Decentralised Private Voting

December 21, 2022

Abstract

There are various existing options for decentralised private voting. This document reviews the options, the solutions, and gives some opinions on which type is best in which situation. It also links to examples of implementations where they exist, and briefly outlines some potential future research directions.

1 Introduction

There are several ways to achieve secure private (and decentralised) voting, which we will go through within this document. Decentralisation is being stressed in the title of this document and within it, due to the fact that many voting schemes that exist in the current body of work rely on a trusted party for vote integrity, or voter privacy, or for a combination of these two alongside availability of the voting system itself. Helios [Adi08] is one such voting scheme, which is used for private voting in, for example, IACR elections[1]. However, in Helios an authority needs to be trusted for privacy. We do not wish for this to be the case in the scheme that we adopt.

We will be using an Ethereum smart contract as a bulletin board and vote counter, but a constraint imposed by this model is that the contract cannot have any private state at all, which means that any system in which the 'authority' needs to generate some private randomness is not available to use in this situation. This means that we need to provide a protocol where anyone who sees the votes can compute the final result themselves, without being able to reveal any information about the votes (except that which is revealed by the final vote itself). A final constraint is that computation is very expensive on Ethereum, and so we desire a protocol with a verification step that is as lightweight as possible. There are existing papers that provide a thorough overview of currently implemented and deployed cryptocurrency based voting schemes[2].

2 Private linkable membership proofs

Private linkable membership proofs achieve voter anonymity though having indistinguishability of voters, with the votes themselves in the clear. This means that different types of voting are easily added, without having to make any more decisions relating to cryptography, or re-program any components that are touching on cryptography. This is because the votes are always available in the clear, and so any desired function or weighting of the votes is applied the same way it would be for a non-private voting protocol.

The protocols discussed below are explained through the lens of having a 'one person, one vote' scheme, but it is easy to extend this type of solution to support different voting functionalities. Examples of different voting models that are easily supported here include arbitrary non-linear voting, and different subgroups having different voting roles. For the case of different subgroups of voters having different voting roles, for example if different groups have different weights, anonymity holds within each type of voter – the votes of voters with different roles *are* distinguishable from one another here.

A limitation of choosing voter anonymity with votes in the clear is that if the goal is to do token-weighted voting (or any other non-standard weighting of votes), the voters may be able to be identified by their number of tokens, which, in the worst case, makes the cryptographic anonymity redundant. This issue has been seen before in both Monero [SALY17, KFTS17], and ZCash [HBHW16, KYMM18], because the amount of money that was being transferred in transactions used to be in the clear, and so though the sender of the money thought that they had

[1] `https://www.iacr.org/elections/2021/`

[2] `https://monami.hs-mittweida.de/frontdoor/deliver/index/docId/13092/file/DruckversionNomanaAyeshaMajeed.pdf`

anonymity, the transaction value acted as an identifier. This can be circumvented either through having different discrete token values, with each pool of that value forming an anonymity set, or it could be mitigated through use of self-tallying encrypted voting, as explored in section 5.

Private membership proofs don't reveal which voters voted, and don't reveal what each voters voted for. They are usable today, and can be very fast to generate, including on phones. Theoretically optimal proving time, verification time, and signature size are all $O(1)$, though those times and sizes have only been reached in practice with some important caveats. End-to-end verifiability is built into the protocol, and having votes in the clear means that verification of the vote outcomes is trivial, with no use of cryptography or specialist software or knowledge required. Protocols are flexible — to support a different type of voting, you don't need to do any additional cryptography, due to the votes being in the clear. In terms of drawbacks, different voting schemes or token based weighting schemes may introduce privacy concerns. The most efficient existing schemes that don't rely on a trusted authority have an $O(n)$ one-time, per-census pre-processing verification cost, and then $O(log(n))$ per vote additional verification cost. They are $O(log(n))$ in size, and proving time is $O(n)$.

2.1 Instantiations

In all of the options below, the protocol works through voters solving an equation solvable only by people who have a secret key corresponding to one of the public keys in the census. Voters then use this proof to sign a message. The message is not encrypted, and the content of the message is the voter's vote. Domain-separation information such as chain ID and process ID can also go into the message in order to provide replay protection.

2.1.1 DualDory

DualDory [BEHM22]: Logarithmic-verifier linkable ring signatures through pre-processing. This is an elliptic curve discrete log-based scheme, with pairings-based pre-processing. This scheme was introduced in mid 2022, and a huge plus is that it is implementable and deployable today, with no future research needed. We would simply have to implement the protocol in Rust, and the verification section in Solidity, and see how concretely expensive it is. DualDory has $O(n)$ voter computation time, signatures of size $O(log(n))$, and verification has a $O(n)$ pre-processing stage (one per census), and then an additional $O(log(n))$ verification computation required per vote. Author's implementation in Go can be found at `https://github.com/yacovm/DualDory`. This paper is based on another paper, called Dualring, which works without the verifier pre-processing stage, and so without pairings-based assumptions. A Dualring implementation in python can be found at `https://github.com/thyuen/dualring`.

2.1.2 Short Group Signatures

Also known as BBS signatures, these signautres were introduced by Boneh, Boyen, and Shasham in 2004 [BBS04]. This is a pairings based scheme, with constant size signatures and proving and verification complexity. This means that the signature size, proving and verification computations do not grow at all with the number of voters, meaning that the can scale indefinitely, supporting censuses the size of the global population. This scheme is currently going through the cfrg standardisation process[3], as it is being used in companies that are trying to provide privacy preserving proof of identity solutions. This means it should be straightforward to implement well.

This issue with this solution is that currently it is linkable only to an authority. Interesting future research could determine whether this authority could be easily distributed or replaced with some publicly verifiable equations.

A rust implementation of an extension of BBS signatures called 'BBS+ signatures' can be found at `https://github.com/mattrglobal/bbs-signatures`, with its documentation found at `https://docs.rs/bbs/latest/bbs/`.

2.1.3 Short linkable ring signatures revisited

This is a strong RSA and elliptic curve DDH based scheme, again achieving constant sized signatures and proving and verification computations [ACST06]. However, use of RSA assumptions means that this scheme would need a bespoke trusted setup, and the difficulty of distributing this work in a privacy-preserving way is an open research problem, though probably a matter of several

[3] `https://identity.foundation/bbs-signature/draft-irtf-cfrg-bbs-signatures.html`

months rather than several years worth of research. There are also some doubts on whether the strong RSA assumption holds when used like this, and so more research could be needed to ensure that the protocol relying on the assumption is well founded.

3 SNARK-based private membership proofs

With SNARK-based private membership proofs, the voter again proves that they are entitled to vote without revealing exactly which party in the census the vote corresponds to. However, in contrast to the proofs discussed in the section above, here we have the trade-offs of using SNARKs rather than a lighter-weight, special purpose membership proof. Using SNARKs means that arbitrary functionality can be added by developers. The signature size is constant, and the verification time consists of a pre-processing per-circuit operation that is polynomial in the size of the circuit, and then an $O(1)$ additional verification cost per signature. The proving time is polynomial in the circuit size, which concretely can be very expensive. There are also setup requirements, including generally needing a large MPC ceremony in order to trust that the soundness properties of this scheme hold in practice.

3.1 Instantiations

3.1.1 Vodconi

Vocdoni has implemented a SNARK based private voting solution, which is currently being put into production. The scheme works as described above (each individual wishing to vote forms a SNARK proving that they appear in the census, and submit this along with their vote). Documentation of the protocol can found at `https://docs.vocdoni.io/architecture/protocol/anonymous-voting/zk-census-proof.html` with an implementation at `https://github.com/vocdoni/zk-franchise-proof-circuit`.

3.1.2 Lookup tables with sublinear prover costs

Some interesting research on lowering the cost of lookup tables inside of SNARKs could be used for making membership proofs with sublinear (in the number of voters) computational complexity for the voter [ZBK$^+$, GW]. The verification time and proof size would stay unchanged at constant per vote, but decreasing the proving time could make private SNARK-based voting accessible to a wider audience. There is no implementations of membership proofs using these protocols, so it would be interesting future work to see how much they reduce the prover costs in practice.

4 Plain digital signatures + tallyable encryption of votes

With tallyable encrypted votes, voters themselves are not indistinguishable from one another, but their votes remain private even after the results of the vote are computed and published. In this model, voters are not anonymous, but their votes are counted without ever being decrypted, and the results of the votes are released without revealing any information about who voted in which way, except that which is revealed by the output of the vote itself.

Because the cryptography here is in the computation of the votes, this type of protocol supports different voting functionalities in a different way. Each functionality would need to have its own cryptographic protocol designed, implemented (and hopefully audited) before its launch, but due to the votes remaining encrypted throughout the entire lifetime of the protocol, the weight- or voter-role-based privacy risks highlighted in the 'Private linkable membership proofs' case are no longer present[4].

5 Private linkable membership proofs + tallyable encryption of votes

This category of solution doesn't reveal which voters voted, or what any voter voted for, they are usable today and eliminate the privacy risks listed in the private linkable membership proofs section. They can have very fast proving and verification times, and small proof sizes. End-to-end

[4] I don't know of any protocols that fit directly into this category – mostly they are found instead in the category above, or below.

verifiability can be built in to the protocol, and different voting / token based weighting schemes can be supported without introducing the same privacy concerns as listed above. However, protocols are still not flexible in terms of easy support of new voting protocols. To support a different type of voting, a new scheme would need to be designed and implemented from scratch, including possibly complicated proofs of security. Also, use of cryptography means it's not trivial to verify, which may be desired in some settings.

5.1 Instatiations

5.1.1 A Smart Contract for Boardroom Voting with Maximum Voter Privacy

In 2017, a self tallying voting protocol based on the discrete log problem was implemented[5] and a paper describing the protocol and its implementation costs on Ethereum was published [MSH17]. Self tallying voting protocols have the limitation that if the last voter is malicious, they could compute the total count of votes without including their vote, and then choose whether to contribute their vote, or whether to abort, potentially causing the whole protocol to abort. We can use cryptography to get around this limitation, but it would require all other voters to come back online and communicate with one another to reconstruct the result, which seems undesirable. Another way around this (which they choose in the paper) is to first have voters participate in a deposit stage, in which they deposit some funds and commit to voting, and then a voting stage, where they contribute their vote and the funds they have previously deposited are released back to them. Involving funds in this way incentivises all of the voters to participate, but introduces the issue of two-round voting, which it may be desirable to avoid. In this protocol, the costs for the election administrator scale based on the number of participants in the census, but the costs per voter are constant.

The protocol as implemented can only handle 0/1 votes, but is based on a paper that offers ways to extend that to multiple voting choices [HRZ10]. It was also implemented and gas costs estimated before Ethereum introduced precompiles for elliptic curve arithmetic, and so an implementation of the extension and calculating updated figures for the pricing of voting would be interesting future work[6].

5.1.2 Reputable List Curation from Decentralized Voting

This is a token curated registry construction based on decentralised, provably secure voting [CMMM20]. Its security is based on the DDH and square DDH in the restricted random oracle model, and can be run without needing any trusted setup. The scheme in this paper is the first provably concurrently secure self-tallying voting protocol. An implementation of this scheme would again be a worthwhile future direction.

As above, self tallying voting protocols have the limitation that if the last voter is malicious, they could compute the total count of votes without including their vote, and then choose whether to contribute their vote, or whether to abort, potentially causing the whole protocol to abort. We can use cryptography to get around this limitation, but it would require all other voters to come back online and communicate with one another to reconstruct the result, which seems undesirable. Another way around this (which they choose in the paper) is to first have voters participate in a deposit stage, in which they deposit some funds and commit to voting, and then a voting stage, where they contribute their vote and the funds they have previously deposited are released back to them. Involving funds in this way incentivises all of the voters to participate, but introduces the issue of two-round voting, which it may be desirable to avoid. In this protocol, the costs for the election administrator scale based on the number of participants in the census, but the costs per voter are constant.

6 Commit and reveal

Commit and reveal does not achieve any notions of privacy, besides *possible* obfuscation of votes during the commitment stage. We are including it here as recommended due to its possible greater suitability that other schemes listed in some specific settings where privacy is not desired. An example is ConsenSys' Partial Lock Commit Reveal voting[7].

[5] `https://github.com/stonecoldpat/anonymousvoting`

[6] This work is also not concurrently secure, and concurrent security would be difficult to prove due to the forking lemma resulting in exponentially many forks when rewinding in this situation. Also interesting work to see if this has any practical consequences in this situation (i.e. does the protocol allow users to open concurrent sessions?).

[7] https://github.com/ConsenSys/ PLCRVoting

Commit and reveal has a commitment stage, and then afterwards a reveal stage. It achieves fairness, as defined below, and as the votes are revealed in the clear in the second stage, it easily achieves end-to-end verifiability. Every individual with a block explorer is able to check (a) that the people who voted were valid voters, and (b) how each voter voted. Automatic execution of, for example, smart contracts, can also be triggered by the result of the votes, as they are available on chain in the clear.

Another way of achieving this is to have consensus nodes generate a private key between them, and send the public key to all parties wishing to vote, who will then publish an encryption of their vote. The consensus nodes can then decrypt the votes, and publish the decryptions, or simply publish the private key to the blockchain so that all parties can decrypt for themselves. This could also trigger any desired executions based on the voting results.

The pros of using commit and reveal are that it is usable today, and very simple to implement. In terms of security properties, it achieves fairness of votes, and easily achieves end-to-end verifiability. It is a useful protocol to use in cases with delegation, or other important-to-verify situations, where privacy is not needed.

The cons of using commit and reveal are that it doesn't achieve normal notions of privacy or anonymity, as it reveals which voters voted, and their exact votes are associated with each voter, entirely in the clear.

7 Conclusion

In this document we have reviewed the main currently available options for private, decentralised voting, as well as listing existing implementations where available. In terms of recommendations of existing solutions If you're doing 'one person, one vote' based protocols, non-snark membership proofs seem to offer the best efficiency, while guaranteeing privacy. If the aim is token weighted voting, use a special purpose encrypted voting protocol, for example as explained in [CMMM20] or [MSH17]. These schemes are more complex in terms of implementation and computational complexity for the prover and verifier than non-snark membership proofs, but are required to maintain privacy while users are voting with different numbers of tokens. If the aim is to support arbitrary different voting functionalities (eg weighted, one person one vote, and non-linear tallying) all with the same overall framework, SNARK based voting solutions would be preferable, as using SNARKs minimises the infrastructure changes needed with a change of voting system.

References

[ACST06] Man Ho Au, Sherman SM Chow, Willy Susilo, and Patrick P Tsang. Short linkable ring signatures revisited. In *European Public Key Infrastructure Workshop*, pages 101–115. Springer, 2006.

[Adi08] Ben Adida. Helios: Web-based open-audit voting. In *USENIX security symposium*, volume 17, pages 335–348, 2008.

[BBS04] Dan Boneh, Xavier Boyen, and Hovav Shacham. Short Group Signatures. In *Advances in Cryptology – CRYPTO 2004*, volume 3152, pages 41–55. Springer Berlin Heidelberg, Berlin, Heidelberg, 2004. Series Title: Lecture Notes in Computer Science.

[BEHM22] Jonathan Bootle, Kaoutar Elkhiyaoui, Julia Hesse, and Yacov Manevich. DualDory: Logarithmic-Verifier Linkable Ring Signatures Through Preprocessing. In *Computer Security – ESORICS 2022*, volume 13555, pages 427–446. Springer Nature Switzerland, Cham, 2022. Series Title: Lecture Notes in Computer Science.

[CMMM20] Elizabeth C Crites, Mary Maller, Sarah Meiklejohn, and Rebekah Mercer. Reputable list curation from decentralized voting. *Proceedings on Privacy Enhancing Technologies*, 4:297–320, 2020.

[GW] Ariel Gabizon and Zachary J. Williamson. plookup: A simplified polynomial protocol for lookup tables.

[HBHW16] Daira Hopwood, Sean Bowe, Taylor Hornby, and Nathan Wilcox. Zcash protocol specification. *GitHub: San Francisco, CA, USA*, page 1, 2016.

[HRZ10] Feng Hao, Peter YA Ryan, and Piotr Zieliński. Anonymous voting by two-round public discussion. *IET Information Security*, 4(2):62–67, 2010.

[KFTS17] Amrit Kumar, Clément Fischer, Shruti Tople, and Prateek Saxena. A traceability analysis of monero's blockchain. In *European Symposium on Research in Computer Security*, pages 153–173. Springer, 2017.

[KYMM18] George Kappos, Haaroon Yousaf, Mary Maller, and Sarah Meiklejohn. An empirical analysis of anonymity in zcash. In *27th USENIX Security Symposium (USENIX Security 18)*, pages 463–477, 2018.

[MSH17] Patrick McCorry, Siamak F Shahandashti, and Feng Hao. A smart contract for boardroom voting with maximum voter privacy. In *International conference on financial cryptography and data security*, pages 357–375. Springer, 2017.

[SALY17] Shi-Feng Sun, Man Ho Au, Joseph K Liu, and Tsz Hon Yuen. Ringct 2.0: A compact accumulator-based (linkable ring signature) protocol for blockchain cryptocurrency monero. In *European Symposium on Research in Computer Security*, pages 456–474. Springer, 2017.

[ZBK+] Arantxa Zapico, Vitalik Buterin, Dmitry Khovratovich, Mary Maller, Anca Nitulescu, and Mark Simkin. Caulk: Lookup Arguments in Sublinear Time.