E-Voting Wasm Cryptography

David Ruescas and Eduardo Robles

Sequent
`david,edu@sequentech.io`
http://https://sequentech.io/

Abstract. The Sequent Voting Platform is an open-source E2EV internet voting system currently used in private organisations and non-legally binding elections of public organisations. The system employs standard cryptographic techniques following in the steps of well-established voting schemes proposed in the academic literature.
We demo core cryptographic components that are being developed for the next generation of Sequent's platform. The main novelty demonstrated is the execution of (heavyweight) cryptographic operations in the browser, in a performant way. Potential applications of this technique are listed and possible benefits for security, privacy and verifiability are suggested.

Keywords: evoting · cryptography · webassembly.

1 Introduction

Like many other systems proposed in the literature, the closest ancestor in Sequent's genealogy tree is Helios[1] in its original mixnet variant. The most significant departures from that Helios design are the use of a threshold distributed key generation mechanism, described in Pedersen[2] and featured in CGS[3] and Distributed Helios[4], and the use of a Terelius-Wikstrom[5] style mixnet rather than the Sako-Kilian[6] one. Other systems with which Sequent shares techniques are Wombat[7] and CHVote[9].

Research and development into Sequent's next generation system is currently underway. Part of this effort has been centred around the use of Rust[10] as a core technology. One of the interesting aspects of this technology is its ability to target WebAssembly[11] through the LLVM[12] toolchain.

Internet voting systems require the use of a client component with which voters select and encrypt their votes, typically in a browser. In the past, these components have been written in javascript or related languages. These components replicate some of the cryptography (for example, ElGamal encryption) that later processes votes in the backend. The initial motivating factor for our investigation of Rust's WebAssembly target was the possibility of merging this overlapping cryptography into a single unified codebase. But there are further interesting possibilities.

2 Applications

2.1 Vote casting

Voting client software can reuse common cryptography packaged in a library compiled to wasm, eliminating duplication.

Suggested benefits

– Security: A unified code base reduces the likelihood of mismatches between client and server cryptography, and reduces the attack surface. The amount of code that needs to be audited is also reduced.
– Performance: Higher performance compared to javascript implementations.

2.2 Ballot verification

Ballot verifiers implementing the Benaloh challenge can reuse common cryptography packaged in a library compiled to wasm, eliminating duplication.

Suggested benefits

As above.

2.3 Election verification

Election verification, usually carried out by specialised software that must be downloaded and configured, can be executed in the browser with no installation.

Suggested benefits

– Verifiability: Making election verification procedures significantly more usable can achieve higher rates of exercised verification, moving the "universal" part of universal verifiability closer to practice.

Note that achieving performant implementations in this use case is particularly difficult as election verification involves compute intensive operations that a priori seem impossible in a browser. We have not listed performance as a benefit here as we are comparing with non-browser, native implementations; in other words, performance is a must-have rather than a benefit for this use case.

2.4 Trustee protocols

Running full trustee nodes on the browser with reduced deployment, administration and training costs.

Suggested benefits

Real world experience has taught us that one of the barriers to running mixnet-based elections with a larger number of independent trustees is the cost that these trustees must incur in terms of deployment, administration and training. This is especially true for elections with fewer resources in human capital and infrastructure. As a result, it is not always easy to procure independent trustees to assume this important responsiblity.

Any objective that is presumably achieved through distribution into independent trustees could be achieved to a greater degree when some of the costs of this distribution are reduced. For example:

- Privacy: Ballot secrecy safeguards achieved through the distribution of private key material and mixing permutations would be achieved to a higher degree if more trustees participate.
- Security: Correctness safeguards achieved through distribution of mixing and tallying would be achieved to a higher degree if more trustees participate.

See previous section regarding performance as a benefit.

References

1. Ben Adida. Helios: Web-based Open-Audit Voting.
2. Torben Pedersen. Non-interactive and information-theoretic secure verifiable secret sharing.
3. Ronald Cramer, Rosario Gennaro, Berry Schoenmakers. A secure and optimally efficient multi-authority election scheme.
4. Véronique Cortier, David Galindo, Stéphane Glondu, Malika Izabachène. Distributed ElGamal à la Pedersen: Application to Helios.
5. Björn Terelius, Douglas Wikström. Proofs of Restricted Shuffles.
6. Kazue Sako, Joe Kilian. Receipt-free mix-type voting scheme — a practical solution to the implementation of a voting booth.
7. Wombat Voting https://wombat.factcenter.org/
8. Niko Farhi. An Implementation of Dual (Paper and Cryptographic) Voting System. http://www.cs.tau.ac.il/ amnon/Students/niko.farhi.pdf
9. Rolf Haenni, Reto E. Koenig, Philipp Locher, Eric Dubuis. CHVote Protocol Specification. https://eprint.iacr.org/2017/325
10. https://www.rust-lang.org/
11. https://webassembly.org/
12. https://llvm.org/