples are Σ-protocols
  - A three-step protocol



Σ-protocol - Security Properties

- Correctness of protocol
- Special Honest Verifier Zero-Knowledge (SHVZK)
  - A simulator that can construct a transcript given the response and challenge
- Simulation Sound Extractability
  - An extractor that can construct the witness given two valid runs of the same commit

Open Vote Network (OVN)

Protocol (using a group where the Decisional Diffie-Hellman (DDH) problem is hard):

- Round 1 (register vote):
  - Each public key is put on the blockchain and committed to (Schnorr ZKP)
- Round 2 (commit to vote):
  - Verify commitment from round 1
  - Compute commitment to vote
- Round 3 (cast vote):
  - Build an OR-proof (0 or 1) and cast vote
- Round 4 (tally):
  - Verify the OR-proofs and commitments
  - Tally result

Open Vote Network (OVN)

Properties

- Self-tallying: After all ballots have been cast anyone can compute the result
- Maximum ballot secrecy: Each ballot is indistinguishable from random input
- Universal verifiability: Anyone can verify the protocol was done correctly

Security

- Commitment (SSProve)
- Schnorr ZK protocol and the OR-construction (SSProve)
- Functional correctness (ConCert)

Related work

Verification process

- EasyCrypt: Not foundational

Unmaintained, but part of the inspiration for SSProve:

- CertiCrypt, Foundational Cryptography Framework (FCF), CryptHOL

Symbolic proofs and provers: Doable in Hax

- using e.g. Squirrel, Tamarin

Alternative voting protocol: ElectionGuard

- is more off-chain but uses similar building blocks.

Conclusion

- Industrial application of type theory
- First time showing both the correctness and security of a smart contract
- Illustrate possibilities for formal methods as requirements of online voting
- Can be made efficient with Libcrux library of verified crypto primitives