Anonymous Voting in DAO's

Fausto Uribe [*], Isaak Hernandez [†], Luann Jung [‡], and Peter Amenewolde [§]

Department of Electrical Engineering and Computer Science, Massachusetts Institute of Technology

May 10, 2022

Abstract

The current state of decentralized autonomous organizations (DAOs) involves transactions on a blockchain and voting schemes which are not anonymous. This presents numerous issues when it comes to confidentiality, voter influence, and voter turnout in DAOs. Due to the decentralized, immutable, and public nature of the blockchain, there are currently no clearly effective ways to achieve anonymous voting in DAOs. In this work, we present solutions for implementing a voting scheme in blockchains that allows for confidentiality, while maintaining integrity and security through the use of zero-knowledge arguments of knowledge and other cryptographic protocols. We propose three approaches and discuss the assumptions and limitations for each.

1 Introduction

Voting confidentiality is important. The concept of private, confidential voting is common place when it comes to political elections; however, in this digital age, voting visibility still seems to be a problem that has not been completely addressed. For some background, there exists certain groups of individuals called decentralized autonomous organizations (DAOs) that utilize some mechanism in order to collectively make decisions about their organization and the protocols and services they control, usually some program on a blockchain. A common governance voting mechanism that is utilized by these decentralized organizations is to assign a number of votes to each individual that is equal to the amount of governance tokens that one is in possession of. One core issue that

[*]fut@mit.edu
[†]isaak@mit.edu
[‡]luju@mit.edu
[§]peterame@mit.edu

exists in this mechanism is that one individual's votes are visible to everyone, due to the inherent public nature of blockchains. Contrast this to how private in nature political elections are and one can see how currently voting in a DAO is problematic. Thinking about the side effects of this public voting system, one can imagine how voters in the group may feel suppressed to vote for a side or direction they truly align with. Additionally, there are unintended side effects like voter intimidation that can arise since an adversary can theoretically look up one's voting track record and harass or coerce them into voting a certain way. So although the blockchain provides many benefits discussed in future section 2.1, the various negative influences on public voting behavior in DAOS should be addressed. In this paper we focus on the token mechanism of governance, and aim to present a set of theoretical solutions utilizing zero knowledge proofs and other cryptographic schemes that can provide a protocol for voting in decentralized autonomous organizations that preserves confidentiality, security and integrity while still being maintained on the blockchain.

2 Background and Related Work

2.1 Blockchain

A blockchain at its core is a digital database that stores a public record of all transactions in a given system. Data on a blockchain is recorded differently than in a standard database. Instead of holding the data in tables, a blockchain will group a certain fixed-size of data into a "block" and then connect (or "chain") the blocks together as new blocks are formed [Zhe+17]. Grouping data this way provides for a historical record of data that is timestamped with then it entered the blockchain. Additionally, since blockchains are immutable in nature – meaning that data of transactions or records can only be added and not deleted – there is a guarantee that the records are secure, true, and valid without the need of a trusted third party. Two aspects of a blockchain that we highlight for the purpose of strengthening DAO voting are decentralization and transparency. Decentralization is crucial since if the records of a DAO were stored in one central place, it would not be much of a DAO at all. Blockchain achieves this by spreading out and duplicating the blockchain among various nodes in a network. This way, even if one node is attacked, it will be obvious where the discrepancy lies and the blockchain will continue to reflect the true records. Transparency is a guarantee with the use of the blockchain since all transactions are public knowledge, but in an effort to enable vote secrecy we implement the use of smart contracts.

2.2 Zero-Knowledge Proofs

Zero-knowledge proofs are a set of two party proofs with a prover, P, and a verifier, V. The prover is then able to prove some information to the verifier without sharing any private information, such as passwords or secret keys.

Constructing a zero-knowledge proof requires both completeness and soundness [GMR85]. Completeness meaning given a true statement, the probability, p, that an honest prover can convince an honest verifier will be greater than the probability a dishonest prover can convince an honest verifier with something false. In order for the proof to be a valid one, $p \geq \frac{1}{2}$ must hold true and p should approach 1 as more prover-verifier interactions ensue.

$$\forall x \in L \ Pr[(P, V)(x) = 1] \geq p \tag{1}$$

Soundness meaning, except with small probability, a dishonest prover cannot convince an honest verifier a given false statement is actually true.

$$\forall x \notin L \ \forall P^* \ Pr[(P^*, V)(x) = 1] \leq 1 - p \tag{2}$$

2.3 Decentralized Autonomous Organizations (DAOs)

A decentralized autonomous organization or DAO for short is an organization with no central leadership. Decisions are made in a "bottom-up" manner, and they function with a strict set of rules that are defined within a smart contract in a blockchain [Coi]. A common mechanism for the governance of these organizations is one that is token based. In this system of governance individuals that possess "governance tokens" get an amount of votes that is equal to the amount of tokens they possess. These tokens are usually initially distributed to the developers of the contract pertaining to the DAO, but other rules can be defined within the contract such that they are distributed to accounts that meet some criteria. These tokens can also be traded or given away in the same way any other tokens would be in a blockchain. If one of the participants of the DAO would like to apply some change, they would submit a governance proposal which would then be voted on by all the other token holders [DJ].

2.4 Smart contracts

At its core, a smart contract is a program that runs on a blockchain where, given a set of specified rules, the contract will execute right when the rules are met. Similar to blockchain's properties, smart contracts are immutable meaning not even the creator or anyone else can change the program once the rules are set and it is on blockchain. Because of this, there is no need for a trusted third party here either since the contract rules are set and is completely transparent. User accounts can interact with a smart contract by submitting transactions that interact with a function defined in its code. Additionally, due to the nature of the blockchain, these transactions are irreversible [Entb].

2.5 Vote visibility on the blockchain

We will use the Ethereum blockchain to demonstrate an example of how a specific vote is visible to anyone. On Ethereum the governance protocol is usually implemented with a contract called Governor by OpenZeppelin. Taking a look at this contract we can verify that in order to cast a vote, you must call a function with plaintext parameters.

```
contract Governance {
    ...
    function castVote(uint256 proposalId, uint8 support) public
        virtual override returns (uint256) {
        address voter = _msgSender();
        return _castVote(proposalId, voter, support, "");
    }
    ...
}
```

Since this is a public function, calling it reveals both your identity and the type vote you have cast, not only to miners, but to anyone that is looking in the blockchain [Enta].

3 Proof of Status

Our initial approach for Anonymous Voting in DAO's involved concealing the identity of the voting user by using a secondary account. Consider a prover P that wants to cast votes based on the balance of a wallet W_a. P also controls a secondary wallet W_b. P must prove to a verifier V that they know the private key to W_a. In order to stay anonymous, P must only interact with V using the secondary wallet and not reveal the public or private key of the main wallet. In this way, P proves their status to cast their votes, without revealing who they are. We accomplish this using a Sigma Protocol Zero Knowledge Proof [Sch91].

3.1 Sigma Protocol

Let PK_* and SC_* be the public key and secret key of W_* where $* \in \{a, b\}$ using a standard RSA encryption scheme. $H(x)$ is a random oracle hash function. There exists a trusted third party T that will return a token $y = [H(W^b)]^{PK_*}$ for wallet $*$ if given PK_*, where W^b is the balance of the wallet. When presented with one of these tokens, T will verify whether it is valid or not. T can be given a value r with PK_* and can verify to a third party if a given value is of form u^r, where u is a valid token. For P to prove they control a wallet with balance g^{bal}, they do the following, interacting with V through W_b.

If $[H(g^{bal})]^s = y^c * t$, then the verifier believes P controls a wallet with balance g^{bal}. The strength of this proof comes from the difficulty in computing discrete logarithm. The verifier believes the proof because it would be computationally infeasible for the prover to compute $s = r + SK_a * c$ such that $[H(g^{bal})]^s = y^c * t$ without knowing the secret key. The verifier knows whichever secret key they have controls a wallet with a balance of g^{bal} because of the verification of y^r with the central authority.

Algorithm 1 Proof of Main Wallet Balance

```
**Input:** $V$ asks $P$ to prove they control a wallet with balance $g^{bal}$
1. $P$ requests $y = [H(g_a^{bal})]^{SK_a}$ from $T$ and passes random value $r$ to $T$
2. $P$ sends commitment $t = y^r$ to $V$
3. $V$ stores $t$ and verifies with $T$ that $t$ is composed of a valid token
4. $V$ sends random challenge $c$ to prover
5. $P$ responds with $s = r + SK_a * c$
6. Verifier checks if $[H(g^{bal})]^s = y^c * t$
```

3.2 Limitations

The main limitation of this approach is the use of trusted third party. The third party has to be trusted with the secret keys of all members of the organization, and trusted to relay the correct information on valid tokens to the verifier. This limitation makes it infeasible to implement in DAO's as currently constructed, but further work could be done to replace T with a decentralized, or perhaps obfuscated alternative.

Another limitation to this approach is the ability to prove you own a wallet multiple times. The proof keeps the verifer from learning the public key of the wallet being proved. An adversarial P could prove they own main wallet W_a using two separate alternate wallets W_b and W_c. The verifer would not know the proofs were on the same main wallet. This shortcoming would allow members of a DAO to vote as many times as they want, provided they have enough alternate wallets. Avoiding this limitation would be trivial to fix if you utilized the trusted third party. However, further research is needed to fix this consistency problem in a decentralized manner.

There are logical extensions to this approach that would be useful if the issues of decentralization are addressed. Currently, the proof is for wallets with exactly a balance of g^{bal}. This could be extended to working for values that are greater than or equal to g^{bal}, or less than or equal to g^{bal}.

4 Vote hiding through public key cryptography

As discussed in Section 3.2, a major drawback of the Proof of Status model for anonymized voting is that issues arise with preventing multiple votes from the same agent. This is because a Proof of Status approach involves hiding all the voters' identities, but not necessarily the votes themselves. Upon identifying this challenge, we began to consider an alternative approach for anonymized voting in DAOs.

We pivoted our scheme for anonymous voting from hiding voters to hiding votes. Our initial approach at this did not involve any form of zero-knowledge proof.

Instead, we employ the basic principles of anti-symmetric public key encryption. Specifically, we make use of a homomorphic encryption scheme $\mathcal{E}$ that supports addition. If $\mathcal{E}$ is homomorphic, that means we may perform some computations on the encrypted values without needing to decrypt them. We particularly are interested in additive homomorphism in algorithms such as Benaloh [CF85], Pailler [Pai99], Damgård–Jurik [DJ00], and other known partially homomorphic cryptosystems. Each of these is viable for use in this scheme as long as addition can be performed on encrypted vectors without compromising the correctness of the result.

For anonymous voting, we establish a smart contract on the blockchain (as described in Section 2.4) that receives vote vectors from voters. Each voter who owns some tokens in the DAO can send a vote vector encrypted using $\mathcal{E}$. The smart contract receives all of the voters' vectors and verifies their identities and whether they own tokens in the DAO (this is public knowledge and easily verifiable). Then, the smart contract aggregates the vote vectors and at some predetermined time t decrypts the final result and publicizes it. More details are described in Algorithm 2 and Figure 1.

For the purposes of our work, we assume that $\mathcal{E}$ refers to the Pailler cryptosystem. For key generation, one chooses two large prime numbers p and q randomly and independently of each other such that $gcd(pq, (p-1)(q-1)) = 1$. Then, compute $n = pq$ and $\lambda = lcm(p-1, q-1)$. Next, select a random integer $g \in \mathbb{Z}_{n^2}^*$ and ensure that n divides the order of g. This can be done by checking the existence of a modular multiplicative inverse $\mu = (L(g^\lambda \mod n^2))^{-1} \mod n$, where $L(x) = \frac{x-1}{n}$. Finally, the public key is (n, g) and the private key is (λ, μ).

To encrypt a message m where $0 \leq m < n$, one selects a random r such that $0 < r < n$ and $r \in \mathbb{Z}_{n^2}^*$. The ciphertext c is

$$c = g^m \cdot r^n \mod n^2.$$

To decrypt a ciphertext c where $c \in \mathbb{Z}_{n^2}^*$, compute the plaintext message m as

$$m = L(c^\lambda \mod n^2) \cdot \mu \mod n.$$

The Pailler cryptosystem [Pai99] described above has the homomorphic property that

$$\begin{aligned}\mathcal{E}(m_1) \cdot \mathcal{E}(m_2) &= (g^{m_1} r_1^n)(g^{m_2} r_2^n) \mod n^2 \\ &= g^{m_1+m_2}(r_1 r_2)^n \mod n^2 \\ &= \mathcal{E}(m_1 + m_2).\end{aligned} \tag{3}$$

Because of this, we can multiply all received encrypted vote vectors together and decrypting the final result will yield the sum of the unencrypted vote vectors.

Algorithm 2 Public Key Cryptography Voting

```
Input: Pailler homomorphic encryption scheme $\mathcal{E}$
1. An end time $t$ when voting will end is agreed on
2. Each voter $i$ comes up with a vote vector $\vec{v_i}$
3. Each voter sends $\mathcal{E}(\vec{v_i})$ to the smart contract
4. Smart contract verifies the identity of voter $i$
5. Smart contract aggregates all verified encrypted votes $\prod_i \mathcal{E}(\vec{v_i})$
6. At time $t$, contract decrypts $\sum_i \vec{v_i} = \mathcal{D}(\prod_i \mathcal{E}(\vec{v_i}))$ using private key
7. Decrypted result is publicized
```

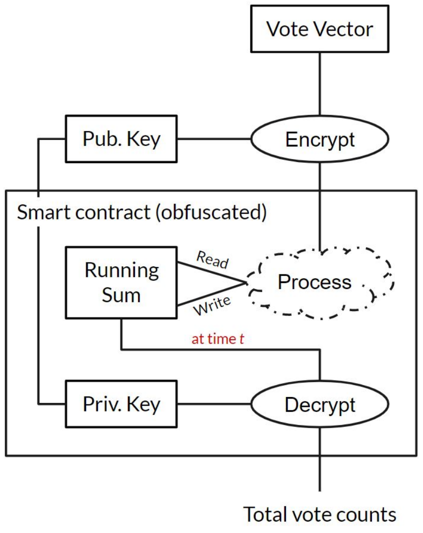

Figure 1: Concept for sending each vote that is encrypted via a partially homomorphic encryption scheme to a smart contract, which aggregates the votes and decrypts the end result.

4.1 Obfuscation

This scheme relies on one major assumption in order to work. Traditionally, smart contracts do not hold their own private key. This is because smart contracts are simply code run on the blockchain and anyone could theoretically download the code at any given point, decompile it, and attempt to gain information. Thus, we propose for the use of code and memory obfuscation in order to conceal the smart contract's private key.

Ideally, the entire workings of the smart contract would be indistinguishably obfuscated – this includes the states of variables and also the logic of the code. This means that given two equivalent programs that have been obfuscated, one cannot determine which of the two outputs came from which original source. A simple example of this concept is the following two programs:

- `return 0`
- `return sign(privkey, 0) - sign(privkey, 0)`

The first program just returns zero, while the other uses an internally contained private key to cryptographically sign a message, does that same operation another time, subtracts the (obviously identical) results from each other and returns the result, which is guaranteed to be zero. If we use an algorithm to obfuscate both programs, then computational indistinguishability is satisfied if the obfuscated programs cannot be distinguished from each other. Thus, someone in possession of an obfuscated program definitely has no way of extracting the private key.

Of course, our approach here would be limited by the strength of the obfuscation algorithm used. There are some promising candidates for obfuscation (such as [Sch+21]), but the field is still very much in development [But].

4.2 Limitations

All things considered, this approach for anonymous voting is not ideal. We acknowledge a couple of important limitations. First, if the private key for the smart contract is recovered at any point, every voter's entire vote is fully decryptable and no longer anonymous. Additionally, the scheme does not truly feel anonymous. Even in DAOs, one member must deploy the smart contract in the first place. Although they cannot modify it after deployment, there is still one person who produced the smart contract. In the presence of a adversarial member of the DAO, placing all of the votes as visible to the smart contract feels unsatisfying and not particularly safe from de-anonymization. Hence, we continue to propose an improvement to our idea of vote hiding through public key cryptography where we also implement secret sharing to help mitigate some of the aforementioned limitations.

5 Improvement on vote hiding through secret sharing

To improve on some of the shortfalls that are present in our previous schemes, we add another layer of complexity in order to make a more robust protocol for retaining ones vote privacy on the blockchain while it still being infeasible to cast an invalid vote. This protocol combines ideas from our previous schemes in the use of zero knowledge proofs and public key cryptography, as well as one of the elementary ideas of cryptography: the one time pad.

Our scheme works a lot like that of vote hiding through public key cryptography (as discussed in Section 4) with a few modifications. Instead of utilizing a single contract we use two contracts with different public-secret key pairs, each of which will receive one of our vote "shares" (explained in more detail later). These additive secret vote shares are encrypted with a public key and sent to the respective contract. Once voting finalizes, the voting counts are added to reveal the correct final vote count, without ever revealing any specific individuals vote.

Algorithm 3 Encrypted Secret Share Voting

```
Input: Homomorphic encryption scheme $\mathcal{E}$
1. An end time $t$ when voting will end is agreed on
2. Each voter $i$ comes up with a vote vector $\vec{v_i}$
3. The voter creates a random vector $\vec{r_i}$ of the same length as $\vec{v_i}$
4. The voter creates two vectors $\vec{u_i} = \vec{v_i} - \vec{r_i}$, $\vec{w_i} = \vec{r_i}$
5. Each voter sends $\mathcal{E}(\vec{u_i})$ to one smart contract, and $\mathcal{E}(\vec{w_i})$ to the other
6. Contracts zero-knowledge verify that the vote shares form a valid vote
7. Each smart contract aggregates received vote shares $\prod_i \mathcal{E}(\vec{u_i})$, $\prod_i \mathcal{E}(\vec{w_i})$
8. At time $t$, the contracts decrypt their counts and publicize them
9. The result from all contracts sum to the total vote count $\sum_i \vec{v_i}$
```

5.1 Creating the vote shares

To further hide our vote throughout the voting process in the blockchain, we will make use of randomness in order to create two vote shares that reveal nothing about an individuals vote when observed independently. We will add this randomness to the vote vector and in this way create a sort of one time pad (as we can only use this specific randomness vector once).

The original vote vector should be a vector of length two that is composed of only 0's and 1's, where each element represents a vote of YES or NO. For example the vector $\vec{v} = \begin{pmatrix} 0 & 1 \end{pmatrix}$ would represent a vote of YES and $\vec{v} = \begin{pmatrix} 1 & 0 \end{pmatrix}$ would represent a vote of NO, where the vectors $\vec{v} = \begin{pmatrix} 0 & 0 \end{pmatrix}$ and $\vec{v} = \begin{pmatrix} 1 & 1 \end{pmatrix}$ represent ABSTAIN. Now we define our random vector $\vec{r} = \begin{pmatrix} r_1 & r_2 \end{pmatrix}$ to be a vector of length 2, whose elements are uniformly random on the range $[-2^{31}, 2^{31}]$ such that it can be stored as a 32 bit integer.

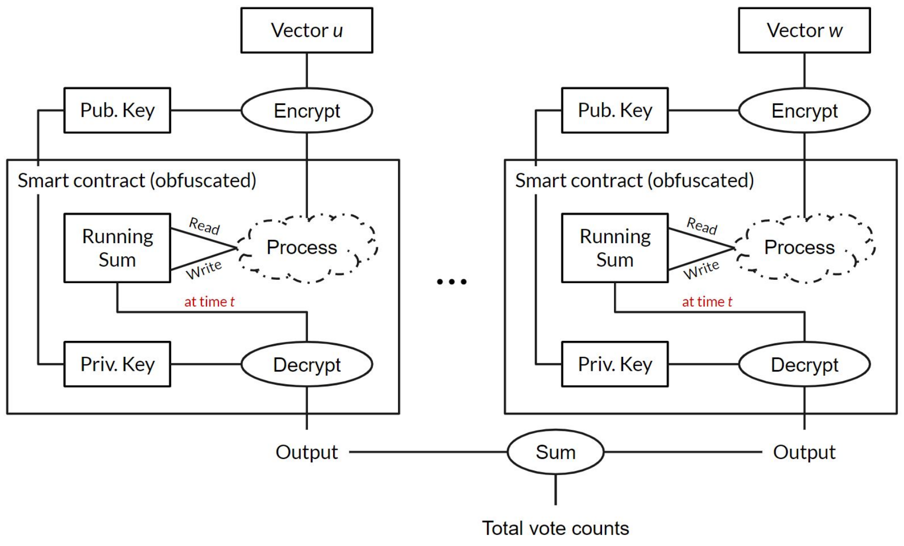

Figure 2: Concept of each voter splitting their vote into two or more vote shares and sending each to a separate smart contract that each have their own private and public key pair. Each smart contract aggregates the shares they receive and the final result is the output of each smart contract summed together.

We will split the original vote vector into two vote shares $\vec{u}$ and $\vec{w}$ in the following way:

$$\begin{aligned} \vec{u} &= \vec{v} - \vec{r} \\ \vec{w} &= \vec{r} \\ \vec{u} + \vec{w} &= \vec{v} \\ \implies \sum_{i=0}^{n} u_i + \sum_{i=0}^{n} w_i &= \sum_{i=0}^{n} v_i \end{aligned} \tag{4}$$

Thus we can send our vote shares to each of the contracts in a manner where they only keep a sum of what appears to be random vectors. However once they are summed together the final vector displays the correct vote count. We would continue to use the Pailler cryptographic scheme as discused in Section 4 to take advantage of partially homomorphic encryption.

5.2 Ensuring well formed vote vectors

Given that the information each contract is receiving appears to be random vectors, it would be easy to imagine how an adversary might be able to take advantage of this system by sending two vectors such that the sum does not correspond to a vector of the form $\{0, 1\}^2$. Hence, we demonstrate a zero knowledge proof protocol that allows the prover to prove to two verifiers that their vote shares add up to a well formed vote vector without revealing their actual vote. In other words we describe a scheme with the following properties:

Correctness $\forall x_A + x_B \in \{0,1\}^2, \Pr[< P, V_A, V_B > (x_A, x_B) = \text{accepted}] = 1$

Soundness $\forall x_A + x_B \notin \{0,1\}^2, \forall P^*$
$\Pr[< P^*, V_A, V_B > (x_A, x_B) = \text{accepted}] \leq \text{negl}$

Privacy/HVZK $\forall$ adv. contract A, view of contract A in $< P^*, V_A, V_B > (x_A, x_B) \longrightarrow x_A \sim \mathcal{U}$

$$\begin{aligned}
\text{Let } \vec{x} = {\scriptstyle (x_1 x_2 \ldots x_n)} &\in \mathbb{F}^n \\
\text{Define polynomials } f, g, h \text{ over } \mathbb{F} \text{ s.t. } &\forall i \in [n] \\
f(i) &= x_i \\
g(i) &= x_i - 1 \\
h &= f \cdot g \\
\text{If } x \in \{0,1\}^n, \forall i \in [n], h(i) &= 0 \\
\text{By same argument if } x \notin \{0,1\}^n, \exists i \in [n], h(i) &\neq 0
\end{aligned} \tag{5}$$

A key result of this scheme is that given additive shares $\vec{x_A}, \vec{x_B}$ each contract can evaluate shares $f(r)$ and $g(r)$ for $r \in \mathbb{F}$ without communicating with each other. Additionally, given shares of the coefficients of polynomial h, the contracts can compute $h(r)$ for $r \in \mathbb{F}$ without communicating. Thus in order to perform the zero knowledge argument, we simply follow these steps:

1. Client computes polynomial h and sends shares of the coefficients of h to contracts as proof
2. Contracts check that for all $i \in [n], h(i) = 0$
3. Contracts check that $f \cdot g = h$ by choosing a random r and checking that $f(r)g(r) = h(r)$

In this way the contracts ensure that the vote shares that they are receiving represent those of a valid vote vector and retain the integrity of the voting process, while also maintaining the privacy of the voters.

6 Discussion and Future Work

In this work, we proposed three theoretical approaches to tackling the concept of anonymized voting in decentralized autonomous organization on the Blockchain. We discussed the inspiration, assumptions, and limitations associated with each one.

Our first approach focused on the idea of hiding all voters' identities, but not necessarily their votes. This would be achieved via a 'Proof of Status' method that uses zero-knowledge proofs. Each voter could assume a alternate wallet and would prove to some verifier that they own some auxiliary wallet that contains at least some amount of tokens from the DAO. In the spirit of zero-knowledge, they would not reveal the identity of their auxiliary account. Then, the voter could submit their vote from their alternate account without revealing their true identity. However, a crippling drawback of this approach is that we cannot assume true decentralization and need some form of central figure in order to facilitate the scheme. Additionally, this method currently has no measures in place to prevent agents from using the same proof of status and voting multiple times.

Our second approach was from a different angle. Instead of hiding identities but not votes, we pivoted towards hiding votes but not necessarily identities. We proposed to use homomorphic public key encryption to encrypt each voter's vote, send it to a smart contract for the DAO, and have the contract handle aggregation, decryption, and release of the final total vote count. However, this also has some drawbacks. Particularly, we rely on a single smart contract's ability to have a private key that will not be compromised. Additionally, one smart contract would have access to every voter's identity and also their full vote, which does not feel particularly robust to attacks.

Finally, we proposed a modification of our second approach to increase the difficulty for an adversary to compromise the scheme. We did this via adding the concept of secret sharing. In essence, we have two smart contracts which work independently and do not communicate. Each voter splits their vote into two separate vectors, encrypts each one using one of the smart contracts' private keys, and then sends the encrypted vectors to the corresponding contract. Each smart contract will aggregate the vote shares they received and decrypt and publicize their final results at the end of the voting period. All members of the DAO can then sum up the two results to get the final vote tally. This makes it so that even if an adversary could recover one of the private keys, they would not be able to gain any information about the voters' votes. Both private keys would be necessary, and the smart contracts can be obfuscated in different ways to greatly increase the computational burden on an adversary.

6.1 Future Work

The critical limitations of our Proof of Status have to be addressed in order to implement a privacy policy that effectively conceals voter identity. Future research would be on alternatives to a central authority that still allows for a form of token verification similar to what was used in our current Proof of Status solution. This research may also offer a solution to the duplicate voting vulnerability, since that problem could be easily solved with a central authority.

A logical extension to our current vote hiding approach is extending the secret sharing from two smart contracts to n smart contracts. The security of the protocol currently is dependent on it being difficult to recover two separate private keys from the two smart contracts. This level of difficulty can be increased if the number of contracts was larger. So, generalizing our protocol to n contracts is logical next step.

While more contracts makes the security stronger, the use of obfuscation to conceal the private keys would still a weakness of our protocol. Future research should seek out alternative methods to obfuscation that are provably hard. Alternatively, an analysis could be done on the strength of security of our protocol with n contracts. If this is sufficiently difficult the use of obfuscation could be justified.

If future research can address the shortcomings of Proof of Status, and the security of Vote Hiding can be strengthened, then combining the two protocol would be valuable. It could facilitate fully anonymous voting, where neither a voter's identity nor their vote could be recovered.

References

[CF85] Josh D. Cohen and Michael J. Fischer. "A robust and verifiable cryptographically secure election scheme". In: *26th Annual Symposium on Foundations of Computer Science (sfcs 1985)*. 1985, pp. 372–382. DOI: `10.1109/SFCS.1985.2`.

[GMR85] S Goldwasser, S Micali, and C Rackoff. "The Knowledge Complexity of Interactive Proof-Systems". In: *Proceedings of the Seventeenth Annual ACM Symposium on Theory of Computing*. STOC '85. Providence, Rhode Island, USA: Association for Computing Machinery, 1985, pp. 291–304. ISBN: 0897911512. DOI: `10.1145/22145.22178`. URL: `https://doi.org/10.1145/22145.22178`.

[Sch91] C. P. Schnorr. "Efficient signature generation by smart cards". In: *Journal of Cryptology* 4.3 (1991), pp. 161–174. DOI: `10.1007/bf00196725`.

[Pai99] Pascal Paillier. "Public-Key Cryptosystems Based on Composite Degree Residuosity Classes". In: vol. 5. May 1999, pp. 223–238. ISBN: 978-3-540-65889-4. DOI: `10.1007/3-540-48910-X_16`.

[DJ00] Ivan Damgård and Mads Jurik. "A Generalisation, a Simpli.cation and Some Applications of Paillier's Probabilistic Public-Key System". In: *Lecture Notes in Computer Science* 7 (Dec. 2000). DOI: `10.7146/brics.v7i45.20212`.

[Zhe+17] Zibin Zheng et al. "An Overview of Blockchain Technology: Architecture, Consensus, and Future Trends". In: *2017 IEEE International Congress on Big Data (BigData Congress)*. 2017, pp. 557–564. DOI: `10.1109/BigDataCongress.2017.85`.

[Sch+21] Moritz Schloegel et al. *Loki: Hardening Code Obfuscation Against Automated Attacks*. 2021. DOI: 10.48550/ARXIV.2106.08913. URL: https://arxiv.org/abs/2106.08913.

[But] Vitalik Buterin. *Privacy on the Blockchain*. URL: https://blog.ethereum.org/2016/01/15/privacy-on-the-blockchain/.

[Coi] Cointelegraph. *What is a decentralized autonomous organization, and how does a DAO work?* URL: https://cointelegraph.com/decentralized-automated-organizations-daos-guide-for-beginners/what-is-decentralized-autonomous-organization-and-how-does-a-dao-work.

[DJ] DJ. *What are Governance Tokens? How Token Owners Shape a DAO's Direction*. URL: https://decrypt.co/resources/what-are-governance-tokens-how-token-owners-shape-dao.

[Enta] William Entriken. *Anatomy of Smart Contracts*. URL: https://ethereum.org/en/developers/docs/smart-contracts/anatomy/.

[Entb] William Entriken. *Introduction To Smart Contracts*. URL: https://ethereum.org/en/developers/docs/smart-contracts/.