A formal security analysis of Blockchain voting

Nikolaj Sidorenco
Aarhus University
Denmark

Laura Brædder
Aarhus University
Denmark

Lasse Letager Hansen
letager@cs.au.dk
Aarhus University
Denmark

Eske Hoy Nielsen
eske@cs.au.dk
Aarhus University
Denmark

Bas Spitters
spitters@cs.au.dk
Aarhus University
Denmark

Abstract

Voting and blockchains are intimately connected. Voting is used in blockchains for consensus, governance, and decentralized organizations (DAOs). Conversely, blockchains can serve as a bulletin board for smaller elections. For these applications, the stakes are high, both financial and societal. Moreover, for blockchains, the adversarial model is complex: the adversary has complete knowledge of the system and full access to the network.

Here we focus on the use of blockchains for smaller elections; so-called boardroom voting. We consider one such protocol: the Open Vote Network (OVN), which provides private voting on a blockchain. Such private voting could also be applied to a DAO, improving the state of the art where voting for DAOs is often pseudonymous instead of private.

The OVN protocol has previously been implemented as a smart contract on Ethereum. It comes with an informal security argument. We propose a more complete analysis, which we formalize in the Coq proof assistant. We thus provide, *for the first time*, a complete analysis of a cryptographic smart contract, which is both provably functionally correct and cryptographically secure. We are extending this to a more realistic implementation.

Keywords: Coq, Smart Contract, Open Vote Network, Hacspec

1 Introduction

Digitization is facilitating many aspects of our society, and voting is no exception. However, in voting the stakes are higher than in many other applications. Moreover, elections require trust for their outcome to be accepted. Blockchains provide a trusted bulletin board, and as such, they have been used as part of voting protocols. Conversely, online voting also serves an important role in the blockchain ecosystem itself. Online voting is used in some modern blockchains for making fundamental changes to the system — so-called blockchain governance; see [6] for an overview.

Online voting systems have, through time, had buggy implementations and questionable security guarantees. Blockchains can solve some of these issues but also raise a number of new questions; see [9] for a description of the many issues with blockchain voting. Two critiques of blockchain-based solutions are their security and the quality of their implementation.

We will provide a means to addressing these issues by focusing on a particular blockchain voting protocol: the Open Vote Network (OVN) [3].

The OVN allows a small group (of say N participants) to vote anonymously.

To perform the vote, all participants must first agree on a public encryption key. The encryption key in the OVN has the following property: when multiplying N values encrypted with the key, one can efficiently uncover the unencrypted product. Moreover, recovering an unencrypted product of M values is only feasible if $M = N$.

The OVN has been implemented as a smart-contract on the Ethereum blockchain [7]. Such a smart contract implementation raises two questions; first is the protocol implemented correctly as a smart contract, and second is the protocol cryptographically secure.

A partially verified implementation of OVN is already available in ConCert [1]. We are extending this work to a more realistic implementation with stronger guarantees and fully verifying it, including proving it cryptographically secure.

Notice that many of the tools we implement to prove the security of the OVN protocol can be reused in proving security for other bigger protocols such as ElectionGuard[1]. For instance, the privacy of ElectionGuard relies, as in OVN and multiple other protocols, upon non-interactive zero-knowledge proofs. ElectionGuard uses both Schnorr proofs and a CDS construction which are made non-interactive via the Fiat-Shamir heuristic. All of this is implemented in our verified implementation of the OVN protocol, and thus we can recycle this code if we want to make a verified implementation of, for instance, ElectionGuard.

We have presented Coq as a framework for proving correctness and security of real-world Rust code and in particular smart contracts.

2 The Open Vote Network Protocol

The Open Vote Network(OVN) is a voting protocol based on ledgers and zero-knowledge proofs [3]. It is decentralized and does not require trusted parties. The OVN protocol achieves this by applying zero-knowledge proofs to detect any deviation from the protocol. These proofs are publicly available in a decentralized manner through the use of a blockchain. The verifiability of the votes depends on brute-forcing the inverse of a given function.

CoqPL'24, January 20, 2024, London, United Kingdom

[1] https://www.electionguard.vote/

The protocol as described in [3], proceeds in two rounds followed by a tallying process.

Round 1: Participants compute public keys, reconstruction keys, and a non-interactive zero-knowledge proof of the relation between their private and public keys. All public keys are put on the public ledger and verified by the participants.

Round 2: Every participant computes their encrypted ballot and a non-interactive zero-knowledge proof of the validity of the vote. Both the ballot and the zero-knowledge proof are published to the ledger.

Tallying: When each ballot is on the ledger and every zero-knowledge proof is checked to be valid, the votes can be tallied. To tally the votes participants compute the product of all encrypted votes. Tallying the votes can be done by anyone. Every vote is publicly available on the ledger and the tallying process requires no information private to the casters of the votes. Hence, the only barrier to computing the tally is the computational requirement of brute-forcing the exponent.

2.0.1 Commitment Phase. Now, OVN is a self-tallying protocol. Unfortunately, this has a fairness drawback seeing as the last participant to cast their vote can compute the tally before everyone else. This results in an adaptive issue meaning that knowledge of the outcome of the tally may potentially influence the voter's cast vote. To remedy this, we will use the approach described in the implementation of OVN in Ethereum [7] which adds an extra round where participants commit to their votes before disclosing the encrypted ballots.

3 Verifying OVN

We base our OVN implementation on the three-round Ethereum OVN implementation [7]. We chose to implement it in Hacspec [2, 8], which is a functional subset of Rust, as Rust is becoming an increasingly popular choice of smart contract language in blockchain. Furthermore, Hacspec being a functional subset of Rust means that the implementation lends itself well to verification in Coq.

Hacspec is a high-assurance cryptography specification language, its main use is to specify cryptographic primitives [2, 8]. However, here we build on ongoing work extending Hacspec for smart contract specification and implementation. One of the strengths of Hacspec is that it is built to be able to translate to multiple different backends. So by using Hacspec we automatically get translations and embeddings to a host of different tools.

We verify this implementation for both functional correctness and cryptographic security. The general approach is shown in figure 1. The OVN implementation in Hacspec is translated using multiple backends to both the ConCert [1] and SSProve [5] tools which are used to prove functional correctness and cryptographic security respectively. Additionally, an equivalence proof of the two implementations is automatically generated by the SSProve Hacspec backend.

Hao et al. [3] give the following security requirements for the OVN protocol:

Definition 3.1 (Maximum ballot secrecy)**.** Each ballot is a ciphertext indistinguishable from random. Thus, revealing nothing about the vote it contains.

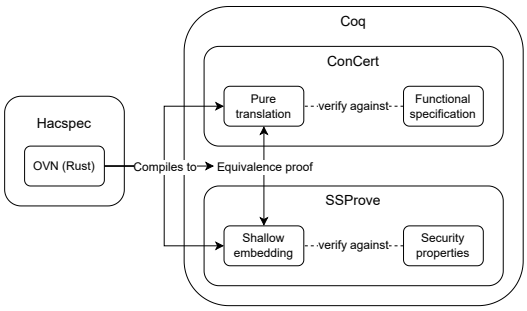

Figure 1. Verification pipeline

Definition 3.2 (Dispute-freeness)**.** The result should be publicly verifiable. Anyone can check whether all parties adhered to the protocol when casting their ballot.

Definition 3.3 (Self-tallying)**.** After all ballots have been cast, anyone can compute the result without external help.

We prove that OVN satisfies maximum ballot secrecy and self-tallying. Dispute-freeness is given directly by the use of zero-knowledge proofs in the protocol which are publicly available on the ledger. This proof has not been formalized yet.

3.1 Cryptographic Security

We prove cryptographic security using the SSProve framework, a framework for state separating cryptographic proofs in Coq [5]. Using the SSProve backend for Hacspec [4] we obtain a shallow embedding of OVN in SSProve which we use to define security games showing indistinguishability of the OVN implementation and an ideal implementation. This is done in the computational model, which is more precise than the symbolic model. The proof of maximum ballot secrecy follows a 'game hopping' style of proof, where the security of the protocol is reduced to well-established cryptographic properties of the assumptions.

3.2 Functional Correctness

Besides proving cryptographic security we also want to show that the functions making up the protocol are functionally correct and that as a whole the smart contract implements the protocol. We prove this using ConCert, which is a framework for smart contract verification in Coq [1]. ConCert abstractly models a blockchain as an immutable append-only ledger. Smart contracts in ConCert are modeled as state-passing functions in Coq.

An OVN implementation in ConCert is obtained using the Coq backend of Hacspec [10].

The blockchain model in ConCert models full execution traces allowing one to state and prove many interesting properties about smart contracts, such as trace properties, invariants over the contract's state, and interactions between contracts. The Self-tallying property of OVN is stated as an invariant over the state of the contract stating that for any reachable state in a valid execution trace, it should be the case that if all followed the protocol then a tally can be computed from the public data in the contract state and that tally equals the sum of all votes.

References

[1] Danil Annenkov, Jakob Botsch Nielsen, and Bas Spitters. 2020. ConCert: A Smart Contract Certification Framework in Coq. In *Proceedings of the 9th ACM SIGPLAN International Conference on Certified Programs and Proofs*. nil. https://doi.org/10.1145/3372885.3373829

[2] Karthikeyan Bhargavan, Franziskus Kiefer, and Pierre-Yves Strub. 2018. hacspec: Towards Verifiable Crypto Standards. In *Security Standardisation Research - 4th International Conference, SSR 2018, Darmstadt, Germany, November 26-27, 2018, Proceedings*. 1–20. https://doi.org/10.1007/978-3-030-04762-7_1

[3] F. Hao, P. Ryan, and Piotr Zielinski. 2010. Anonymous Voting by Two-Round Public Discussion. *IET Inf. Secur.* (2010). https://doi.org/10.1049/IET-IFS.2008.0127

[4] Philipp G. Haselwarter, Benjamin Salling Hvass, Lasse Letager Hansen, Théo Winterhalter, Catalin Hritcu, and Bas Spitters. 2023. The Last Yard: Foundational End-to-End Verification of High-Speed Cryptography. Cryptology ePrint Archive, Paper 2023/185. https://eprint.iacr.org/2023/185 https://eprint.iacr.org/2023/185.

[5] Philipp G. Haselwarter, Exequiel Rivas, Antoine Van Muylder, Théo Winterhalter, Carmine Abate, Nikolaj Sidorenco, Cătălin Hriţcu, Kenji Maillard, and Bas Spitters. 2023. SSProve: A Foundational Framework for Modular Cryptographic Proofs in Coq. *ACM Trans. Program. Lang. Syst.* 45, 3, Article 15 (jul 2023), 61 pages. https://doi.org/10.1145/3594735

[6] Aggelos Kiayias and Philip Lazos. 2022. SoK: Blockchain Governance. https://doi.org/10.48550/ARXIV.2201.07188

[7] Patrick McCorry, Siamak F. Shahandashti, and Feng Hao. 2017. A Smart Contract for Boardroom Voting with Maximum Voter Privacy. In *Financial Cryptography and Data Security (Lecture Notes in Computer Science)*, Aggelos Kiayias (Ed.). Springer International Publishing, Cham, 357–375. https://doi.org/10.1007/978-3-319-70972-7_20

[8] Denis Merigoux, Franziskus Kiefer, and Karthikeyan Bhargavan. 2021. *hacspec: succinct, executable, verifiable specifications for high-assurance cryptography embedded in Rust.* Technical Report. Inria. https://hal.inria.fr/hal-03176482

[9] Sunoo Park, Michael Specter, Neha Narulaand, and Ronald L Rivest. 2021. Going from bad to worse: from Internet voting to blockchain voting. *Journal of Cybersecurity* 7, 1 (2021).

[10] Mikkel Milo Rasmus Holdsbjerg-Larsen, Bas Spitters. 2022. A Verified Pipeline from a Specification Language to Optimized, Safe Rust. CoqPL. https://cs.au.dk/~spitters/CoqPL22.pdf