S2DV:Scalable and Secure DAO Voting

Sermin KOCAMAN1 and Ali DOGAN2,3

 FAME CRYPT, Ankara, Turkiye sermin.cakin@gmail.com
Istanbul Settlement and Custody Bank (TAKASBANK) Istanbul, Turkiye alidoganmath@gmail.com
Informatics Institute, Istanbul Technical University, Istanbul, Turkiye

Abstract. Decentralized Autonomous Organization operates without a central entity, being owned and governed collectively by its members. In this organization, decisions are carried out automatically through smart contracts for routine tasks, while members vote for unforeseen issues. Scalability in decisionmaking through voting on proposals is essential to accommodate a growing number of members without sacrificing security. This paper addresses this challenge by introducing a scalable and secure DAO voting system that ensures security through Groth16 zk-SNARKs and exponential ElGamal encryption algorithm while achieving scalability by verifiably delegating heavy computations to untrusted entities. While offline computation on the exponential ElGamal homomorphic encryption algorithm is enabled to reduce the computational cost of the blockchain, Groth16 is allowed to maintain robust off-chain calculation without revealing any further details. Specifically, the Groth16 proof guarantees that (i) the encrypted votes accurately reflect the voter's voting power, ensuring no unauthorized weight manipulation; (ii) only valid non-negative vote values are encrypted, preventing unintended or malicious vote tampering; and (iii) the homomorphic summation is performed correctly. The implementation shows that the proofs are verified remarkably fast, making the S2DV protocol highly suitable for scalable DAO voting, while preserving the security of the election.

Keywords: Decentralized Autonomous Organization, Voting, zk-SNARK

1 INTRODUCTION

Blockchain technology has facilitated the establishment of self-organized decentralized application, especially known as the Decentralized Autonomous Organization (DAO), governed by rules encoded as smart contracts on the blockchain [8]. The smart contract in DAOs guarantees automatic rule execution for predictable situations, while the distributed ledger records critical information on the blockchain. Although these structures fulfill requirements in certain circumstances, they are insufficient in case of dynamic and unpredictable circumstances. At that point, human decisions are essential for reaching a collective decision on the rules by voting. Members within the DAOs put forth proposals, engage in discussions, and cast votes regarding the management of the organization. The outcome of the election signifies the decision of the community. In 2016, the first DAO, referred to as *The DAO*, was established in the literature [20]. Subsequently, many DAO platforms emerged, contributing to the advancement of this

framework, including DAOstack ⁴, Mantle (previous BitDAO) ⁵, Mango DAO ⁶, DAO-haus ⁷, Radicle ⁸, and Aragon ⁹. More information on them can be found in [16, 22].

Although several forms of DAO platforms exist, community-driven voting is crucial for establishing collective decision-making for all DAOs. The blockchain in DAOs enables transparent and trustworthy voting processes within organizations [22], While each DAO platform uses different voting models in blockchain, most of the DAO platforms employ a coin-based voting mechanism [2]. In this model, each member has the right to vote according to their vote power determined by their token. A proposal is accepted upon approval by a majority of members, or in the case of an absolute majority, wherein a predefined quorum is met. This model's approach to voting power prioritizes equity over equality in specific contexts. For instance, in decentralized finance (DeFi), individuals who have contributed a greater number of tokens or have been actively involved with the community should be allocated greater voting power, as critical decisions are likely to have a more significant impact on them.

For a robust voting system, important cryptographic primitives need to be used for specific purposes. For the correctness of each operation in voting, Zero-Knowledge Proofs (ZKPs) take the stage for privacy-preserving validation. Nonetheless, their computational costs require effective proofs. For the vote confidentiality, homomorphic encryption algorithm enables computations on encrypted votes. While these cryptographic primitives enhance security, they can introduce performance challenges. When the number of members in DAO's decision-making process is increased, the underlying cryptographic primitives significantly impact the system's performance and efficiency. For this reason, scalable the voting system with desirable security is essential for DAOs. In this study, we design a voting system that balances between desirable characteristics.

Our Contribution. In this study, we developed a scalable and secure voting system for DAOs, named as S2DV. In S2DV, the confidentiality of the votes is ensured through additive exponential ElGamal homomorphic encryption, and implemented with threshold decryption to prevent the disclosure of intermediate results before the election is over, ensuring that voters' decisions are not influenced. Also, encrypted votes are paired with a Groth16-based ZKP. The proof ensures that the encrypted votes accurately represent the voter's token-based power, prevents the encryption of negative values, and validates the correct application of homomorphic encryption and summation, allowing precise aggregation without revealing individual votes. Importantly, we assigned offline calculations on encrypted votes, aggerated before submission, to untrusted off-chain entities in a manner that allows verification with ZKP. This approach enables certain computations to occur locally, while ensuring verification of the operation with the usage of cryptographic protocols. To showcase the efficiency of our design, we create prototype implementations and perform various experiments for different implemen-

⁴ https://daostack.io/

⁵ https://www.mantle.xyz/

⁶ https://dao.mango.markets/dao/MNGO

⁷ https://daohaus.club/

⁸ https://radicle.xyz/

⁹ https://www.aragon.org/

tation options. We also compare our Groth16-based system with system built using Σ -protocols and Bulletproof.

Paper Organization. In this paper, Section 2 gives an overview of voting and the necessary cryptographic primitives utilized in the construction of S2DV. While Section 3 presents formal construction and an efficient instantiation of S2DV, Section 4 gives the security and performance analysis of it. Finally, Section 5 concludes the paper.

2 PRELIMINARY

This section gives the differences between traditional voting and DAO-based voting, followed by an overview of the fundamental cryptographic primitives necessary for S2DV system construction. Specifically, following the discussion of the encryption algorithm, we present the Σ -protocols as a basis for system comparison. Although S2DV system does not inherently rely on Σ -protocols, we introduce our S2DV-adapted Σ -based proofs to demonstrate their applicability within our framework, enabling a comparative evaluation. Finally, we present the Groth16 zk-SNARK protocol, which forms the core of our cryptographic design, ensuring efficient and privacy-preserving verification.

2.1 Traditional Voting Versus DAO Voting

The voting in DAO differs from the traditional voting systems in some aspects. In traditional voting systems, various voting requirements need to be fulfilled [9, 24, 32]. One of the important requirements is the notion of eligibility, which ensures that only authorized voters are allowed to cast a vote [17]. The other requirement is uniqueness, which guarantees that each voter's ballot cannot be counted more than once during the vote-tallying process [9]. These requirements for traditional voting have different criteria when it comes to DAO voting. They differ within the DAO voting framework as follows:

- Eligibility. While eligibility in traditional voting systems is granted with identitybased registration to participate in an election, the authorization required for participation in DAO voting is provided by possessing the necessary token or having a specific membership in DAOs. In DAO voting, the criteria for eligibility can include one-token one vote, one-person one vote, or restrictions to specific members to limit participation [12].
- Uniqueness. Uniqueness in traditional voting systems prevents double-voting in tallying, ensuring equal vote privileges. However, in the context of DAO voting, the privileges associated with voting can vary based on the vote power determined by the token. This means that although members of DAO have an equal right to vote, the impact of each vote may vary, leading to an unequal influence on the outcome of the election.

On the other hand, the decision-making process in DAO voting can employ various decision-making mechanisms beyond a simple majority in traditional voting to reach a consensus. A supermajority requires a minimum percentage of "yes" votes for a proposition to pass, while a quorum necessitates a certain percentage of voter participation

[12]. These varied methodologies enable DAOs to adjust to distinct situations and promote more inclusive and effective governance.

Various Dao platforms offer many different voting models, including Holographic, Conviction, Reputation (Weighted), Quadratic, and Token-based Votings [12, 21]. Their properties and key metrics can be found at [15].

2.2 Exponential ElGamal Homomorphic Encryption Algorithm

The exponential version of the ElGamal encryption algorithm enables additive operations on encrypted data by encoding the message *m* as g^m [13]. Also, its distributed version supports secure collaborative decryption, preventing the disclosure of data through a joint decryption process. Let *p*, *q* be both primes, $g \in \mathbb{G}_q$ be a generator of a cyclic group \mathbb{G} of order *q*. Threshold Exponential ElGamal algorithm works as follows [6]:

- **KGen** $(pp) \rightarrow (sk, pk)$: Taking public parameters $pp = (\mathbb{G}, g, \mathbb{Z}_p)$, key generation algorithm samples secret key as $x \leftarrow \mathbb{Z}_p^*$, then outputs public key as $pk = g^x$. For the threshold version of exponential ElGamal, Pedersen's distributed key generation algorithm can be executed [29].
- **Enc**(*pk*, *m*) \rightarrow (*C*₁, *C*₂): Taking message $m \in \mathbb{Z}_p$, and public key *pk*, encryption algorithm samples random $r \in \mathbb{Z}_p^*$ and outputs ciphertext as (*C*₁, *C*₂) = ($g^r, g^m \cdot pk^r$).
- **Dec**(*pk*, *sk*, (*C*₁, *C*₂)) \rightarrow *m*: On input *pk*, *sk*, and ciphertext (*C*₁, *C*₂), decryption algorithm outputs *m* such that $g^m = C_2/(C_1)^x$.

Addition (Additive homomorphism of exponential ElGamal). Given two ciphertexts encrypting messages $m_1, m_2 \in \mathbb{Z}_q$ under the same public key pk, their componentwise multiplication results in a new ciphertext encrypting the sum of the original messages:

$$Enc(m_1, pk) \cdot Enc(m_2, pk) = (g^{r_1 + r_2}, g^{m_1 + m_2} \cdot pk^{r_1 + r_2})$$
$$= Enc(m_1 + m_2, pk)$$

2.3 Σ -Protocols

This section introduces the Σ -protocols-based design of the S2DV system to enable a systematic comparison with the original S2DV framework, which utilizes the Groth16 zk-SNARK protocol. In our token-based DAO voting, voters need to provide some ZKPs to ensure the correctness of their vote. In detail, they need to prove that the total distributed voting power matches their allocated vote weight while also demonstrating that each encrypted vote is non-negative through a range proof. The first one can be generated by adapting vote sum proof in traditional voting system. Traditional voting with the exponential ElGamal algorithm necessitates the proof of equality of two discrete logarithms as a vote sum proof to ensure that, despite the submission of multiple encrypted votes, the voter did not select more than one candidate [23]. However, for the token-based DAO voting system, the proof of equality of two discrete logarithm

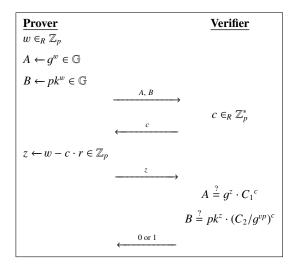


Fig. 1: Proof of equality for the vote power

needs to be adapted to prove the vote power of the member/voter, as voting power is distributed to the candidates according to preference. On the other hand, the non-negativity range proof for DAO voting can be demonstrated utilizing adapted Bulletproofs [7]. It should be noted that since Σ -protocols do not inherently support proving homomorphic summation within the proof itself, the aggregation of encrypted votes in Σ -based design must be performed on-chain by the smart contract.

Proof of equality of two discrete logarithms The execution of classic Σ protocol [30] in parallel proves the equality of two discrete logarithm [10]. In the context of token-based DAO voting that uses exponential ElGamal, this proof can be tailored to the member's vote power vp.

Assume that a member M generates exponential ElGamal-encrypted vote as $V_{c_i} = (C_{1i}, C_{2i}) = (g^{r_i}, g^{p_x} \cdot pk^{r_i})$ for each candidate c_i $(1 \le i \le n)$, where p_x represents the member's vote power allocated to c_i , and the total vote power satisfies $\sum p_x = vp$. Its encrypted vote with aggregated candidates will be $V_c = (\prod_{i=1}^n C_{1i}, \prod_{i=1}^n C_{2i}) = (g^r, g^{vp} \cdot pk^r) = (C_1, C_2)$, where $r = \sum_{i=1}^n r_i$. The member can prove that its ballot with aggregated candidate decrypts to total vote power. The proof of equality for the vote power is depicted in Figure 1. This protocol is designed to establish the following relation:

$$\left\{ \left(vp \in \mathbb{Z}_p, \ g, \ pk, \ C_1, \ C_2 \in \mathbb{G}; \ r \in \mathbb{Z}_p \right) : C_1 = g^r \wedge C_2 = g^{vp} \cdot pk^r \quad \right\}$$
(1)

Proof of non-negativity of an encrypted vote Range proofs achieves to prove that the member's vote power p_x is non-negative. The Bulletproof protocol, designed by Bünz et

al. [7], is commonly used for range proofs and offers a highly efficient solution in terms of both proof size and verification time. Bulletproof is non-interactive and composable inner-product range proof that allow provers to show that multiple values lie within a specific range using a concise, combined proof. The protocol builds on the inner-product arguments (IPA) developed by Bootle et al. [4]. Bulletproof reduced the size of the commitment vector by applying a $\log_2 n$ recursive transformation that compresses an *n*-dimensional vector into a 1-dimensional vector, thereby decreasing the complexity of the original IPA. Furthermore, the Bulletproof protocol does not require a trusted setup phase, which differentiates it from zero-knowledge proof methods like zk-SNARKs and highlights its advantages for decentralization.

Let $v \in \mathbb{Z}_p$, and consider V, an element in a group G, which serves as a Pedersen commitment to the value v using a randomness parameter γ . The goal of the proof system is to demonstrate to the verifier that v falls within the range of integers from 0 to $2^n - 1$. Simply put, the proof system seeks to verify the following relation:

$$\left\{ \left(g, \ h \in \mathbb{G}, \ V, \ n; \ v, \ \gamma \in \mathbb{Z}_p \right) : V = h^{\gamma} g^v \wedge v \in [0, 2^n - 1] \right\}$$
(2)

In the Σ -based design of the S2DV system, the nonnegativity proof requires transformation to align with the specific cryptographic primitives. In the system, a homomorphic ElGamal encryption $(C_1, C_2) = (g^r, g^m \cdot pk^r)$ needs to be transformed to be compatible with Bulletproofs, which require inputs in the form $C = (g^m \cdot h^r)$. To bridge this gap, a ZKP is provided to demonstrate that the message in the commitment used as input to Bulletproof is equal to the message in the ciphertext. Then, the commitment can then be given as input to bulletproof. This commitment proof is depicted in Figure 2 and designed for the following relation:

$$\left\{ \left(g, h, C_2, pk \in \mathbb{G}; m, r, r' \in \mathbb{Z}_p\right) : C_2 = g^m \cdot pk^r \wedge C = g^m \cdot h^{r'} \right\}$$
(3)

2.4 Groth16 zk-SNARK Protocol

A zk-SNARK (Zero-Knowledge Succinct Non-Interactive Argument of Knowledge) is a cryptographic protocol that allows a prover to demonstrate to a verifier that a computation F has been executed correctly. Specifically, the prover shows knowledge of a witness w such that y = F(x, w), where x is publicly known. A key feature of zk-SNARKs is their succinctness: both the size of the proof and the time required for verification are significantly smaller than the size of the computation F and the witness w.

The Groth16 zk-SNARK [19] is particularly notable for its efficiency in verifying rank-1 constraint systems (R1CS), as established in [18]. It stands out due to its minimal proof size and rapid verification, maintaining its status as the benchmark for practical applications. These advantages have made Groth16 widely adopted, emphasizing the importance of advancing its design. Groth16 relies on asymmetric pairings between the groups \mathbb{G}_1 , \mathbb{G}_2 , and \mathbb{G}_T , with the proof consisting of two elements from \mathbb{G}_1 and one element from \mathbb{G}_2 .

Groth's zk-SNARK protocol leverages Quadratic Arithmetic Programs (QAPs)[18], [28] to represent the satisfiability of R1CS instances in a computationally efficient manner using specific low-degree polynomials. The protocol effectively consolidates the *M*

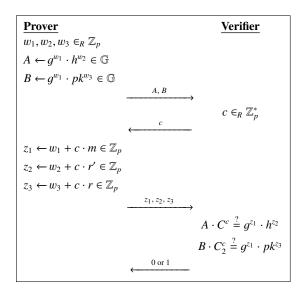


Fig. 2: Proof of commitment

constraints into a single equation involving univariate polynomials of degree O(M). The prover's task is to demonstrate to the verifier that this equation is satisfied. Importantly, it suffices for the verifier to confirm that the equation holds at a randomly chosen point, as distinct low-degree polynomials (relative to the field size) can only coincide at a limited number of points.

3 OUR PROPOSED S2DV PROTOCOL

This section begins with an overview of our DAO voting protocol, followed by a detailed discussion of each stage. The Groth16-based S2DV Framework can be divided into 9 steps, as shown in Figure 3. Apart from the smart contract administrator, who sets up the voting smart contract, there are three main parties involved: voters, the smart contract, and proposal creators. The procedure can be analyzed in three phases: preelection, election, and post-election.

Pre-election starts with a distributed key generation (DKG), initiated by the proposal creators, to establish encryption keys for the election. They then upload this public key along with the Groth16 verification key. Once all voters have registered for the election, the proposal creators trigger the start of the voting process. In the election phase, a voter retrieves the current encrypted votes, homomorphically adds their own vote, and generates a Groth16 proof. They then submit the proof along with the new encrypted votes to the smart contract. The contract verifies the proof and updates the encrypted votes accordingly. To finalize the election, the proposal creators trigger the completion of the election and perform threshold decryption. They submit the decrypted voting

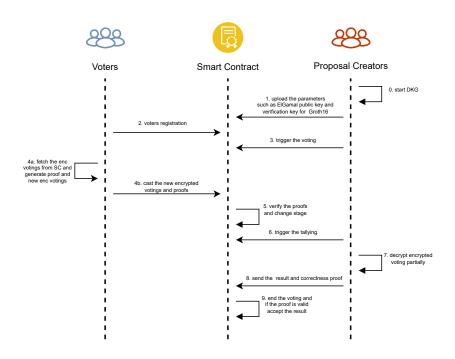


Fig. 3: S2DV Framework

results and corresponding proofs to the smart contract. If the proofs are valid, the smart contract finalizes the election.

3.1 Formal Construction

The formal construction of the employed functionalities is given in Algorithm 1. For the syntax, we employed standard notations. If A denotes an algorithm, then A(x) represents the output of this algorithm when applied to input x. While $x \leftarrow y$ represents the straightforward assignment operation. , A(x) \rightarrow y signifies that y is the output of the algorithm A. Furthermore, Kgen, Enc, Dec, Pr, Vr, \mathcal{HE} , and \mathcal{ZKP} denote key generation, encryption, decryption, proof, verification, homomorphic encryption, and zeroknowledge proof, respectively. Additionally, n_v , n_c , and t represent the number of the voter, the number of the proposal creators, and threshold value of creators, respectively.

EncDKG function of Algorithm 1 is executed by the proposal creators using Pedersen's distributed key generation [29]. After each proposal creator generates its partial secret key sk_i and its corresponding public key pk_i , the election's encryption key pk is generated distributively. This key generation ensures that no single entity possesses complete control over the private/decryption key.

VoteGen function of Algorithm 1 is run by voters, enabling them to cast their encrypted votes. Each voter retrieves an aggregated vote *AggVote* from the smart contract and appends his encrypted vote C_i to it with homomorphic operations. To achieve integrity and prevent malicious acts, the voter produces a zero-knowledge proof (ZKP) π_{Vote_i} , demonstrating that the vote was accurately constructed without revealing its content. Specifically, the ZKP proves that the ciphertext legitimately constructed within the allowed range (e.g., non-negativite), that the vote power assigned to the voter is correctly enforced, and that summation is performed correctly. This process eliminates the need for a trusted tallying authority as vote aggregation are conducted in a distributed manner, while offering transparent verification due to its ZKPs.

VoteVerif function is run by the smart contract to verify the correctness of submitted proofs. Upon successful verification, smart contract accepts the aggregated vote *AggVote* and changes the phase of the election.

ParDecVote function is executed by proposal creators to decrypt the aggregated vote partially. Each proposal creator takes the aggregated vote *AggVote* confirmed by the smart contract, then applies partial decryption with its secret key sk_i . Additionally, they generate the correct decryption proof π_{pd_i} to prove that partial decryption performed using the designated secret key share associated with the public key generated during the Distributed Key Generation. More details of partial decryption for threshold ElGamal system can be found in [11].

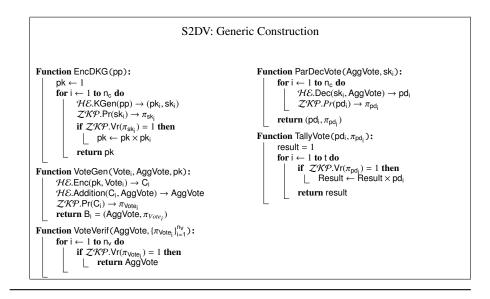
TallyVote function is run by the smart contract to declare the election result. The smart contract first verifies the correctness proof of description, then aggregates the partially decrypted values to announce the result.

3.2 An Efficient Instantiation

Encryption (HE). While most homomorphic encryption algorithms encounter scalability challenges as the number of candidates rises, ElGamal algorithm is effective in these situations [25]. Thus, ElGamal algorithm [13] is selected for encryption due to its efficiency in homomorphic encryption. More specifically, the exponential variant of ElGamal, wherein the message is encoded in the exponent, is employed owing to its additive properties. Following the traditional decryption process in ElGamal, this version necessitates an additional step after decryption. However, as the message is confined to the number of (voters \times vote power), this additional discrete logarithm operation is not difficult to compute. Even efficient solving techniques like Baby-step Giant-step (BSGS) can be used to reduce time complexity [31].

Proofs (\mathcal{ZKP}) In the system, ZKP verifies that the encrypted votes correctly reflect the voter's token-based power, that no negative values are encrypted, and that homomorphic summation are applied correctly, enabling accurate aggregate without disclosing individual votes. Both zk-SNARKs and Σ -protocols can be utilized to verify these properties except homomorphic summation. When using Σ -protocols, homomorphic summation cannot be directly performed by the voter, as Σ -protocols do not support this operation. In the case of Σ -based design, the aggregated. This introduces an additional challenge, as performing aggregation within the smart contract increases computational overhead and may slow down the overall system. Therefore, while Σ -protocols

Algorithm 1: S2DV: Generic Construction



can be used for the system, the homomorphic summation proof must be handled differently, necessitating a change in system design. Given these considerations, Groth16 zk-SNARKs provide a more efficient solution. By directly incorporating homomorphic summation within the proof, zk-SNARKs ensure that the system can handle aggregation off-chain while minimizing transaction costs and maintaining system efficiency. Thus, despite the use of Σ -protocols for comparison, zk-SNARKs should be preferred for large-scale DAO voting applications.

4 ANALYSIS

4.1 Security Analysis

The security of the protocol is guaranteed by employing cryptographic primitives, thereby strengthening voter trust. S2DV satisfies privacy, robustness, fairness, and verifiability, thereby demonstrating that it constitutes a secure and reliable Decentralized Autonomous Organization (DAO) voting system.

Privacy. Privacy entails severing the links between the voter and the vote [17]. Vote privacy of the proposed protocol achieves privacy by employing the exponential El-Gamal algorithm under the decisional Diffie-Hellman (DDH) assumption. The DDH assumption claims that, without knowledge of the secret exponents *a* and *b*, an attacker cannot distinguish between the two tuples (g^a, g^b, g^{ab}) and (g^a, g^b, g^c) , where c is a random value. This ensures that, even if an adversary captures the encrypted vote, they cannot extract any meaningful information about the voter's choice.

Robustness. It refers to the system's resilience despite potential malicious actions from voters or election entities [26]. S2DV achieves this by leveraging ZKPs in each stages. Within the system, each voter retrieves the aggregated vote from the smart contract, appends its vote along with vote power and non-negativeness range proofs, and then submits it back to the smart contract. These proofs guarantee that the supplied votes adhere to valid parameters and preclude any manipulation. As a result, the only remaining step in the process is the decryption of the aggregated vote stored in the smart contract. However, this decryption process is already carried out by the proposal creators, who also publish its ZKP to prove the correctness of the decryption.

Fairness. Fairness refers to the absence of any results being revealed before the election is completed [17]. In S2DV, distributed key generation is utilized to create encyption key. After the election has ended, the proposal creators collaboratively decrypt the aggregated votes with its secret key. Since nobody knows the election's decryption key, this process ensures that the election results is preserved during the election phase.

Verifiability. Election verifiability guarantees that voters can verify the correctness of the election result without relying on some authority [3]. In S2DV, voters can confirm that their votes have been accurately recorded through a smart contract on the blockchain, ensuring transparency and immutability. Additionally, the aggregation of votes can be independently verified by any party using homomorphic encryption, accompanied by the correct ZKPs, which guarantees the integrity of the vote tallying process without revealing private information. Thus, each stage, from vote casting to result counting, is independently verifiable by participants and external observers.

Name	#Candidates	Generation [ms]	Verification [ms]	#Constraints
$Tx(\Sigma)$	3	14.9	12.5	-
Tx (Groth16)	3	114	0.772	26209
$\operatorname{Tx}(\Sigma)$	4	20.1	16.3	-
Tx (Groth16)	4	191	0.794	34945
$\operatorname{Tx}(\Sigma)$	5	24.6	20.4	-
Tx (Groth16)	5	203	0.815	43681
$\operatorname{Tx}(\Sigma)$	6	28.8	23.8	-
Tx (Groth16)	6	209	0.822	52417

4.2 Performance Analysis

Table 1: Transaction Performance Metrics

As previously mentioned, we developed two distinct implementations of S2DV for the system comparison: one utilizing zk-SNARKs and the other based on Σ -protocols. Since zk-SNARK constraints can be easily defined in a higher-level language, the zk-SNARK-based implementation offers superior extendability and ease of implementation compared to the Σ -protocols version, which necessitates manual construction of ZKPs to achieve optimal performance. Therefore, the zk-SNARK-based S2DV implementation is more likely to be adopted in practice and is the primary focus of our evaluation. To compare, we present the system's applicability and performance results for both zk-SNARK-based and Σ protocol-based implementations, which are available in our Github ¹⁰. Our evaluation of the scheme was conducted on a standard user computer equipped with 8 GB of RAM, 6 cores, and a 2.90GHz CPU.

Implementation based on zk-SNARKs. The zk-SNARK implementation of S2DV is built using the gnark library [5], leveraging the BN256 curve [1] and the Groth16 proof system [19].

Implementation based on Σ **-protocols and Bulletproof.** Using the go-ethereum library [14] for Σ -protocols, the P-256 curve [27] was used. The zkrp library developed by ING Bank was used for Bulletproof.

Table 1 presents the proving, verification times and number of candidates for both of our implementations, along with the number of R1CS constraints in the Groth16-based implementation. All results are averaged over 50 runs and rounded to three significant figures. This table demonstrates that, in contrast to the sigma protocol, the verification time is substantially shorter in a Groth16-based proof, even though the prover time is larger. Given that verification is performed on-chain, choosing the Groth16-based system is a reasonable approach to minimize transaction fees.

5 CONCLUSION

In this study, we introduced S2DV, a secure and scalable voting protocol for DAOs. Our protocol uses threshold decryption to avoid premature result disclosure while maintaining vote confidentiality with additive exponential ElGamal homomorphic encryption. In order to ensure voting integrity, encrypted votes are combined with Groth16-based zk-SNARKs, which confirm that votes correctly reflect the voter's token, that only legitimate non-negative values are encrypted, and that homomorphic summation is applied appropriately.

A key innovation of our approach is the delegation of offline computations to untrusted off-chain entities, while ensuring their correctness through cryptographic proofs. By performing homomorphic aggregation off-chain instead of in the smart contract, we reduce the burden on the contract. This enables efficient computation without compromising security. Our prototype implementation and experimental evaluation demonstrate the feasibility of S2DV, showing that while Groth16 zk SNARK incurs a higher proving time compared to sigma protocols, it significantly reduces verification time. Given that verification occurs on-chain, this trade-off is justified as it minimizes transaction fees. Overall, our work highlights the potential of leveraging zk-SNARKs and homomorphic encryption to enhance scalability and security in DAO voting.

¹⁰ https://github.com/midmotor/privacy-preserve-dao-voting

Bibliography

- Barreto, P.S., Naehrig, M.: Pairing-friendly elliptic curves of prime order. In: International workshop on selected areas in cryptography. pp. 319–331. Springer (2005)
- [2] Bellavitis, C., Fisch, C., Momtaz, P.P.: The rise of decentralized autonomous organizations (daos): a first empirical glimpse. Venture Capital 25(2), 187–203 (2023)
- [3] Benaloh, J., Rivest, R., Ryan, P.Y., Stark, P., Teague, V., Vora, P.: End-to-end verifiability. arXiv preprint arXiv:1504.03778 (2015)
- [4] Bootle, J., Cerulli, A., Chaidos, P., Groth, J., Petit, C.: Efficient zero-knowledge arguments for arithmetic circuits in the discrete log setting. In: Annual International Conference on the Theory and Applications of Cryptographic Techniques. pp. 327–357. Springer (2016)
- [5] Botrel, G., Piellard, T., Housni, Y.E., Kubjas, I., Tabaie, A.: Consensys/gnark: v0.11.0 (Sep 2024). https://doi.org/10.5281/zenodo.5819104, https://doi.org/10.5281/zenodo.5819104
- [6] Brandt, F.: Efficient cryptographic protocol design based on distributed el gamal encryption. In: International Conference on Information Security and Cryptology. pp. 32–47. Springer (2005)
- [7] Bünz, B., Bootle, J., Boneh, D., Poelstra, A., Wuille, P., Maxwell, G.: Bulletproofs: Short proofs for confidential transactions and more. In: 2018 IEEE symposium on security and privacy (SP). pp. 315–334. IEEE (2018)
- [8] Cai, W., Wang, Z., Ernst, J.B., Hong, Z., Feng, C., Leung, V.C.: Decentralized applications: The blockchain-empowered software system. IEEE access 6, 53019– 53033 (2018)
- [9] Cetinkaya, O., Cetinkaya, D.: Towards secure e-elections in turkey: requirements and principles. In: The Second International Conference on Availability, Reliability and Security (ARES'07). pp. 903–907. IEEE (2007)
- [10] Chaum, D., Pedersen, T.P.: Wallet databases with observers. In: Annual international cryptology conference. pp. 89–105. Springer (1992)
- [11] Cramer, R., Gennaro, R., Schoenmakers, B.: A secure and optimally efficient multi-authority election scheme. European transactions on Telecommunications 8(5), 481–490 (1997)
- [12] Ding, Q., Liebau, D., Wang, Z., Xu, W.: A survey on decentralized autonomous organizations (daos) and their governance. World Scientific Annual Review of Fintech 1, 2350001 (2023)
- [13] ElGamal, T.: A public key cryptosystem and a signature scheme based on discrete logarithms. IEEE transactions on information theory **31**(4), 469–472 (1985)
- [14] Ethereum Foundation: go-ethereum: Official Go implementation of the Ethereum protocol. https://github.com/ethereum/go-ethereum (2025), accessed: Feb 6, 2025
- [15] Fan, Y., Zhang, L., Wang, R., Imran, M.A.: Insight into voting in daos: conceptual analysis and a proposal for evaluation framework. IEEE Network 38(3), 92–99 (2023)

- [16] Faqir-Rhazoui, Y., Arroyo, J., Hassan, S.: A comparative analysis of the platforms for decentralized autonomous organizations in the ethereum blockchain. Journal of Internet Services and Applications 12, 1–20 (2021)
- [17] Fujioka, A., Okamoto, T., Ohta, K.: A practical secret voting scheme for large scale elections. In: Advances in Cryptology-AUSCRYPT'92: Workshop on the Theory and Application of Cryptographic Techniques Gold Coast, Queensland, Australia, December 13–16, 1992 Proceedings 3. pp. 244–251. Springer (1993)
- [18] Gennaro, R., Gentry, C., Parno, B., Raykova, M.: Quadratic span programs and succinct nizks without pcps. In: Advances in Cryptology–EUROCRYPT 2013: 32nd Annual International Conference on the Theory and Applications of Cryptographic Techniques, Athens, Greece, May 26-30, 2013. Proceedings 32. pp. 626– 645. Springer (2013)
- [19] Groth, J.: On the size of pairing-based non-interactive arguments. In: Advances in Cryptology–EUROCRYPT 2016: 35th Annual International Conference on the Theory and Applications of Cryptographic Techniques, Vienna, Austria, May 8-12, 2016, Proceedings, Part II 35. pp. 305–326. Springer (2016)
- [20] Hassan, S., De Filippi, P.: Decentralized autonomous organization. Internet Policy Review 10(2) (2021)
- [21] Hellström, E.: Fair voting system for permissionless decentralized autonomous organizations (2022)
- [22] Ibrahimy, M.M., Norta, A., Normak, P.: Blockchain-based governance models supporting corruption-transparency: A systematic literature review. Blockchain: Research and Applications 5(2), 100186 (2024)
- [23] Joaquim, R.: How to prove the validity of a complex ballot encryption to the voter and the public. Journal of information security and applications 19(2), 130–142 (2014)
- [24] Langer, L.: Privacy and verifiability in electronic voting. Ph.D. thesis, Technische Universität Darmstadt (2010)
- [25] Li, L.: An electronic voting scheme based on ElGamal homomorphic encryption for privacy protection. In: Journal of Physics: Conference Series. vol. 1544, p. 012036. IOP Publishing (2020)
- [26] Magkos, E., Kotzanikolaou, P., Douligeris, C.: Towards secure online elections: models, primitives and open issues. Electronic Government, an International Journal 4(3), 249–268 (2007)
- [27] National Institute of Standards and Technology: FIPS PUB 186-4: Digital Signature Standard (DSS). Tech. rep., National Institute of Standards and Technology (2013), https://doi.org/10.6028/NIST.FIPS.186-4, accessed: Feb 6, 2025
- [28] Parno, B., Howell, J., Gentry, C., Raykova, M.: Pinocchio: Nearly practical verifiable computation. Communications of the ACM 59(2), 103–112 (2016)
- [29] Pedersen, T.P.: Non-interactive and information-theoretic secure verifiable secret sharing. In: Annual international cryptology conference. pp. 129–140. Springer (1991)
- [30] Schnorr, C.P.: Efficient signature generation by smart cards. Journal of cryptology 4, 161–174 (1991)
- [31] Shanks, D.: Class number, a theory of factorization, and genera. In: Proc. Symp. Math. Soc., 1971. vol. 20, pp. 415–440 (1971)

[32] Wang, K.H.K., Mondal, S.K., Chan, K.C., Xie, X.: A review of contemporary evoting: Requirements, technology, systems and usability. Data Science and Pattern Recognition 1(1), 31 (2017)