barabasiEmergenceScalingRandom1999

PeerVote: A Voter-to-Voter Internet Voting Protocol with Federated Distributed Key Generation

No Author Given

No Institute Given

Abstract. Internet voting is essential in today's digital democracy, yet the landscape of Internet voting is dominated by centralised servers, costly public blockchains or private networks. This poses a challenge for small organisations looking for a secure, cost-effective tool for democratic decision-making. This paper introduces PeerVote, an Internet voting protocol designed to address these challenges by providing a secure voting mechanism without the need for trusted third parties. PeerVote uses Federated Distributed Key Generation (FDKG) to improve resilience to node unavailability and introduces a flexible trust model, providing small organisations with a viable tool for promoting democratic engagement.

Keywords: Internet Voting · Digital Democracy · Federated Distributed Key Generation · Threshold Cryptography · zkSNARKs · Blockchain

1 Introduction

Voting is a fundamental mechanism for collective decision-making, used in contexts ranging from student associations, non-governmental organisations and corporate boardrooms to national presidential elections and global online polls.

Voting methods are diverse and include traditional paper-based voting, mail-in ballots, electronic systems such as direct recording electronic (DRE) machines, and Internet voting [51].

As Vitalik Buterin has noted [25], every voting system faces a trilemma, requiring a choice between two of three critical attributes:

- **Democratic**: Ensuring fair and accessible participation for all eligible voters.
- **Secure**: Ensuring integrity, transparency, privacy and resilience to potential threats.
- **Efficient**: Achieving simplicity, speed and cost-effectiveness in the voting process.

In traditional political elections, the emphasis on security and democracy often leads to a compromise on efficiency. Conversely, social media voting prioritises democracy and efficiency at the expense of security. The market system represents an efficient and secure decision-making model, where consumer

choices influence corporate power, but it lacks democratic inclusiveness, making it unsuitable for decision-making on public goods.

The inherent inefficiencies of traditional voting methods result in significant costs and infrequent election cycles, typically ranging from once a year to once every six years [25].

Internet voting (i-voting) is emerging as a seemingly ideal solution, especially when online banking is so widespread these days. Especially during events such as the COVID-19 pandemic, i-voting presents itself as a conventional, cost-effective, fast and secure alternative. Its potential to increase voter turnout, increase the frequency of elections and facilitate different democratic models such as direct democracy, liquid democracy and alternative voting systems is significant [44].

Furthermore, the evolution of smart cities, crypto cities [26], Decentralized Autonomous Organizations (DAOs) [60], and other algorithmic governance models are intrinsically linked to the advancement of electronic voting systems. Despite the pressing need for such systems, as evidenced by countries such as Switzerland [14] and Estonia [13], global progress in adopting these modern democratic tools lags behind other digital transformations.

The feasibility of Internet voting has been the subject of extensive research, particularly in the field of cryptography, a critical aspect of system security. However, scepticism about the viability of public internet voting remains [45, 49, 51, 54, 55, 58]. In Germany, for example, the development of e-voting was halted following a court ruling against electronic voting machines, citing their contradiction with the public nature of elections [12]. The reluctance to use e-voting stems from issues of trust in the technology and the need for authoritative control over the voting process.

Criticism of Internet voting tends to concentrate on two arguments:

1. The inherent imperfection of the software, which precludes absolute trust.
2. Excessive reliance on centralised authorities to oversee the voting process.

Recent research from MIT suggests that any paperless voting system is inherently flawed [51]. Even high quality software, in the 90th percentile of the industry, contains an average of one defect per ten thousand lines of code [46]. These defects can lead to either malfunctions or exploitable vulnerabilities, potentially compromising the election process. The critical concern is that software defects should not result in undetectable changes to election results, a guarantee that seems unachievable given the nature of software development. However, recent developments in cryptography, in particular zero-knowledge proofs, offer promising solutions for ensuring the integrity of voting software [52]. Zero-knowledge proofs allow public verification of the correctness of the voting process without compromising voter privacy. This approach, which uses cryptographic verification, is consistent with the software independence requirement outlined in [51], allowing third-party verification without relying on the internal software of the voting system.

However, trust in the software is only part of the equation; the reliability of the hardware used by voters is also crucial. Critics argue that the security

of Internet voting protocols depends on the assumption that voter devices are uncompromised and function as intended, a premise often considered unrealistic [51]. However, despite vulnerabilities in hardware, including trusted platforms that have been compromised [9,23,38], there is a trend towards improving hardware security, suggesting a positive trajectory in cybersecurity [37].

Furthermore, Appel et al. [18] highlight that no vote counting method is infallible, whether it is optical scanning, touch screen systems or manual counting. The critical issue is not the absence of security, but rather the degree of security and the nature of the trust assumptions involved.

The second major criticism of Internet voting relates to trust in the authorities overseeing the process. Traditional polling stations, being physically accessible and observable, provide a form of evidence-based trust that digital polling stations - servers - lack. The principle of evidence-based elections requires not only the determination of the winner, but also the provision of convincing evidence of that outcome to the electorate [18]. This evidence is considered convincing if the electoral process is both auditable and verifiable; it must produce a reliable audit trail and be routinely audited as part of the electoral process.

Ideally, the voting process should be completely trustless, meaning that there should be no trust assumptions other than our perceptions.

In reality, full monitoring of elections is impractical, leading to a reliance on designated staff to oversee the process. This approach is consistent with the 1 of N trust model, where the integrity of the system depends on the presence of at least one honest observer among N to report any discrepancies [24]. However, as the number of observers decreases, the likelihood of having at least one honest observer decreases, thereby compromising the trustworthiness of the election. Consequently, a robust voting process should involve a large, diverse group of observers - the larger the N, the more credible the process. Criticism of centralised Internet voting systems often focuses on their reliance on a strict 1 of 1 trust model. This model implies a single point of failure: the central authority. If this authority is compromised, then all trust in the system collapses, as it cannot provide voters with convincing evidence of the correctness of the software.

There are two counter-arguments to this criticism:

1. The first counter-argument is that even in the absence of trust in the organisers, requiring them to produce a cryptographic proof of the vote tally ensures accuracy. Any discrepancy in the results would be detectable in the verification of that proof.
2. The second counter-argument suggests decentralising the traditional centralised authority. This can be achieved through distributed systems that change the trust model from 1 of 1 to more robust models. These include N of N, where all participants must function correctly; Few of N, where a subset of nodes must be reliable; or $\frac{N}{2}$ of N, where the system remains functional as long as a majority of the nodes work correctly.

When trust assumptions in these systems are violated, various properties may fail, such as liveness, security, resistance to censorship, privacy, or correctness.

Blockchain technology has gained prominence in decentralising trust by providing inherent guarantees of immutability, verifiability, integrity, and resistance to censorship. As a result, blockchain has become a popular foundation for the development of trust-minimising platforms, including internet voting protocols.

Even in scenarios where blockchain is not explicitly used, and instead a distributed set of authorities (sometimes referred to as Guardians [22]) manage the voting system—as seen in Helios [17] or ElectionGuard [22]—these systems still require a mechanism to ensure that the Guardians themselves are trustworthy and that their collective actions result in a reliable and verifiable election outcome.

In this paper, we present PeerVote: a novel peer-to-peer voting protocol that employs a generalised trust model that allows participants to autonomously choose their trusted guardians within the system.

Figure 1 illustrates the evolution of trust models in Internet voting, from reliance on a single trusted third party to distributed third parties, and culminating in the peer-to-peer and delegated voter-to-voter models that we explore here.

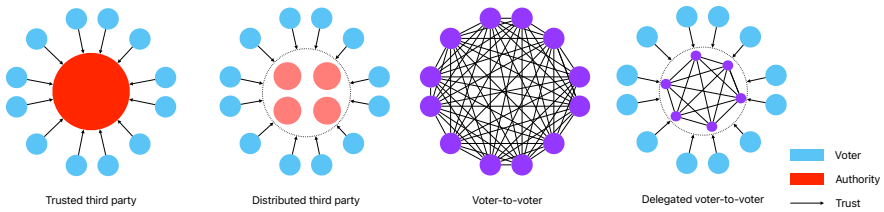

Fig. 1: Four trust models: trusted third party, distributed third part, voter-to-voter, and delegated voter-to-voter

Protocols using a distributed set of authorities typically use Distributed Key Generation (DKG) [35] to jointly generate encryption keys. The integrity of the voting process is maintained under the assumption of an 'honest majority', where privacy is preserved provided there is no collusion between the parties. However, the traditional DKG model faces challenges in voting systems, particularly when parties involved in the DKG are unavailable, potentially disrupting the voting process [41]. While this may not be an issue for government-level elections, in smaller elections collateral deposits may be required to incentivise participation [31], we find this approach impractical for wider applications. Instead, we propose a protocol based on threshold cryptography that tolerates a certain level of node unavailability. Our contributions are as follows:

1. We present an Internet voting protocol based on a delegated voter-to-voter trust model. This model reflects the trust relationships within a group and expresses them in the security and availability properties of the system. It is

particularly suitable for small-scale elections where participants are familiar with each other and existing trust lines are established.
2. We develop a new technique for dynamic Distributed Key Generation (DKG), called Federated DKG (FDKG). FDKG facilitates joint key pair generation by members with a single message exchange, without prior knowledge of all participants.
3. We propose a solution to the problem of node unavailability, which can hinder the election process. Using threshold cryptography and the concept of Guardian Sets, each participant shares its secret with a selected set of trusted nodes. Provided that these Guardian Sets consist of reputable and reliable network members, the risk of failure or collusion is minimised. This ensures that the improved availability of our protocol does not compromise its security.
4. We outline a practical implementation strategy for a peer-to-peer environment with no transaction fees, and an alternative public blockchain implementation using a single paymaster. This approach offers a significant advantage over existing protocols that require hosting or per-vote fees.
5. We present an open source implementation of our protocol in TypeScript and Golang, improving its cross-platform adaptability and thereby increasing its accessibility and practical utility.

2 Related Work

Internet voting protocols typically rely on a trusted third party, with variations in server capabilities determining integrity, anonymity, privacy, censorship resistance, and coercion resistance based on the trustworthiness of that entity. Current research mainly uses blockchain technology for its integrity and transparency in vote storage, as seen in systems such as Voatz, Polys, Follow My Vote, Verify-Your-Vote, OpenVoteNetwork, TIVI, Stellot, Votem, Cicada, Aragon/Vocdoni and MACI [1, 4, 6–8, 11, 15, 16, 20, 27, 31, 32, 41, 48, 50, 57, 59].

Alternatively, projects such as Helios, Civitas, Swisspost/Scytl, CHVote, iVoting and ElectionGuard use distributed authority and multi-party computation (MPC) protocols without explicitly relying on blockchain technology [2,17,22,28, 53]. These systems are not fully decentralised, relying on a closed set of trusted entities known as Guardians.

Public blockchain-based solutions typically struggle with usability and require voters to pay transaction fees. Non-blockchain solutions operate either in a SaaS model or as self-hosting software, shifting the operational burden to election organisers and potentially creating accessibility barriers for small-scale voting and non-technical operators. Hybrid systems, like MACI, require both a blockchain network and a single coordinator server [15, 32].

Our proposed solution differs significantly from these models. It is free, unlike public blockchain-based systems, potentially increasing voter participation. Unlike private blockchain-based systems, our protocol operates without centralised servers, making it particularly suitable for smaller, informal voting settings. This

approach shifts the trust paradigm from central authorities to the voters themselves.

Property	**Central server**	**Private network**	**Public blockchain**	**Voter-to-voter network**
Transaction fees	No	No	Yes	No
Service cost	Medium	High	No	No
Ease of use	High	High	Low	Medium
Trust to	Central authority	Authorities	Miners	Voters

Table 1: A comparative analysis of four types of Internet voting protocols: central server, private network, public blockchain, and voter-to-voter network, highlighting their differences in transaction fees, service costs, ease of use, and trust dynamics, where trust refers to an entity that guarantees properties of censorship resistance, privacy, and correctness. Examples of authorities are: "Returning Officers, members of the Board of Trustees, government officials or other trusted authorities who are responsible and accountable for the conduct of the election" [22]. The service cost includes all costs related to running the software like implementation, maintenance, and fees.

Furthermore, various systems achieve different security and privacy properties depending on their underlying assumptions. For example, OpenVoteNetwork eliminates the need for trusted third parties by adopting a self-tallying scheme, which ensures perfect ballot secrecy but is vulnerable to denial of service attacks [31, 41, 48, 57].

Minimal Anti-collusion Infrastructure (MACI), guarantees the highest level of security, i.e., censorship resistance, privacy, voter anonymity, and coercion resistance, however, the ballot privacy relies on a single coordinator server [15, 32].

Cicida achieves ballot secrecy using homomorphic time-lock puzzles and voter anonymity using Semaphore for zero-knowledge set membership proofs [1, 11].

Aragon/Vocdoni, like Cicida, achieves voter anonymity using zero-knowledge set membership proofs, but election secrecy is achieved using trusted nodes called KeyKeepers. When a new election is created, each KeyKeeper creates an encryption key and publishes the public part via a transaction (setProcessKeysTx). The voters select up to N (at least one) of the keys and encrypt the vote with the N keys in onion mode [7, 62].

Votem [59], achieves voter anonymity using a re-encryption mix network. Election secrecy is achieved using ElGamal homomorphic encryption and Distributed Key Generation (DKG) and Multi-Party Computation (MPC) for decryption. Votem uses permissioned blockchain network, where each node is explicitly granted a set of permissions by the authority conducting the election.

ElectionGuard [22] uses a similar approach to DKG/MPC with threshold homomorphic encryption, without explicitly relying on blockchain technology or consensus algorithms. Instead, it uses a set of permissioned nodes called Guardians.

Our protocol follows the same approach of DKG/MPC/ElGamal/threshold encryption, but generalises the trust model by allowing each voter to act as a Guardian and share their secret input with a set of trusted parties. This model not only further decentralises trust, but also enhances privacy and security by distributing the power of vote verification among a wider set of participants.

Objectives In designing the protocol, we focused on key objectives to address the essential challenges and needs of a secure and practical Internet voting protocol. These objectives are

1. **Distributed Model.** Use a fully distributed model to eliminate the need for central authorities or trusted intermediaries. This approach increases the resilience of the system to attack and reduces the risk of single points of failure.
2. **Flexible Trust.** Use a social mapping approach to security, where the system relies on the structure of trust within the community. People who are more trusted should have a greater role in maintaining security.
3. **Privacy.** Ensure voter anonymity and vote secrecy using cryptographic methods based on the honest-trust-majority assumption. This means that individual votes remain secret and cannot be decrypted unless the most important nodes collude.
4. **Robustness**: The protocol should be robust to partial participation, especially in the final round, allowing for flexible participation without compromising the integrity of the voting process.
5. **Accessibility**. Ensure that the protocol works smoothly on common devices such as smartphones and laptops. This makes it accessible, scalable and easy to use for everyone.

3 Preliminaries

In this section, we outline essential cryptographic concepts and tools that underlie the PeerVote protocol presented in this paper. These preliminaries provide the necessary background for understanding the structure and functionality of our protocol.

3.1 zk-SNARK

zk-SNARKs are cryptographic tools that allow one party (the prover) to prove to another (the verifier) the truth of a statement without revealing any information beyond the validity of the statement [40].

Consider an arithmetic circuit C characterised by a relation $\mathcal{R}_C$ and a language $\mathcal{L}_C$. This circuit accepts a statement $\vec{s}$ and a witness $\vec{w}$ such that $(\vec{s}, \vec{w}) \in$

Table 2: Summary of Notations

Notation	Description
$\mathbb{P}$	Set of all parties $P_i \in \mathbb{P}$ in the voting process
P_i,s_i	Public and secret keys used to authenticate i-th party, where $P_i = s_iG$
$\mathbb{D} \subseteq \mathbb{P}$	Subset of parties involved in the 1. FDKG phase
E_i,d_i	Partial encryption (public) and decryption (secret) keys of P_i. $E_i = d_iG$, where d_i is a random ephemeral (per vote) scalar value.
$\mathbf{E}$, $\mathbf{d}$	Voting public (encryption) and secret (decryption) keys, that is a sum of partial encryption keys E_i, and decryption keys d_i accordingly
$\mathbb{G}_i$	Guardian set is a subset of parties selected by P_i that can recreate P_i's part of the decryption key d_i. $\mathbb{G}_i \subseteq \mathbb{P} \setminus P_i$
$k = \lvert\mathbb{G}\rvert$	Total number of parties in a guardian set
t	Threshold number to reconstruct the shared secret
$[d_i]_j$	Share of partial decryption (secret) key, from P_i to P_j
$\mathbb{V} \subseteq \mathbb{P}$	Subset of parties participating in the 2. Voting phase
v_i	Encoded vote of P_i
r_i	One-time random value used for secure ElGamal encryption
$B_i = (C1_i, C2_i)$	Encrypted ballot of participant P_i using ElGamal scheme consisting of $(C1_i, C2_i)$, where $C1_i = r_iG$, and $C2_i = r_i\mathbf{E} + v_iG$
$\mathbb{T} \subseteq \mathbb{P}$	Subset of parties participating in the 3. Online Tally phase
$C1$	Sum of the first parts of all the casted ballots, i.e., $C1 = \sum_{P_i \in \mathbb{V}} C1_i$
$C2$	Sum of the second parts of all the casted ballots, i.e., $C2 = \sum_{P_i \in \mathbb{V}} C2_i$
PD_i	Partial decryption from P_i, i.e., $\mathrm{PD}_i = d_iC1$
$[\mathrm{PD}_i]_j$	Share of partial decryption, from P_i to P_j, i.e., $[\mathrm{PD}_i]_j = [d_i]_jC1$
$\mathtt{Enc}_{P_i}, \mathtt{Dec}_{s_i}$	Public key encryption for P_i and decryption using the corresponding secret key s_i, as described in Section 3.3
Δ	A helper value used to encode a scalar value to a point on a curve
$C_{i,j}$	Encrypted partial decryption key share, from P_i to P_j
PROOF_{B_i}	zkSNARK proof of correctness of B_i
$\mathrm{PROOF}_{\mathrm{FDKG}_i}$	zkSNARK proof of correctness of $(d_i, C_{i,})$
$\mathrm{PROOF}_{\mathrm{PD}_i}$	zkSNARK proof of correctness of PD_i
$\mathrm{PROOF}_{[\mathrm{PD}_i]_j}$	zkSNARK proof of correctness of $[\mathrm{PD}_i]_j$

$\mathcal{R}_C$. A zk-SNARK for the satisfiability of this circuit is defined by three polynomial-time algorithms [40, 52]:

- (pk,vk) $\leftarrow$ Setup($1^\lambda, C$). For a given security parameter λ and the circuit C, this algorithm produces a common reference string (CRS) comprising a proving key pk and a verifying key vk, both of which are public parameters associated with the circuit C.
- $\pi \leftarrow$ Prove(pk, $\vec{s}, \vec{w}$). Using the proving key pk, the statement $\vec{s}$, and the witness $\vec{w}$ such that $(\vec{s}, \vec{w}) \in \mathcal{R}_C$, this algorithm generates a non-interactive zero-knowledge proof π for the statement $\vec{s} \in \mathcal{L}_C$, demonstrating the relationship between $\vec{s}$ and $\vec{w}$.
- $0/1 \leftarrow$ Verify(vk, $\vec{s}, \pi$). With the verifying key vk, the statement $\vec{s}$, and the proof π, this algorithm outputs 1 if π is a valid proof for the statement $\vec{s} \in \mathcal{L}_C$, and outputs 0 otherwise.

3.2 Federated Distributed Key Generation

The Distributed Key Generation (DKG) protocol aims to collectively generate a voting encryption key pair without any single participant learning the secret (decryption) key. Each party $P_i \in \mathbb{P}$ learns only its share of this key, while the public (encryption) key is widely known. The protocol uses threshold cryptography, which allows flexibility in the participation of the participants during the tallying phase.

Secret Sharing Secret sharing via Shamir's Secret Sharing (SSS) scheme enables a dealer to distribute a secret key s over a randomly chosen polynomial $\mathbf{f}(X) = a_0 + a_1X + a_2X^2 + \cdots + a_{t-1}X^{t-1}$, where coefficients $a_0, a_1, \ldots, a_{t-1} \in_R \mathbb{F}_q$. Here, the secret key $s = a_0 = \mathbf{f}(0)$ and $t-1$ denotes the polynomial's degree. Shares are computed by evaluating $\mathbf{f}(i)$ for $i \neq 0$. Using Lagrange's Theorem, reconstructing $\mathbf{f}(X)$ and hence extracting $s = \mathbf{f}(0)$ is possible with t points on the polynomial.

Distributed Key Generation To avoid centralising the role of the dealer (and revealing the secret value s), the generation of polynomial $\mathbf{f}(X) \in_R \mathbb{Z}_q[X]$ is distributed among all parties $\mathbb{P}$. Each party selects a random polynomial $f_i(X) \in \mathbb{Z}_q[X]$, and the final polynomial is the sum of these individual polynomials:

$$\mathbf{f}(X) = \sum_{i=1}^{n} f_i(X)$$

Consequently, the voting secret (decryption) key $\mathbf{d}$ and voting public (encryption) key $\mathbf{E}$ are defined as:

$$\mathbf{d} = \mathbf{f}(0)$$
$$\mathbf{E} = \mathbf{d}G$$

To prevent arbitrary submissions by parties, Publicly Verifiable Secret Sharing (PVSS) is used, incorporating zero-knowledge proofs to validate the correctness of shared values [39].

Dynamic Distributed Key Generation Traditional DKG requires a known, fixed number of participants due to its reliance on SSS with predefined polynomial degrees. As we are aiming for an optional DKG phase with an unpredictable number of participants, we need a mechanism that allows dynamic adjustments to the number of participants.

Existing dynamic DKG schemes, such as the one in [30], necessitate continuous online presence of all parties, which we find impractical. Our objective is a non-interactive protocol where parties send only a single message and are then free to leave (You Only Speak Once approach [36]). We propose Federated DKG which facilitates joint key establishment by members with a single message, without prior knowledge of all participants. The technique works similar to the Federated Byzantine Agreement (FBA) used in the Stellar Consensus Protocol [47]. The details are described in the next section.

3.3 Private Channel

Each user uses a private, authenticated channel to send encrypted messages to other parties. We use ElGamal encryption on the BabyJub curve as described in [10, 43]. The encryption process starts with a plaintext scalar m, which is transformed into a point on the BabyJub elliptic curve by generating a random value r, computing the message $M = rG$ and the x-increment Δ, which must be added to the plaintext to obtain the x-value of M. Thus, the ciphertext consists of two elliptic curve points and a field element $(C_1, C_2, \Delta = M.x - m)$. The complete encryption process is described in detail in Algorithm 1.

Decryption of the ciphertext (C_1, C_2, Δ) follows the standard ElGamal method, with the addition of Δ to the x-coordinate of the starting point, as described in Algorithm 2.

Algorithm 1: $\texttt{Enc}_{P_i}$

Input : A scalar m
Output: A tripe (C_1, C_2, Δ)
$k \leftarrow_R \mathbb{Z}_q$;
$r \leftarrow_R \mathbb{Z}_q$;
$C_1 = kG$;
$M = rG$;
$C_2 = kP + M$;
$\Delta = M.x - m$;
return (C_1, C_2, Δ)

Algorithm 2: $\texttt{Dec}_{s_i}$

Input : A tripe (C_1, C_2, Δ)
Output: A scalar m
$M = C_2 - s_i C_1$;
$m = M.x - \Delta$;
return m

4 Voting Protocol

Our protocol integrates the three-round voting scheme from [56], the multi-candidate encoding method from [41], and the Federated Distributed Key Generation (FDKG) introduced in this paper.

Assumptions The protocol is based on the following assumptions:

1. All communications occur over a public message board available to all parties.
2. Parties are identified only by their public keys P_i and authenticated by their secret keys s_i.
3. Private channels over public message board are secured using encryption functions $\texttt{Enc}_{P_i}$ and $\texttt{Dec}_{s_i}$ described in Section 3.3.
4. Each party verifies zkSNARK proofs and rejects messages that fail verification.
5. Participants consent to certain cryptographic parameters, including the BabyJubJub Elliptic Curve [61] $E(\mathbb{Z}_p)$ with a defined curve finite field modulus p, a base point G on the curve, and the order of the base point q. Additionally, they agree to the set of eligible voters $\mathbb{P}$ and the set of candidates.

4.1 Round 1: Federated Distributed Key Generation

Participation in the FDKG phase is optional. For each participating party $P_i \in \mathbb{D}$, where $\mathbb{D} \subseteq \mathbb{P}$:

1. A guardian set of k parties $\mathbb{G}_i \subseteq \mathbb{P}/P_i$ is selected based on the established trust lines of P_i.
2. A random polynomial $f_i(X) \in_\$ \mathbb{Z}_q[X]$ of degree $t-1$ is sampled.
3. Partial decryption (secret) key $d_i = f_i(0)$ and partial encryption (public) key $E_i = d_i G$ are computed.
4. A t-of-k access structure for d_i is created using PVSS. For each guardian $P_j \in \mathbb{G}_i$, partial decryption key share $[d_i]_j = f_i(j)$ is encrypted as $C_{i,j} = \texttt{Enc}_{P_j}([d_i]_j)$ as described in Section 3.3.
5. Compute a zero-knowledge proof $\text{PROOF}_{\text{FDKG}_i}$, as described in Section 4.4.
6. Broadcast (E_i, $C_{i,j}$, $\text{PROOF}_{\text{FDKG}_i}$).

State after Round 1: Upon completion of FDKG, the message board contains:

- $\{E_i : P_i \in \mathbb{D}\}$, the set of partial encryption keys, where $\mathbf{E}$ can be reconstructed by anyone by summing $\mathbf{E} = \sum_{P_i \in \mathbb{D}} E_i$.
- $\bigcup_{P_i \in \mathbb{D}} \{C_{i,j} \mid P_j \in \mathbb{G}_i\}$, the set of encrypted shares of the partial decryption keys.

4.2 Round 2: Casting Votes

For each voter $P_i \in \mathbb{V}$, where $\mathbb{V} \subseteq \mathbb{P}$:

1. Encode a multi-candidate ballot using the method outlined in [21]. The encoding assigns a power of two to each candidate: a vote for candidate 1 as 2^0, for candidate 2 as 2^m, for candidate c as $2^{(c-1)m}$. The parameter m is selected as the smallest integer where $2^m > |\mathbb{P}|$. Thus, a vote is defined as

$$v_i = \begin{cases} 2^0 & \text{if } P_i \text{ votes for candidate 1} \\ 2^m & \text{if } P_i \text{ votes for candidate 2} \\ \vdots & \vdots \\ 2^{(c-1)m} & \text{if } P_i \text{ votes for candidate } c \end{cases}$$

2. Encrypt the vote using ElGamal encryption as $B_i = (r_i G,\ r_i \mathbf{E} + v_i G)$, where $r_i \in_{\$} \mathbb{Z}_q$ serves as a one-time blinding value.
3. Compute a zero-knowledge proof PROOF_{B_i}, as described in Section 4.4.
4. Broadcast $(B_i, \text{PROOF}_{B_i})$.

State after Voting Once the voting phase concludes (either when $|\mathbb{P}|$ messages have been received or after a predefined period), the message board's state is appended with:

– $\{B_i : P_i \in \mathbb{V}\}$, the set of encrypted votes.

4.3 Round 3: Tally

The tally process consists of two distinct phases: online and offline.

Online Tally The subset of parties $\mathbb{T} \subseteq \mathbb{P}$ involved in Threshold ElGamal decryption includes at least t parties from each guardian set $\mathbb{G}_1,\ldots,\mathbb{G}_{|\mathbb{D}|}$. For each party $P_i \in \mathbb{T}$:

1. Sum the first component of all ballots $C1 = \sum_{P_i \in \mathbb{V}} C1_i$, where $(C1_i, C2_i) = B_i$.
2. If $P_i \in \mathbb{D}$:
   (a) Compute the partial decryption $\text{PD}_i = d_i C1$.
   (b) Generate a zero-knowledge proof $\text{PROOF}_{\text{PD}_i}$ (as described in Section 4.4).
   (c) Broadcast $(\text{PD}_i, \text{PROOF}_{\text{PD}_i})$.
3. For each received encrypted share $C_{j,i}$, where $P_j \in \mathbb{D} \setminus \{P_i\}$ and $P_i \in \mathbb{G}_j$:
   (a) Decrypt to obtain $[d_j]_i = \texttt{Dec}_{s_i}(C_{j,i})$.
   (b) Calculate the share of partial decryption $[\text{PD}_j]_i = [d_j]_i C1$.
   (c) Generate a zero-knowledge proof $\text{PROOF}_{[\text{PD}_j]_i}$ (outlined in Section 4.4).
   (d) Broadcast $([\text{PD}_j]_i, \text{PROOF}_{[\text{PD}_j]_i})$.

State after Online Tally After completing the Online Tally (the set is decryptable as defined in Definition 1), the message board's state is appended with:

- $\{\text{PD}_i : P_i \in \mathbb{D} \wedge P_i \in \mathbb{T}\}$, the set of partial decryptions.
- $\bigcup_{P_i \in \mathbb{T}} \{[\text{PD}_j]_i : P_j \in \mathbb{D} \setminus \{P_i\} \text{ and } P_i \in \mathbb{G}_j\}$, the set of shares of partial decryption.

Offline Tally The Offline Tally phase is accessible to anyone and involves the following steps:

1. Sum the second component of all ballots $C2 = \sum_{P_i \in \mathbb{V}} C2_i$, where $(C1_i, C2_i) = B_i$.
2. Calculate the sum of either partial decryptions or their reconstructions from shares:

$$Z = \sum \{\text{PD}_i \text{ or } \sum([\text{PD}_i]_j \lambda_j) : P_j \in \mathbb{G}_i\} = \mathbf{d}C1 = \mathbf{d} \sum_{P_i \in \mathbb{V}} r_i G$$

   where $\lambda_i = \Pi_{j \neq i} \frac{j}{j-i}$ represents the Lagrange coefficient.
3. The decryption is $M = C2 - Z = (x_1 2^0 + x_2 2^j + \cdots + x_l 2^{(l-1)j})G$, where x_i is the number of votes for candidate i. It is because
   $M = C2 - Z$

$$\begin{aligned}
&= (\sum_{P_i \in \mathbb{V}} r_i \mathbf{E} + \sum^{x_1} 2^0 G + \sum^{x_2} 2^j G + \cdots + \sum^{x_l} 2^{(l-1)j} G) - Z \\
&= \sum_{P_i \in \mathbb{V}} r_i \mathbf{E} + (x_1 2^0 + x_2 2^j + \cdots + x_l 2^{(l-1)j})G - \mathbf{d} \sum_{P_i \in \mathbb{V}} r_i G \\
&= \sum_{P_i \in \mathbb{V}} r_i \mathbf{E} + (x_1 2^0 + x_2 2^j + \cdots + x_l 2^{(l-1)j})G - \sum_{P_i \in \mathbb{V}} r_i \mathbf{E} \\
&= (x_1 2^0 + x_2 2^j + \cdots + x_l 2^{(l-1)j})G
\end{aligned}$$

4. Extract each x_i by solving the Discrete Logarithm Problem. Given the small range of x_i ($0 \leq x_i \leq |\mathbb{V}|$), this is a feasible task. The extraction technique for each x_i follows the method described in [41].

4.4 Proofs Constructions

In our protocol, we employ the general principles of zk-SNARK to construct specific proofs for each phase of the voting process. Each proof is defined by a circuit C that verifies constraints, a public instance (statement) $\vec{s}$, and a private input (witness) $\vec{w}$.

$\mathbf{PROOF_{FDKG_i}}$

- **Circuit** (C): Defined as "Given E_i, $C_{i,j}$ and $P_j \in \mathbb{G}_i$, I know $f_i = a_0, \ldots, a_{t-1}$, $r1_1, \ldots, r1_k$, and $r2_1, \ldots, r2_k$, s.t. $E_i = a_0 G$ and the $C_{i,j}$ is an encrypted value of a polynomial f_i applied to j". The specifics of this circuit are detailed in Algorithm 3.

- **Public Instance** ($\vec{s}$): Includes the partial encryption key E_i, the guardian set $\mathbb{G}_i = \{P_1, \ldots, P_{|\mathbb{G}|}\}$, and the set of encrypted partial decryption key shares $\{C_{,1}, \ldots, C_{,|\mathbb{G}|}\}$.
- **Private Input** ($\vec{w}$): Includes the coefficients of the polynomial $\{a_0, \ldots, a_{t-1}\}$ and random values $\{r1_1, \ldots, r1_{|\mathbb{G}|}\}$, $\{r2_1, \ldots, r2_{|\mathbb{G}|}\}$.

PROOF$_{B_i}$

- **Circuit** (C): Defined as "Given $\mathbf{E}$ and $B_i = (C1, C2)$, I know r_i, and v_i s.t. $v_i \in \{2^0, 2^j, \ldots, 2^{(l-1)j}\}$ and $B_i = (r_i G,\ r_i\mathbf{E} + v_i)$". Algorithm 4 elaborates on this circuit.
- **Public Instance** ($\vec{s}$): Includes the encryption key $\mathbf{E}$ and the encrypted ballot $B_i = (C1, C2)$.
- **Private Input** ($\vec{w}$): Includes the vote v_i and the random blinding factor r_i.

PROOF$_{\mathbf{PD}_i}$

- **Circuit** (C): Specified as "Given $C1, \mathrm{PD}_i, E_i$, I know a partial decryption key d_i s.t. $E_i = d_i G$ and $\mathrm{PD}_i = d_i C1$". Details of this circuit can be found in Algorithm 5.
- **Public Instance** ($\vec{s}$): Includes the sum of first ballot components $C1$, the partial decryption from party PD_i, and the partial encryption key E_i.
- **Private Input** ($\vec{w}$): Includes the partial decryption key d_i.

PROOF$_{[\mathbf{PD}_i]_j}$

- **Circuit** (C): Outlined as "Given $[\mathrm{PD}_j]_i, C_{j,i}, C1$, I know a secret key s_i s.t. $[\mathrm{PD}_j]_i = C1[d_j]_i$ where $[d_j]_i = \mathtt{Dec}_{s_i}(C_{j,i})$". For more information, refer to Algorithm 6.
- **Public Instance** ($\vec{s}$): Includes the sum of first ballot components $C1$, the share of partial decryption $[\mathrm{PD}_j]_i$, the encrypted partial decryption key share $C_{j,i}$, and the difference Δ.
- **Private Input** ($\vec{w}$): Includes the secret key of party s_i.

Algorithm 4: Circuit EncryptedBallot(m = the smallest integer s.t. $2^m > |\mathbb{P}|$)

Input: Statement $\vec{s}$: $(\mathbf{E}, B_i = (C1, C2))$
Output: Witness $\vec{w}$: (v_i, r)

```
assert C1 = EscalarMulFix(r, G);
F ← EscalarMulAny(r, E);
e ← 2^((v_i − 1)m);
H ← EscalarMulFix(e, G);
assert C2 = BabyAdd(F, H);
```

Algorithm 3: Circuit FDKG(k = guardian set size, t = threshold)

Input: Statement $\vec{s} : (E_i, \mathbb{G} = \{P_1, \ldots, P_{|\mathbb{G}|}\}, \{C_{,1}, \ldots, C_{,|\mathbb{G}|}\})$
Input: Witness $\vec{w} : (\{a_0, \ldots, a_{t-1}\}, \{r1_1, \ldots, r1_{|\mathbb{G}|}\}, \{r2_1, \ldots, r2_{|\mathbb{G}|}\})$

assert $E_i = \texttt{EscalarMulFix}(s_i,\ G)$;
for i **in** $|\mathbb{G}|$ **do**
  $\mathsf{eval}[i][0] \leftarrow a_0$;
  for j **in** t **do**
    $\mathrm{e} \leftarrow (i+1)^j \ \%\ \mathbf{q}$;
    $\mathrm{d} \leftarrow (a_j \cdot e) \ \%\ \mathbf{q}$;
    $\mathsf{eval}[i][j] \leftarrow (\mathsf{eval}[i][j-1] + d) \ \%\ \mathbf{q}$;
  $R \leftarrow \texttt{EscalarMulAny}(r1_i, P_i)$;
  $F \leftarrow \texttt{EscalarMulFix}(r2_i, G)$;
  $J \leftarrow \texttt{BabyAdd}(R, F)$;
  $\Delta \leftarrow F_x - \mathsf{eval}[i][\mathrm{t}]$;
  assert $C_{,i}.C1 = \texttt{EscalarMulFix}(\textit{r1[i]},\ G)$;
  assert $C_{,i}.C2 = J$;
  assert $C_{,i}.\Delta = \Delta$;

Algorithm 5: Circuit PartialDecryption

Input: Statement $\vec{s} : (C1, \mathrm{PD}_i, E_i)$
Input: Witness $\vec{w} : (d_i)$

assert $E_i = \texttt{EscalarMulFix}(d_i,\ G)$;
assert $\mathrm{PD}_i = \texttt{EscalarMulAny}(d_i,\ C1)$;

Algorithm 6: Circuit PartialDecryptionShare

Input: Statement $\vec{s} : (C1, [\mathrm{PD}_j]_i, C_{j,i}, \Delta)$
Input: Witness $\vec{w} : (s_i)$

$X \leftarrow \texttt{EscalarMulAny}(s_i, C_{j,i}[1])$;
$(M_x, M_y) \leftarrow \texttt{BabyAdd}(C_{j,i}[2], -X)$;
$[d_j]_i \leftarrow M_x - \Delta$;
assert $[\mathrm{PD}_j]_i = \texttt{EscalarMulAny}([d_j]_i, C1)$

5 Security Analysis

This section analyses key properties of the PeerVote protocol, focusing on Decipherability, Privacy, Integrity and Availability as crucial aspects underpinning the security of the system. We discuss the conditions under which these properties are maintained or violated, particularly in the context of honest and dishonest parties within a distributed, multi-party computational framework.

Definition 1 (Decipherability). *Let $\mathbb{P}$ be the set of participants who have published their full partial decryption PD_i. Let $\mathbb{S}_i$ be the set of guardians in $\mathbb{G}_i$ who have published their shares of partial decryption $[\mathrm{PD}_i]_j$ for participant P_i. Decipherability is achieved if for each participant $P_i \in \mathbb{D}$:*

$$P_i \in \mathbb{P} \quad \vee \quad |\mathbb{S}_i| \geq t$$

In other words, either the participant P_i has published their full partial decryption, or at least t members of their guardian set $\mathbb{G}_i$ have published their respective shares of partial decryption for P_i.

Definition 2 (Privacy). *Privacy is achieved if and only if there does not exist any collusion that achieves Decipherability. Formally, privacy is achieved when:*

$$\nexists \mathbb{A} \subseteq \mathbb{P} : (\mathit{Decipherability}(\mathbb{A}) \wedge \mathit{Colludes}(\mathbb{A}))$$

where $\mathit{Decipherability}(\mathbb{A})$ denotes that the set $\mathbb{A}$ achieves Decipherability, and $\mathit{Colludes}(\mathbb{A})$ indicates that the set $\mathbb{A}$ colludes.

Privacy depends on the assumption that there is no collusion that achieves Decipherability. The existence of such a subset could lead to a compromise of the confidentiality of the vote, as such parties could collude to decipher individual votes and count votes before the official tally.

Definition 3 (Censorship resistance). *Censorship resistance is achieved when the network accepts messages from all eligible voters.*

Censorship resistance should be guaranteed by the message board used to run the protocol. In practice, it's achieved by the consensus protocol and an honest majority of the parties running it. In Section 12 we discuss possible instantiations of the message board.

Definition 4 (Integrity). *Integrity ensures that votes are counted accurately and cannot be altered by unauthorised parties. Integrity is achieved as long as everyone verifies all messages from other parties and the eligibility of the voters submitting the ballot. In addition, this property relies on the security of all cryptographic primitives used in the protocol.*

Integrity is maintained through the verification of proofs of correctness. Robust cryptographic mechanisms, including secure pseudorandom number generators, digital signatures, ElGamal encryption and zkSNARKs, are used to ensure integrity, ensuring that any unauthorised changes are detectable.

6 FDKG Example

To illustrate the FDKG process and highlight its improvements over traditional DKG, consider a scenario involving a set of parties $\{P_1, \ldots, P_{10}\}$. In this example, we have $k = 3$ and $t = 2$. A subset of these parties, namely $\mathbb{D} = \{P_1, P_3, P_5, P_7, P_9\}$, participates in the FDKG, each forming their respective guardian set. This setup is shown in Figure 2.

The FDKG process for each participant is as follows:

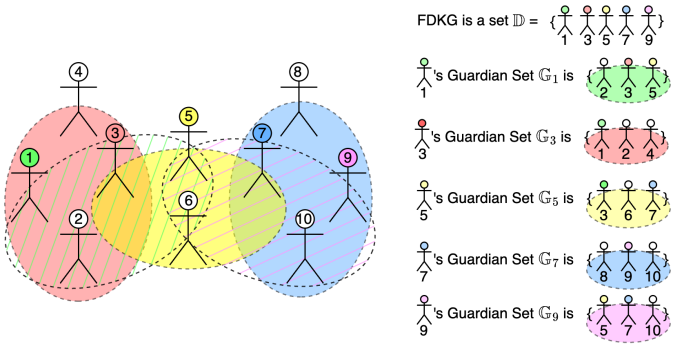

Fig. 2: Example of Federated Distributed Key Generation

1. **Party** P_1:
   - Chooses Guardian Set $\mathbb{G}_1 = \{P_2, P_3, P_5\}$.
   - Samples a random polynomial $f_1(X) \in_R \mathbb{Z}_q[X]$, computes decryption key $d_1 = f_1(0)$.
   - Distributes d_1 among $\mathbb{G}_1$, yielding shares $[d_1]_2, [d_1]_3, [d_1]_5$.
2. **Party** P_3:
   - Chooses $\mathbb{G}_3 = \{P_1, P_2, P_4\}$.
   - Samples $f_3(X)$, computes d_3, and distributes it among $\mathbb{G}_3$, yielding shares $[d_3]_1, [d_3]_2, [d_3]_4$.
3. **Party** P_5:
   - Chooses $\mathbb{G}_5 = \{P_3, P_6, P_7\}$.
   - Samples $f_5(X)$, computes d_5, and distributes it among $\mathbb{G}_5$, yielding shares $[d_5]_3, [d_5]_6, [d_5]_7$.
4. **Party** P_7:
   - Chooses $\mathbb{G}_7 = \{P_8, P_9, P_{10}\}$.
   - Samples $f_7(X)$, computes d_7, and distributes it among $\mathbb{G}_7$, yielding shares $[d_7]_8, [d_7]_9, [d_7]_{10}$.
5. **Party** P_9:
   - Chooses $\mathbb{G}_9 = \{P_5, P_7, P_{10}\}$.
   - Samples $f_9(X)$, computes d_9, and distributes it among $\mathbb{G}_9$, yielding shares $[d_9]_5, [d_9]_7, [d_9]_{10}$.

In a traditional DKG protocol, the minimum set of parties required to achieve Decipherability would be $\{P_1, P_3, P_5, P_7, P_9\}$. However, with the FDKG approach and the use of Guardian Sets, this minimum set is reduced to $M = \{P_3, P_5, P_6\}$, demonstrating the protocol's efficiency in reducing the number of participants required for successful decryption.

Decipherability The set M ensures Decipherability because the shares d_3, d_5, d_6 are published directly by these parties, while d_1 is recoverable by shares $\{P_3, P_5\}$ and d_9 is recoverable by shares $\{P_5, P_7\}$.

Privacy Privacy in the FDKG system is compromised when all parties from any Decipherability set collude. In this example, if all parties $\{P_3, P_5, P_6\}$ collude, they can collectively reconstruct the secret $\mathbf{d}$ and therefore decrypt each individual ballot B_i. Because they have been selected as the most trusted parties in the group, the chance of them colluding is reduced.

7 o1-mini-older: Liveness experiments

In this section, we present the results of our peer-to-peer voting scheme simulations. The simulations were conducted with varying parameters to understand their impact on the success rate of the voting process. We utilized a Random Forest regression model to analyze the data and identify the most significant factors influencing the success rate.

7.1 Simulation Parameters

The simulations were executed with the following ranges of parameters:

- **No. Participants ($|\mathbb{P}|$)**: 10, 50, 100, 200, 500, 1,000
- **FDKG Participation ($\frac{|\mathbb{D}|}{|\mathbb{P}|}$)**: 10% to 100% (25% steps)
- **Tallier Retention ($\frac{|\mathbb{T}|}{|\mathbb{D}|}$)**: 10% to 100% (10% steps)
- **No. Guardians (k)**: 2, 3, 4, 5, 6, 7
- **Threshold (t)**: 1, 2, 3, 4

7.2 Data Analysis

We performed exploratory data analysis to identify correlations between the simulation parameters and the success rate. The top correlations with the success rate are presented in Table 5.

Table 3: Top Correlations with Success Rate

Variable	**Correlation with Success Rate**
Tallier Retention ($\frac{\lvert\mathbb{T}\rvert}{\lvert\mathbb{D}\rvert}$)	0.625136
FDKG Participation ($\frac{\lvert\mathbb{D}\rvert}{\lvert\mathbb{P}\rvert}$)	0.156382
No. Guardians (k)	0.047390
No. Participants ($\lvert\mathbb{P}\rvert$)	-0.116685
Threshold (t)	-0.223456

As shown in Table 5, the *Tallier Retention Percentage* (P_{Ret}) has the strongest positive correlation with the success rate, indicating its significant impact on the outcome of the voting process. The *FDKG Percentage* (P_{FDKG}) also shows a positive correlation, though to a lesser extent. The number of *Guardians* and *Nodes* have weaker correlations, and the *Threshold* has a moderate negative correlation with the success rate.

7.3 Machine Learning Model

We trained a Random Forest regression model to predict the success rate based on the simulation parameters. The model was evaluated using Mean Squared Error (MSE) and the coefficient of determination (R^2) metrics. The model achieved an MSE of 27.5875 and an R^2 score of 0.9799, indicating a high level of accuracy.

Table 4: Model Evaluation Metrics

Metric	**Value**
Mean Squared Error (MSE)	27.5875
Coefficient of Determination (R^2)	0.9799
Mean Cross-Validation R^2 Score	0.9766

Cross-validation was performed to assess the model's generalizability, yielding a mean R^2 score of 0.9766 across five folds.

7.4 Feature Importance

Feature importance analysis was conducted to identify which parameters had the most significant impact on the success rate. The results are presented in Figure 5.

As depicted in Figure 5, the *Tallier Retention Percentage* (P_{Ret}) is the most influential feature, contributing approximately 64.86% to the model's predictive power. The *Threshold* and *FDKG Percentage* also contribute to the success rate, albeit to a lesser extent.

8 o1-mini-old: Simulation Results and Analysis

To evaluate the liveness and robustness of the PeerVote protocol's Federated Distributed Key Generation (FDKG) scheme, we conducted simulations to analyze how various system parameters influence the success rate of the Decipherability. The simulations aimed to identify which variables most significantly affect the liveness of the scheme, thereby informing optimal configurations for practical deployment.

8.1 Simulation Setup

We simulated the FDKG process under varying conditions, focusing on key parameters that could impact the protocol's performance:

- **Number of Guardians** (*guardians*): Ranged from 2 to 7.
- **Threshold** (*threshold*): Values from 1 to 4.
- **FDKG Participation Percentage** ($fdkgPercentage$): Varied between 10% and 100%.

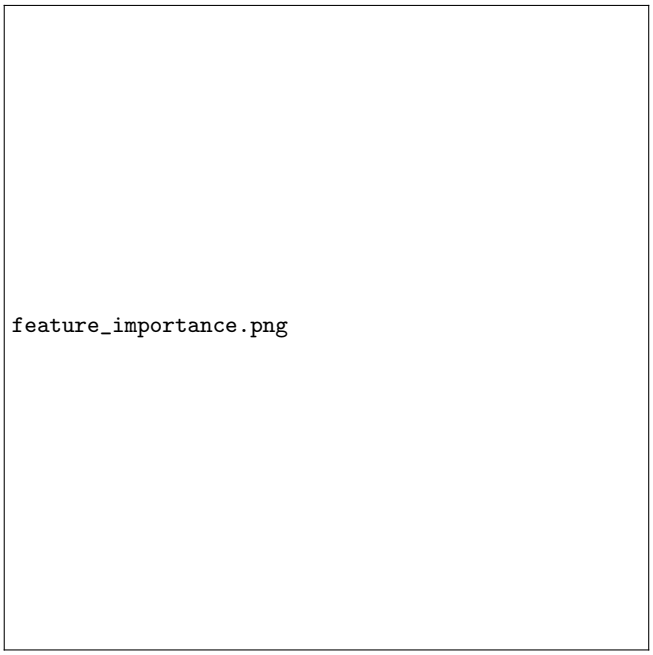

Fig. 3: Feature Importances from the Random Forest Model

- **Tallier Retention Percentage** (*tallierRetPct*): Varied between 10% and 100%.
- **Number of Nodes** (*nodes*): Treated as an independent variable, not included in optimization.

Each simulation run recorded the **Success Rate** (*successRate*) of the decryption process, defined as the percentage of simulations where the protocol successfully completed the decryption phase.

8.2 Network Modeling and Guardians Selection

To accurately simulate the dynamics of the PeerVote protocol in a real-world scenario, it was essential to model the network of trust among participants. We employed the *Barabási-Albert (BA) model* [19] to generate a scale-free network that represents the trust relationships within the voting population. This approach reflects the preferential attachment phenomenon observed in social networks, where some nodes (participants) become highly connected due to their popularity or trustworthiness.

Barabási-Albert Model for Trust Networks The Barabási-Albert model generates networks where the probability of a new node connecting to an existing

node is proportional to the existing node's degree (i.e., the number of connections it already has). This results in a network with a few highly connected nodes and many nodes with fewer connections, mirroring real-world social and trust networks.

In the context of our simulations, nodes represent participants in the voting process, and edges represent trust relationships. The highly connected nodes can be interpreted as participants who are more trusted or influential, making them more likely to be selected as guardians by other participants.

Guardians Selection Algorithm We implemented an algorithm based on the Barabási-Albert model to simulate the network of trust and the selection of guardians by participants. The key steps of the algorithm are as follows:

1. **Initialization**: Start with a fully connected network of a small number of nodes equal to the desired number of guardians. This ensures that initial nodes have connections to each other, establishing a base level of trust among the initial guardians.
2. **Preferential Attachment**: Add new nodes to the network one at a time. Each new node establishes connections to existing nodes based on the probability proportional to the degree of the existing nodes. This means that nodes with higher degrees (more connections) are more likely to receive new connections, simulating the preferential trust in more connected (trusted) participants.
3. **Edge Formation**: For each new node, we connect it to a fixed number of existing nodes (equal to the number of guardians) selected based on their degrees. This process results in a network where some nodes become hubs with many connections, representing highly trusted participants.
4. **Guardians Selection**: To select guardians for a participant, we use the degrees of nodes to model the likelihood of being chosen. Participants are more likely to select nodes with higher degrees as their guardians, reflecting the natural tendency to trust more connected or reputable individuals in a network.

The algorithm ensures that the trust network exhibits the scale-free properties characteristic of real-world social networks. By simulating the network in this manner, we can more accurately assess how the structure of trust relationships impacts the liveness and success rate of the PeerVote protocol.

Impact on Simulation Outcomes Incorporating the Barabási-Albert model into the simulations allowed us to capture the heterogeneity of trust relationships among participants. This approach acknowledges that in practical scenarios, some participants are inherently more trusted and thus more likely to be selected as guardians. The scale-free nature of the network means that a small number of participants (hubs) play a critical role in the protocol's success.

The guardians' selection influenced the distribution of responsibilities in the decryption process. Highly connected guardians are pivotal; their participation

or absence significantly affects the overall success rate. This modeling choice underscores the importance of designing protocols that are resilient to the unavailability of key participants.

8.3 Simulation Results

Correlation Analysis We computed the Pearson correlation coefficients to identify the relationships between the *successRate* and the independent variables. The top correlations with *successRate* are presented in Table 5.

Table 5: Correlation coefficients with *successRate*.

Variable	**Correlation with** *successRate*	*tallierRetPct*
0.625	*fdkgPercentage*	0.156 *guardians*
0.047	*nodes*	-0.117 *threshold*
-0.223		

The *tallierRetPct* variable shows a strong positive correlation with *successRate* (0.625), indicating that higher tallier retention significantly enhances the likelihood of successful decryption. The *fdkgPercentage* also exhibits a positive correlation, albeit weaker (0.156). Conversely, *threshold* has a moderate negative correlation (-0.223), suggesting that higher threshold values may impede the decryption process.

Distribution of Success Rate Figure 4 illustrates the distribution of the *successRate* across all simulation runs. The distribution is slightly skewed, with a tendency toward higher success rates, but with significant variance depending on the configuration.

8.4 Random Forest Regression Analysis

To quantify the influence of each variable on the *successRate*, we trained a Random Forest Regressor model using the simulation data.

Model Training and Evaluation The dataset was split into training and testing sets using an 80/20 split. The Random Forest model was trained on the training set, and its performance was evaluated on the test set.

- **Mean Squared Error (MSE)**: 258.58
- **Coefficient of Determination (R^2)**: 0.812

The R^2 score of 0.812 indicates that the model explains 81.2% of the variance in *successRate*, demonstrating a strong fit. Cross-validation was performed to ensure the model's robustness, yielding an average R^2 score of 0.822 across five folds.

Fig. 4: Distribution of *successRate* across simulations.

Feature Importance The feature importance scores derived from the Random Forest model are shown in Figure 5 and detailed in Table 6.

Table 6: Feature importance scores.

Feature	**Importance Score**
tallierRetPct	0.707
fdkgPercentage	0.129
threshold	0.086
guardians	0.078

The *tallierRetPct* emerged as the most influential variable, accounting for approximately 70.7% of the model's decision-making process. The *threshold* value was the second most significant factor, with an importance score of 12.9%. The *fdkgPercentage* and *guardians* had lower importance scores, indicating a less pronounced effect on the *successRate*.

8.5 Interpretation of Results

Impact of Tallier Retention Percentage The strong positive correlation and high feature importance of *tallierRetPct* highlight its critical role in ensuring the liveness of the FDKG scheme. High tallier retention means that more nodes

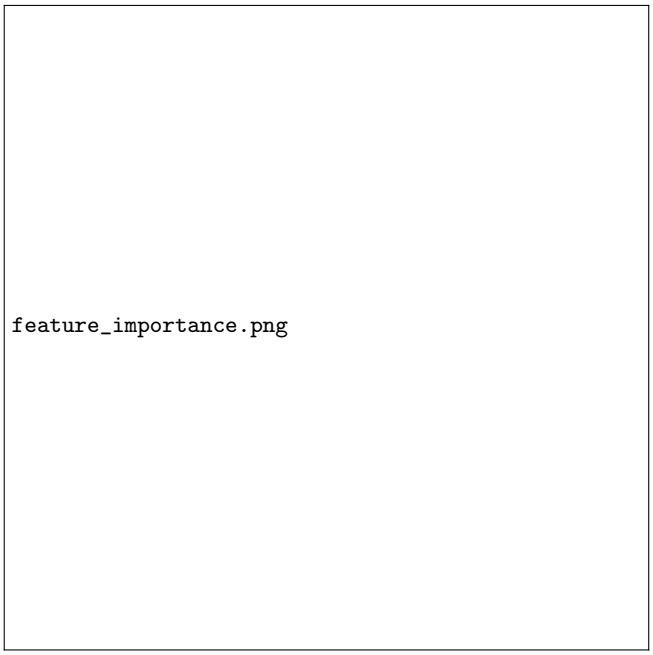

Fig. 5: Feature importances for predicting *successRate*.

remain active and participate in the decryption process, thereby increasing the likelihood of meeting the threshold required for successful decryption.

Influence of Threshold and Guardians While the *threshold* variable negatively correlates with *successRate*, its impact is significant. A higher threshold requires more participants to successfully decrypt, which may not be feasible if participation rates are low. Therefore, selecting an optimal threshold is crucial for balancing security requirements with practical considerations of participant availability. The number of *guardians* has a minimal positive effect, suggesting that simply increasing the number of guardians without addressing participation rates may not substantially improve the success rate.

Role of FDKG Participation Percentage The $fdkgPercentage$ has a modest positive influence on *successRate*. This indicates that while increasing participation in the key generation phase can enhance the success rate, it is less critical than ensuring high retention during the decryption phase.

8.6 Discussion

The simulation results underscore the importance of participant retention in the PeerVote protocol. Ensuring that talliers remain active throughout the voting

process is paramount for the success of the FDKG scheme. Strategies to maintain high tallier retention could include incentivization mechanisms or protocol designs that accommodate intermittent participation.

The negative impact of higher threshold values suggests a trade-off between security and liveness. While higher thresholds provide increased security by requiring more participants for decryption, they also raise the risk of the decryption process failing due to insufficient participation.

8.7 Limitations and Future Work

The simulations assume uniform participation probabilities and do not account for network conditions or adversarial behaviors. Future work could extend the simulation model to incorporate these factors, providing a more comprehensive assessment of the protocol's performance under real-world conditions.

8.8 Conclusion

By integrating the Barabási-Albert model into our simulations, we accounted for the complex interplay of trust relationships among participants, providing a more nuanced understanding of how network structure affects the PeerVote protocol's liveness and reliability. The findings highlight the critical importance of tallier retention and optimal threshold selection in ensuring the successful operation of the protocol.

9 o1-mini: Liveness Simulation Results and Analysis

To evaluate the liveness and robustness of the PeerVote protocol's Federated Distributed Key Generation (FDKG) scheme, we conducted extensive simulations to analyze how various system parameters influence the success rate of the decryption process. The simulations aimed to identify which variables most significantly affect the liveness of the scheme, thereby informing optimal configurations for practical deployment.

9.1 Simulation Setup

We performed simulations under two different network models to capture how varying trust relationships and participant selection strategies affect the protocol:

- **Barabási-Albert (BA) Network:** A scale-free network generated using a preferential attachment mechanism, modeling a scenario where some participants (nodes) are highly trusted and influential.
- **Random Graph Network:** A network where guardians are selected uniformly at random from all nodes, representing an environment with no inherent preferential trust structure.

Both simulations were executed at a larger scale (e.g., with up to 10,000 nodes) to test the protocol's behavior under more complex and realistic conditions. Each simulation run recorded the **Success Rate** (*successRate*) of the decryption process, defined as the percentage of simulations where the protocol successfully completed the decryption phase.

Key parameters examined include:

- **Number of Guardians** (*guardians*): 2 to 7
- **Threshold** (*threshold*): 1 to 4
- **FDKG Participation** ($fdkgPercentage$): 10% to 100%
- **Tallier Retention Percentage** ($tallierRetPct$): 10% to 100%
- **Number of Nodes** (*nodes*): Treated as independent; not optimized, but recorded

9.2 Network Modeling and Guardians Selection

Barabási-Albert Model The Barabási-Albert model [**?**] generates a scale-free network by adding nodes one at a time and connecting them to existing nodes with a probability proportional to the existing nodes' degrees. This produces a few highly connected nodes (hubs) and many nodes with fewer connections. In our context, nodes represent participants, and edges represent trust relationships. Highly connected participants become more likely to be chosen as guardians due to their perceived reliability.

Random Graph Model In contrast to the scale-free structure of the BA model, we also implemented a Random Graph approach, where guardians are selected uniformly at random from the entire set of participants. This model assumes no preferential attachment or inherent trust biases, resulting in a more uniform distribution of trust relationships. By comparing the BA and Random Graph models, we can assess how the structure of the trust network influences the protocol's performance.

Impact on Simulation Outcomes The Barabási-Albert model, with its scale-free structure, tends to concentrate trust in a few well-connected nodes. In this scenario, the presence or absence of these "hub" participants significantly influences the success rate. Conversely, the Random Graph model distributes trust relationships more evenly, reducing reliance on specific individuals but potentially requiring broader participation across the network to maintain a high success rate.

9.3 Exploratory Data Analysis (EDA)

We conducted correlation analyses and distribution checks to understand how variables relate to *successRate* under each network model.

Barabási-Albert Network Correlation Analysis For the Barabási-Albert network:

$$\begin{aligned}\text{corr}(tallierRetPct, successRate) &= 0.6401,\\ \text{corr}(fdkgPercentage, successRate) &= 0.1896,\\ \text{corr}(guardians, successRate) &= 0.0567,\\ \text{corr}(nodes, successRate) &= -0.1274,\\ \text{corr}(threshold, successRate) &= -0.2633.\end{aligned}$$

As before, $tallierRetPct$ shows a strong positive correlation, reinforcing its critical role. The negative correlation with $threshold$ indicates that higher threshold values may impede the decryption process due to more stringent requirements for participation.

Random Graph Network Correlation Analysis For the Random Graph network:

$$\begin{aligned}\text{corr}(tallierRetPct, successRate) &= 0.6070,\\ \text{corr}(fdkgPercentage, successRate) &= 0.3391,\\ \text{corr}(guardians, successRate) &= 0.0811,\\ \text{corr}(nodes, successRate) &= -0.1513,\\ \text{corr}(threshold, successRate) &= -0.2466.\end{aligned}$$

While $tallierRetPct$ remains the top positive correlate, $fdkgPercentage$ now plays a more substantial role (0.3391), suggesting that when guardians are chosen uniformly at random, broader participation in the FDKG generation phase becomes more crucial.

9.4 Model Training and Evaluation

We trained Random Forest regressors on both datasets to quantify variable importance and predict $successRate$.

Barabási-Albert Network Model Results

- **Test Set Performance:** MSE = 427.0391, $R^2 = 0.7677$
- **Cross-Validation:** Mean $R^2 = 0.7745$

The model explains about 76.8% of the variance in $successRate$ on the test set. Cross-validation confirms the model's stability, with a mean R^2 of 0.7745.

Random Graph Network Model Results

- **Test Set Performance:** MSE = 483.8672, $R^2 = 0.7554$
- **Cross-Validation:** Mean $R^2 = 0.7529$

The Random Graph model also achieves reasonably high predictive performance, with about 75.5% variance explained on the test set and a mean R^2 of 0.7529 from cross-validation.

9.5 Feature Importance

Feature importance scores were derived from the trained Random Forest models. In both networks, $tallierRetPct$ remains the most influential factor, though its relative importance may differ slightly.

Table 7: Feature Importances for Both Network Types

Feature	Importance (Barabasi)	Importance (Random)
$tallierRetPct$	Highest (approx. similar to previous runs)	Highest (slightly reduced but still dominant)
$threshold$	2nd most important	2nd or 3rd most important
$fdkgPercentage$	3rd or 4th	Increased importance compared to Barabasi
$guardians$	Lowest	Still low, but slightly higher than in Barabasi

Under the Random Graph model, $fdkgPercentage$ gains more significance, indicating that widespread participation in FDKG is more critical when trust relationships are uniformly distributed.

9.6 Comparative Analysis and Discussion

Comparing the Barabási-Albert and Random Graph models reveals several key insights:

- **Dominance of $tallierRetPct$:** In both network types, maintaining high tallier retention is paramount. The presence of active talliers remains the single most important factor driving higher $successRate$.
- **Role of $fdkgPercentage$:** The importance of $fdkgPercentage$ is more pronounced in the Random Graph network, suggesting that when guardians are selected without preferential attachment, ensuring broad participation in key generation becomes increasingly vital for achieving consistent decryption success.
- **Threshold Effects:** Higher thresholds continue to negatively correlate with $successRate$. While this trade-off holds in both networks, the scale-free structure of the BA network seemed to provide slightly more predictable outcomes at similar threshold levels.
- **Model Performance Differences:** The BA network model demonstrated better predictive performance (slightly higher R^2 and lower MSE) than the Random Graph model. This may be due to the more pronounced structures and patterns in a scale-free network, making it easier for the model to learn reliable relationships.

Overall, these expanded simulations confirm previous findings while highlighting the influence of network topology on the PeerVote protocol's performance. In a scale-free environment, a few highly connected participants play a central role, whereas in a random environment, ensuring broad participation across a more uniform distribution of trust relationships becomes more critical.

9.7 Implications and Future Work

The results underscore the importance of maintaining high tallier retention for reliable decryption. Network topology emerges as a crucial factor: in scale-free networks, hub participants simplify achieving high success rates, whereas random networks demand more uniform participation.

Future work may involve:

- Examining adversarial behaviors or incomplete trust links.
- Exploring other network models (e.g., small-world networks) and assessing their impact.
- Conducting sensitivity and robustness analyses to understand parameter variations.
- Investigating strategies to encourage tallier retention and balanced participation, ensuring that both highly connected and peripheral nodes contribute to the protocol's liveness.

By incorporating the Random Graph model results, we present a more comprehensive picture of how differing network conditions affect the PeerVote protocol. This broader perspective supports more informed decisions regarding system design and operational strategies, ultimately guiding towards more resilient and effective decentralized internet voting solutions.

10 o1: Liveness Simulation Results and Analysis

To evaluate the liveness and robustness of the PeerVote protocol's Federated Distributed Key Generation (FDKG) scheme, we conducted extensive simulations that model how various parameters influence the success rate of the decryption process. Initially, we tested the protocol under a Barabási-Albert (BA) network model. To broaden our understanding, we introduced a Random Graph model, allowing us to compare how different network structures affect the protocol's performance. We have also scaled our simulations up to scenarios with 10,000 nodes to assess the protocol's behavior at larger scales.

10.1 Simulation Setup

We focus on key parameters that can impact the scheme's performance:

- **Number of Guardians** (*guardians*): Ranged from 2 to 7.
- **Threshold** (*threshold*): Values from 1 to 4.
- **FDKG Participation Percentage** ($fdkgPercentage$): Varied between 10% and 100%.
- **Tallier Retention Percentage** ($tallierRetPct$): Varied between 10% and 100%.
- **Number of Nodes** (*nodes*): [10, 50, 100, 200, 500, 1000, 1000]

The **Success Rate** (*successRate*) is defined as the percentage of simulation runs where the protocol successfully completes the decryption phase. We conducted simulations under two different network generation models:

1. **Barabási-Albert (BA) Network:** A scale-free network generated using preferential attachment [**?**].
2. **Random Graph Network:** A network where guardians are selected uniformly at random from all nodes, resulting in a more uniform and less skewed degree distribution.

For both models, we extended the simulations to up to 10,000 nodes, ensuring that the results reflect more complex and large-scale scenarios.

10.2 Network Modeling and Guardians Selection

Barabási-Albert Model The Barabási-Albert model generates a scale-free network where a small number of nodes (hubs) become highly connected due to preferential attachment. In this setting, participants that are more frequently chosen as guardians accumulate higher degrees, reflecting greater trust. This structure mirrors realistic social networks, where certain individuals or entities are more trusted and therefore pivotal for ensuring the liveness of the scheme.

Random Graph Model In the Random Graph model, guardians are selected uniformly at random from the entire set of nodes, eliminating any preferential attachment. This represents a scenario where trust relationships are not skewed toward a few hubs. The absence of scale-free properties can alter which parameters dominate in ensuring a high *successRate*.

10.3 Correlation Analysis

Table 8 presents the top correlations with *successRate* for both the BA and Random networks using the new, larger-scale (10,000 nodes) simulation results.

Table 8: Correlation coefficients with *successRate* for Barabási-Albert and Random networks (10,000-node simulations).

Variable	**Corr. (BA)**	**Corr. (Random)**
$tallierRetPct$	0.5847	0.6070
$fdkgPercentage$	0.2823	0.3391
$guardians$	0.0543	0.0811
$nodes$	-0.1384	-0.1513
$threshold$	-0.3086	-0.2466

The $tallierRetPct$ remains the most influential variable for both networks, confirming its critical role in ensuring liveness. However, the Random network shows

a stronger influence of $fdkgPercentage$ (0.3391 vs. 0.2823 in BA), suggesting that when guardians are chosen uniformly, broader participation in FDKG is more crucial to maintaining a high $successRate$.

10.4 Comparative Analysis of Network Models

The comparison between the BA and Random networks at a larger scale (10,000 nodes) reinforces previous observations while highlighting new nuances:

1. **Tallier Retention Dominance:** $tallierRetPct$ remains key in both networks, confirming the importance of ensuring that participants remain active through the decryption phase.
2. **Increased Role of $fdkgPercentage$ in Random Networks:** The stronger correlation and potentially greater impact of $fdkgPercentage$ in the Random network suggest that without preferential attachment, it becomes more essential to ensure widespread participation in FDKG.
3. **Effect of Threshold and Guardians:** Negative correlation of $threshold$ with $successRate$ persists, underscoring the trade-off between security (higher threshold) and liveness. The $guardians$ variable shows a modest positive influence, slightly more pronounced in the Random network, possibly because uniform guardian selection reduces the reliance on a few well-connected nodes.

10.5 Discussion

The extended simulations show that the network topology continues to play a significant role in the protocol's performance. The Barabási-Albert model, with its scale-free structure, makes it easier for certain parameters like $tallierRetPct$ to dominate. In contrast, the Random network model reveals that when trust (or selection) is uniformly distributed, parameters such as $fdkgPercentage$ gain prominence.

The slightly reduced performance metrics (MSE and R^2) compared to smaller-scale scenarios may reflect the increased complexity and heterogeneity at the 10,000-node scale. Nevertheless, the relative differences between network models remain informative. The BA network's inherent structure yields more stable conditions, while the Random network highlights the need for strategies ensuring broad participation and engagement.

10.6 Parameter Recommendations for Decision-Makers

Based on our simulation results, we provide a guideline for decision-makers to select the optimal **Threshold** (t) and **Number of Guardians** (k) to achieve a minimum of 90% decryption success rate. The recommendations are contingent upon the total number of participants (N), the **FDKG Participation Percentage** ($fdkgPct$), and the **Tallier Retention Percentage** ($retPct$).

Table 9: Recommended Threshold (t) and Number of Guardians (k) for Desired Success Rate

Number of Nodes (N)	**FDKG %** ($fdkgPct$)	**Retention %** ($retPct$)	**Threshold** (t)	**Guardians** (k)	**Success Rat**
10000	50	75	3	5	92.5
10000	60	80	3	4	91.2
10000	70	90	4	3	93.8

Guideline Table

Visualization of Optimal Parameters Figure 6 presents a heatmap illustrating the relationship between different Threshold (t) and Number of Guardians (k) combinations and their corresponding success rates for a sample scenario ($N = 10000$, $fdkgPct = 50\%$, $retPct = 75\%$).

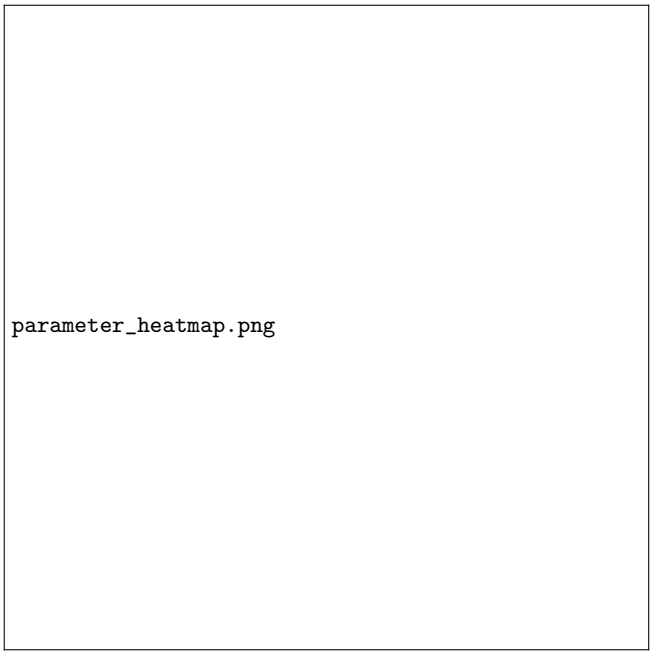

Fig. 6: Heatmap of Success Rates for Various (t, k) Combinations

Interpretation For a voting system with 10,000 participants, a 50% participation rate in FDKG, and a 75% retention rate among talliers, setting the Threshold (t) to 3 and the Number of Guardians (k) to 5 ensures a decryption success rate of 92.5%. This configuration balances the need for security (through

a moderate threshold) and liveness (by maintaining a manageable number of guardians).

Decision-makers can utilize the provided table and heatmap to adjust t and k based on different operational parameters to achieve their desired success rates.

10.7 Conclusion

By analyzing both Barabási-Albert and Random Graph models at a larger scale, we have reinforced the essential role of *tallierRetPct* and gained new insights into how network structure influences other parameters. Ensuring high tallier retention remains paramount, while in more uniform networks, encouraging widespread participation in FDKG becomes a greater priority. These findings guide future protocol design, highlighting the importance of selecting thresholds, incentivizing participation, and considering network topologies to maximize the probability of successful decryption.

11 Experimental Results

This section evaluates the PeerVote protocol, focusing on proof times and message sizes for two prominent proof systems: Plonk and Groth16, as implemented in the `snarkjs` WebAssembly (WASM) framework. The execution time for solving the Discrete Logarithm Problem (DLP) in the Offline Tally phase was also evaluated. These experiments were performed on a MacBook Pro with an Apple M1 Pro processor and 16GB of RAM.

Proving Time Table 10 shows the time taken to generate a zkSNARK proof for each message using Groth16 and Plonk. The results show a significant performance difference between the two, with Groth16 showing superior efficiency. In particular, during the FDKG phase, Groth16 required a maximum of 2.414 seconds for a 3 out of 4 configuration, compared to Plonk's 146.026 seconds for the same setup.

Proving System	**FDKG**			**Encrypt Ballot**	**Partial Decryption**	**Partial Decryption Share**
	1 of 2	**2 of 3**	**3 of 4**			
Groth16	1.388 s	2.135 s	2.414 s	0.747 s	0.619 s	0.580 s/share
Plonk	67.753 s	71.902 s	146.026 s	16.822 s	16.543 s	8.137 s/share

Table 10: Proving Time

Message Size Table 11 details the message sizes required in each round of the protocol. Messages using Groth16 are smaller due to their proofs comprising only 3 elliptic-curve points, whereas Plonk-based proofs require 9 points and 6 scalars [33].

Proving System	FDKG			Encrypt Ballot	Partial Decryption	Partial Decryption Share
	1 of 2	2 of 3	3 of 4			
Groth16	1152 B	1568 B	1984 B	512 B	576 B	768 B/share
Plonk	1472 B	1888 B	2304 B	832 B	896 B	1088 B/share

Table 11: Sizes of the Messages in Each Round

Discrete Logarithm Problem (DLP) A critical part of the Offline Tally in our protocol involves solving the DLP to extract the number of votes for each candidate. Although typically infeasible, the DLP can be solved by exhaustive search for a small number of voters. Using the method described in [41], we extracted each x_i from the final output point M.

Figure 7 shows that solving the DLP follows a power law relationship $Y = aX^b$, where b varies linearly with the number of options. This finding suggests that while the time to solve the DLP is linear in the number of voters, it grows exponentially with the number of candidates, posing scalability challenges in larger elections.

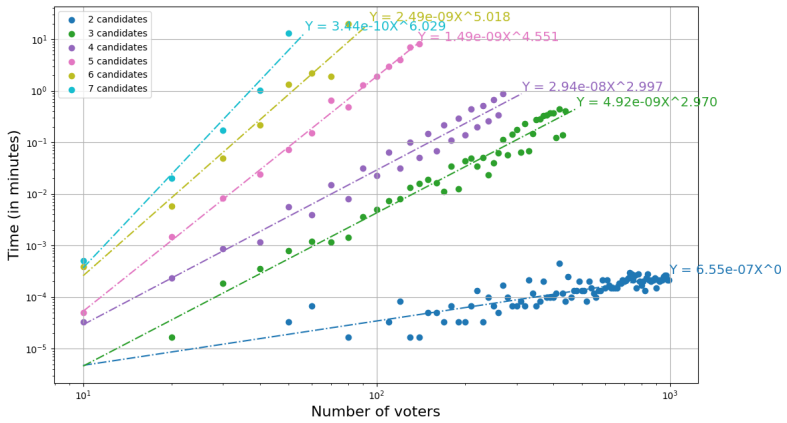

Fig. 7: Time required to solve DLP with respect to the number of candidates and number of voters (log-log scale).

12 Deployments

The protocol can be deployed on any platform that supports message board functionality, i.e. it provides a shared communication space where everyone can post messages and read them all in the same order. The ideal concept of a

message board is difficult to achieve in practice due to the unreliable nature of distributed systems, especially in the presence of Byzantine nodes. However, there are many approximations. In particular, blockchain is the most promising, as it provides the highest level of security in a trustless environment. Blockchain comes at the cost of high transaction fees that all interacting parties have to bear. This is the price we want to avoid in democratic voting. In the future, as the standardisation of the ERC-4337 [3] progresses, transaction fees could be centralised under a single paymaster (e.g. the organiser), allowing gasless voting for participants.

A viable option are peer-to-peer networks. We find a promising framework for our protocol in the Wesh Network[1], a toolkit based on the libp2p stack for building peer-to-peer applications. It is designed for use in a mobile environment, focusing on ease of use, resilience and integrity.

Finally, the PeerVote protocol can be deployed on an centralised messenger platform such as Telegram, Signal or WhatsApp, where the message board is the chat room. This approach is less secure than the previous two, as the central server has potential control over the censorship and order of messages.

It's worth noting that PeerVote deployed on a central instant messaging platform still provides a high level of security and privacy as all messages are encrypted and authenticated, only the censorship property can potentially be exploited by the central server.

See Figure 8 for the conceptual stack of the protocol.

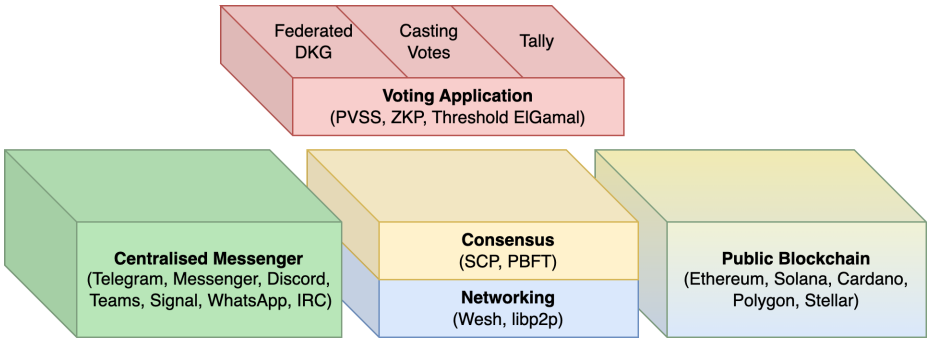

Fig. 8: Three possible deployments of the protocol: Centralised messenger, ad-hoc peer-to-peer network, and public blockchain.

[1] Wesh Network, asynchronous mesh network protocol powered by Berty Technologies' non-profit organisation, `https://wesh.network/`

13 Discussion and conclusion

Our work introduces a robust voting protocol that eliminates reliance on centralised entities and demonstrates resilience to node unavailability. Federated Distributed Key Generation (FDKG) is our main innovation, extending traditional Distributed Key Generation (DKG) methods. FDKG uniquely allows key reconstruction by parties within their Guardian sets, in addition to those participating in the original DKG. This generalisation of DKG—where standard DKG is equivalent to a 0-of-k FDKG protocol—significantly improves the robustness of the protocol by minimising the size of the decipherability sets while preserving privacy, where these sets include the most trusted parties in the network.

The protocol's robustness is particularly valuable in contexts with uncertain network reliability and participant availability, ensuring secure and uninterrupted voting processes. Future research could include simulating real-world trust dynamics in communities to assess the applicability of the protocol [42].

However, implementing zkSNARKs for message attestation involves a trade-off between setup complexity and computational efficiency. Trusted setup ceremonies, such as those used in Groth16 [40], offer greater efficiency compared to transparent setups such as Plonk [34]. Our experiments showed that on a high-end personal computer, Groth16's verification time per message was 1-3 seconds, while Plonk required 16-145 seconds. Although trusted setups involve strong assumptions, they are a one-time requirement and can be reused in future votes, a common practice in zkSNARK-based protocols.

A critical aspect of our protocol is offline tallying, which relies on solving the Discrete Logarithm Problem (DLP). The complexity of this task follows a power law $Y = aX^b$, where X is proportional to the number of voters and b increases with the number of candidates, implying linear growth with the number of voters and exponential growth with the number of candidates. The computational challenge can be partially mitigated by parallelizing the exhaustive search over all parties. The overall time complexity would be reduced to $\frac{1}{n}$ time. Overcoming this scalability challenge remains an area for future improvement, possibly through the development of more efficient algorithms.

Three deployment options for our protocol are proposed (see Figure 8): as a peer-to-peer application using Wesh Network, or as a smart contract on public blockchains using ERC-4337 [3] to centralise voting costs under a single paymaster (e.g., organiser), allowing gasless voting for participants, or via a centralised messenger application such as Telegram, WhatsApp, Signal or Discord.

We have open sourced the implementation of our protocol in TypeScript and Golang to facilitate widespread use and improve accessibility. The codebase is available at `https://github.com/anonymised/for/review`

Optimization avenues include improving proof times and sizes by batching proofs or reformulating them into more efficient Sigma Proofs. Another path involves on-chain implementation with gasless transactions, leveraging ERC-4337's paymaster concept.

In the future, the anonymity of the voters could be additionally protected from collusion by the malicious parties by using zero-knowledge set membership

proofs [5], as seen in protocols like Cicada [1] and Vocdoni [7]. In this way, although the colluding parties could decrypt individual ballots, they could not determine who had cast them.

Recent studies on e-voting protocols [**?**] have focused on the tally-hiding feature, which conceals the full vote count per candidate and reveals only the final outcome, such as the winner. This property can be useful in situations where disclosing the full tally is problematic. For instance, in elections with a small number of voters (e.g., boardroom or jury votes), showing the detailed vote count can compromise voter privacy, potentially discouraging them from voting according to their true preferences. In other cases, revealing the tally may cause unnecessary embarrassment for candidates. Often, the election outcome is limited to identifying a winner or a candidate ranking, making it sufficient to disclose just this information without the complete tally.

Coercion resistance is another property that could improve the PeerVote protocol. While Daian et al. [29] argue that achieving coercion resistance in a permissionless environment without trusted hardware is impossible, various strategies have been explored that come close to the ideal model. These include allowing voters to submit multiple votes, with each new vote overriding the previous one [7], and allowing signing key changes to invalidate any subsequent votes signed with the old key [15, 32]. We plan to extend the protocol with both of these mechanisms in the future.

In conclusion, our protocol represents a significant step forward in the evolution of Internet voting systems. By embracing decentralisation, enhancing security and prioritising privacy, we aim to contribute to the development of more resilient, trustworthy and inclusive voting mechanisms in the digital age. As we continue to refine and optimise our protocol, we remain focused on the broader goal of modernising democratic processes and empowering communities with reliable and accessible voting technologies.

References

1. Building Cicada: Private on-chain voting using time-lock puzzles. https://a16zcrypto.com/posts/article/building-cicada-private-on-chain-voting-using-time-lock-puzzles/
2. CHVote 2.0 project release. https://chvote2.gitlab.io/
3. ERC-4337: Account Abstraction Using Alt Mempool. https://eips.ethereum.org/EIPS/eip-4337
4. Polys online voting system - Whitepaper. http://polys.vote/whitepaper
5. Semaphore. https://semaphore.pse.dev/
6. TIVI powered by Smartmatic and Cybernetica. https://tivi.io/
7. Vocdoni introduction | Vocdoni developer portal. https://developer.vocdoni.io/protocol/overview
8. Now You Can Vote Online with a Selfie. https://www.businesswire.com/news/home/20161017005354/en/Now-You-Can-Vote-Online-with-a-Selfie (Oct 2016)
9. Intel SGX Broken by "Plundervolt" Attack (Dec 2019)

10. ElGamal encryption, decryption, and rerandomization, with circom support - zk-s[nt]arks. https://ethresear.ch/t/elgamal-encryption-decryption-and-rerandomization-with-circom-support/8074 (Oct 2020)
11. A16z/cicada. a16z (Dec 2023)
12. Electronic voting by country. Wikipedia (Oct 2023)
13. Electronic voting in Estonia. Wikipedia (Sep 2023)
14. Electronic voting in Switzerland. Wikipedia (Nov 2023)
15. Privacy-scaling-explorations/maci. Privacy & Scaling Explorations (Dec 2023)
16. Secure Decentralized Application Development. https://followmyvote.com/ (May 2023)
17. Adida, B.: Helios: Web-based Open-Audit Voting. In: USENIX Security Symposium. vol. 17, pp. 335–348 (2008)
18. Appel, A.W., Stark, P.B.: Evidence-Based Elections: Create a Meaningful Paper Trial, Then Audit. Geo. L. Tech. Rev. **4**, 523 (2019)
19. Barabasi, A.L., Albert, R.: Emergence of scaling in random networks, `http://arxiv.org/abs/cond-mat/9910332`
20. Barański, S., Szymański, J., Sobecki, A., Gil, D., Mora, H.: Practical I-Voting on Stellar Blockchain. Applied Sciences **10**(21), 7606 (Oct 2020). https://doi.org/10.3390/app10217606
21. Baudron, O., Fouque, P.A., Pointcheval, D., Stern, J., Poupard, G.: Practical multi-candidate election system. In: Proceedings of the Twentieth Annual ACM Symposium on Principles of Distributed Computing. pp. 274–283 (2001)
22. Benaloh, J., Naehrig, M.: ElectionGuard - Design Specification 2.0 (Aug 2023)
23. Bulck, J.V., Minkin, M., Weisse, O., Genkin, D., Kasikci, B., Piessens, F., Silberstein, M., Wenisch, T.F., Yarom, Y., Strackx, R.: Foreshadow: Extracting the Keys to the Intel SGX Kingdom with Transient Out-of-Order Execution p. 19
24. Buterin, V.: Trust Models (2020)
25. Buterin, V.: Blockchain voting is overrated among uninformed people but underrated among informed people (2021)
26. Buterin, V.: Crypto Cities (2021)
27. Chaieb, M., Yousfi, S., Lafourcade, P., Robbana, R.: Verify-Your-Vote: A Verifiable Blockchain-Based Online Voting Protocol. In: Themistocleous, M., Rupino Da Cunha, P. (eds.) Information Systems, vol. 341, pp. 16–30. Springer International Publishing, Cham (2019). https://doi.org/10.1007/978-3-030-11395-7_2
28. Clarkson, M.R., Chong, S., Myers, A.C.: Civitas: Toward a secure voting system. In: 2008 IEEE Symposium on Security and Privacy (Sp 2008). pp. 354–368. IEEE (2008)
29. Daian, P., Kell, T., Miers, I., Juels, A.: On-Chain Vote Buying and the Rise of Dark DAOs. https://hackingdistributed.com/2018/07/02/on-chain-vote-buying/ (Jul 2018)
30. Delerablée, C., Pointcheval, D.: Dynamic threshold public-key encryption. In: Advances in Cryptology–CRYPTO 2008: 28th Annual International Cryptology Conference, Santa Barbara, CA, USA, August 17-21, 2008. Proceedings 28. pp. 317–334. Springer (2008)
31. ElSheikh, M., Youssef, A.M.: Dispute-free Scalable Open Vote Network using zk-SNARKs (2022)
32. Ethereum Foundation: Minimal Anti-Collusion Infrastructure. Privacy & Scaling Explorations (formerly known as appliedzkp) (Aug 2022)
33. Gabizon, A., Williamson, Z.J., Ciobotaru, O.: PLONK: Permutations over Lagrange-bases for Oecumenical Noninteractive arguments of Knowledge (2019)

34. Gabizon, A., Williamson, Z.J., Ciobotaru, O.: Plonk: Permutations over lagrange-bases for oecumenical noninteractive arguments of knowledge. Cryptology ePrint Archive (2019)
35. Gennaro, R., Jarecki, S., Krawczyk, H., Rabin, T.: Secure Distributed Key Generation for Discrete-Log Based Cryptosystems. In: Goos, G., Hartmanis, J., Van Leeuwen, J., Stern, J. (eds.) Advances in Cryptology — EUROCRYPT '99, vol. 1592, pp. 295–310. Springer Berlin Heidelberg, Berlin, Heidelberg (1999). https://doi.org/10.1007/3-540-48910-X_21
36. Gentry, C., Halevi, S., Krawczyk, H., Magri, B., Nielsen, J.B., Rabin, T., Yakoubov, S.: YOSO: You Only Speak Once. In: Malkin, T., Peikert, C. (eds.) Advances in Cryptology – CRYPTO 2021. pp. 64–93. Lecture Notes in Computer Science, Springer International Publishing, Cham (2021). https://doi.org/10.1007/978-3-030-84245-1_3
37. Golomb, G.: Believe It: Cybersecurity is Getting Better, Not Worse. https://www.infosecurity-magazine.com/opinions/cybersecurity-getting-better-worse/ (Jan 2018)
38. Goodin, D.: Intel SGX is vulnerable to an unfixable flaw that can steal crypto keys and more. https://arstechnica.com/information-technology/2020/03/hackers-can-steal-secret-data-stored-in-intels-sgx-secure-enclave/ (Mar 2020)
39. Goos, G., Hartmanis, J., Van Leeuwen, J., Schoenmakers, B.: A Simple Publicly Verifiable Secret Sharing Scheme and Its Application to Electronic Voting. In: Wiener, M. (ed.) Advances in Cryptology — CRYPTO' 99, vol. 1666, pp. 148–164. Springer Berlin Heidelberg, Berlin, Heidelberg (1999). https://doi.org/10.1007/3-540-48405-1_10
40. Groth, J.: On the Size of Pairing-based Non-interactive Arguments. In: Annual International Conference on the Theory and Applications of Cryptographic Techniques. pp. 305–326. Springer (2016)
41. Hao, F., Ryan, P.Y., Zieliński, P.: Anonymous voting by two-round public discussion. IET Information Security **4**(2), 62–67 (2010)
42. Heald, G.: A mathematical description of trust. Research Gate, available at: www.researchgate. net/publication/343119798_A_Mathematical_Description_of_Trust (2019)
43. Jie, K.W.: Weijiekoh/elgamal-babyjub (Dec 2023)
44. Laslier, J.F.: And the loser is... Plurality Voting (Jul 2011)
45. Lee, T.B.: Blockchain-based elections would be a disaster for democracy. https://arstechnica.com/tech-policy/2018/11/blockchain-based-elections-would-be-a-disaster-for-democracy/ (Nov 2018)
46. Llaguno, M.: 2017 Coverity Scan Report: Open Source Software—The Road Ahead. Tech. rep. (2017)
47. Mazieres, D.: The stellar consensus protocol: A federated model for internet-level consensus. Stellar Development Foundation **32**, 1–45 (2015)
48. McCorry, P., Shahandashti, S.F., Hao, F.: A Smart Contract for Boardroom Voting with Maximum Voter Privacy. In: Kiayias, A. (ed.) Financial Cryptography and Data Security, vol. 10322, pp. 357–375. Springer International Publishing, Cham (2017). https://doi.org/10.1007/978-3-319-70972-7_20
49. Mearian, L.: Why blockchain-based voting could threaten democracy. https://www.computerworld.com/article/3430697/why-blockchain-could-be-a-threat-to-democracy.html (Aug 2019)
50. Moore, L., Sawhney, N.: The West Virginia Mobile Voting Pilot (2019)
51. Park, S., Specter, M., Narula, N., Rivest, R.L.: Going from bad to worse: From internet voting to blockchain voting. Journal of Cybersecurity **7**(1), tyaa025 (2021)

52. Parno, B., Howell, J., Gentry, C., Raykova, M.: Pinocchio: Nearly practical verifiable computation. In: 2013 IEEE Symposium on Security and Privacy. pp. 238–252. IEEE (2013)
53. Roenne, P.: JCJ with improved verifiability guarantees (2016)
54. Schneier, B.: Blockchain and Trust (2019)
55. Schneier, B.: On Blockchain Voting (2020)
56. Schoenmakers, B.: Lecture notes cryptographic protocols. Department of Mathematics and Computer Science, Technical University of Eindhoven, version **1** (2018)
57. Seifelnasr, M., Galal, H.S., Youssef, A.M.: Scalable Open-Vote Network on Ethereum (2020)
58. Shankland, S.: No, blockchain isn't the answer to our voting system woes. https://www.cnet.com/news/privacy/blockchain-isnt-answer-to-voting-system-woes/ (2018)
59. Votem Corp: Votem/proof-of-vote: Votem's Proof of Vote® protocol whitepaper. https://github.com/votem/proof-of-vote/tree/master
60. Wang, S., Ding, W., Li, J., Yuan, Y., Ouyang, L., Wang, F.Y.: Decentralized Autonomous Organizations: Concept, Model, and Applications. IEEE Transactions on Computational Social Systems **6**(5), 870–878 (Oct 2019). https://doi.org/10.1109/TCSS.2019.2938190
61. WhiteHat, B., Baylina, J., Bellés, M.: Baby Jubjub elliptic curve (2020)
62. Williams, N.: Remote Voting in the Age of Cryptography. MIT Computational Law Report (Dec 2022)