Zero-Knowledge Proofs for Secure and Private Voting Systems.

Individual Component - Multi-Party Computation based Vote Counting system using Advanced Cryptographic Technique.

Project ID – IT21318252

Final Project Thesis

Student ID: IT21303920

Name: Mushtaq M. B. M

Supervisor: Mr. Kavinga Yapa

B.Sc. (Hons) Degree in Information Technology Specializing in Cyber Security

Department of Information technology

Sri Lanka Institute of Information technology

April 2025

Table of Contents

Introduction

The accelerated movement toward the digital means of governance and services has created an increased interest in adopting new forms of electronic voting (e-voting) systems. E-voting systems are acclaimed for enabling faster, easier, and more efficient elections, particularly in the case of large election events or those without clear boundaries (like the Canadian Federal Election) in sparsely populated areas. Nonetheless, digital voting introduces a number of challenges with security, privacy, transparency, and trust that are essential components to the defined characteristics of democracy. Such high-profile controversies such as the Kenyan elections of 2017 and the U.S. elections of 2020 have exposed the weaknesses of e-voting systems based predominantly on centralized infrastructure with opaque counting features and limited options for post-election examination of votes. This underscores the necessity of an alternative approach to e-voting systems based on secure, decentralized, and verifiable counting. [1]

Current e-voting architectures generally involve either centralized tallying systems that risk manipulation and single points of failure, or homomorphic encryption, which is privacy-preserving but often computationally intensive and suffers scalability issues. Other approaches, such as mix-nets and blind signatures, or block-chain based platforms, partially improve privacy of votes or auditability, but often not both, whilst not being able to simultaneously combine decentralized threshold, lightweight computation, and verifiable eligibility in either a transparent and practical system.

This research gap demonstrates a need for an overall vote counting framework that is secure, scalable and transparent that employs Multi-Party Computation (MPC) with Shamir's Secret Sharing (SSS) to decentralize the tallying process, but incorporates zk-SNARKs for verification of voter eligibility and blockchain for auditability. There are very few plausible designs that bring everything together or can provide end-to-end guarantees from vote casting to tally verification without compromising voter privacy or system performance.

The research problem this thesis addresses is the design and implementation of a privacy-preserving, threshold-based vote counting system that decentralizes control, preserves vote secrecy, even while the vote is being computed, and fully auditable and trustworthy to all parties.

Literature Review

Evolution of Digital Voting

The evolution of voting systems—from a simple paper ballot system to electronic voting machines (EVMs) and, more recently, Internet and blockchain-based systems—was guided by a series of clear values: convenience, accessibility, and administrative efficiency in the conduct of elections. The development of new voting technologies aimed at reducing manual errors, speeding up the return of results, and enabling voter turnout through more remote participation of citizens who otherwise faced obstacles to voting. However, as voting systems move towards a more digital approach, they will typically generate new and increasingly more significant challenges as they apply to privacy, integrity, transparency, and public trust. [2] [3]

The initial digital systems that employed electronic voting used centralized servers to receive and tabulate votes. These systems permitted electronic vote counting and eliminated some of the logistical work associated with having an election, but the drawback was serious. Centralized systems perform poorly when it comes to the qualities one expects from an election: centralized systems have a single point of failure, leaving the entire system vulnerable; centralized systems have the potential for insider tampering and are exposed to outside cyberattacks; and, neither the voter, observer, or candidate can determine whether the vote was truly cast, recorded, or counted as originally intended, or otherwise. In fact, these early digital systems became synonymous with "black-box" systems, even if overtly called a digital voting system.

In instances of an early digital system with a poor outcome in a politically sensitive or heatedly contested election, there was heightened skepticism that the system operated appropriately when the system can have erroneous outcomes due to the firewalls that separated itself from the worled. The problem with that is, if something goes wrong, like unexpected software errors, system crashes, or delays publishing results, the presence of a verifiable system or an alternative approach to auditing in many early digital systems did not afford the ability to provide any sort of accountability to any voters or all voters.

The limitations in many of these systems recognized the need for alternatives that allowed for voting and to improve upon traditional centralized digital systems as alternatives to polling activation in more secure, transparent and independent environments, while respecting the democratic principles of anonymity, accountability, and fairness within acceptable processes while continuing to embrace advances in technology. This thesis addresses these challenges by introducing a **distributed, cryptographically secure framework** for vote counting using MPC and Shamir’s Secret Sharing. [4]

Centralized Vote Counting: A Critical Weakness

In conventional electronic voting systems, voting typically occurs through a process of encrypting votes during transmission, which are then stored in a central database until the conclusion of the election. Once all of the votes have been collected, the votes are decrypted and tallied at a central location. This can be helpful for efficiency and speed, but it creates a significant vulnerability due to a single point of failure. If the central point is compromised by a cyber-attack, insider threat, or software failure, votes can be changed, deleted, or made up altogether without easy detection, and the final outcome can be easily modified without easy detection. [5]

Additionally, most of these systems lack end-to-end verification. The reality is that there are typically no means for either voters or independent auditors to verify that a vote was recorded and counted as cast. This lack of verification yields low voter confidence, especially in high-stakes or closely contested elections. [6]

The lack of verifiable records and systems for substantial auditing in numerous early systems further diminished trust. These shortcomings have emphasized the demand for secure, transparent and verifiable options for conventional centralized digital voting systems that would fundamentally embrace privacy, accountability and fairness as democratic norms while still availing themselves of the benefits of new technologies.

Researchers and election security scholars have advised ways to implement a decentralized system that is cryptographically secure. These are implemented in a way that distributes the vote storage and vote tallying obligations toward multiple independent nodes, thus minimizing risk around a single point of failure. Moreover, election security scholars recognize that a cryptographically assured method to include end-to-end verifiability would permit voters to independently verify that their votes were, in fact, cast, recorded and counted correctly. The basis for enhancing transparency and trust in electoral systems is highlighted within the provisions to expand the data available on the processes that have participated in the operations surrounding necessary verification of election outcomes.

Establishing the cutting-edge secure voting systems will require researchers, security scholars and voters to simultaneously assess usability, scalability and accessibility; meeting the appropriate balance for significant adoption by election administrators, voters, and globally across society inexpensively. Ongoing research and pilot programs are exploring the viability of these voting systems to satisfy security requirements, information assurance requirements and realizing what voters and election administrators need, in terms of limitations and options of use for the voting and elections process. [7]

Cryptographic Voting Protocols

The increase in issues of privacy and integrity associated with online voting systems led to the development of various cryptographic protocols. One such method, homomorphic encryption, allows for total voting counts to take place on encrypted data, meaning that totals can be derived without ever decrypting votes. This works well to keep votes confidential but incurs significant computational overhead and exposes the system to risk from any negligence or collusion by anyone with access to the private decryption key that can otherwise compromise the entire system.

There are other methods that exist, such as blind signatures and mix-nets, which seek to allow for verification of the vote while ensuring anonymity. Blind signatures allow voters to receive valid acceptance without showing yet yielding the content of their vote. Mix-nets obfuscate the link between the voter and the ballot through the shuffling of encrypted votes. While these types of systems do exist while accomplishing similar objectives they are usually complex, not scalable, and have potentially challenging implementations, reducing their feasibility in actual large scale elections. The need for a lower complexity distribution based or threshold voting system is needed instead.

Multi-Party Computation (MPC)

Multi-Party Computation (MPC) is an essential cryptographic tool, that allows a set of participants to compute a function on their inputs without revealing to each other their respective inputs. This method allows for cooperative computation with guaranteed privacy, for participants' sensitive data is not revealed nor disclosed at any time, thus a collaborative process can be sustained. Regarding electronic voting, the use of MPC represents an appealing method for secure vote counting since votes can be collated and tallied without revealing how anyone voted, and preserving the secrecy of the ballot.

Unlike to a traditional voting system using a central authority to move from votes into a decrypted tally, MPC distributed system computation is accomplished through independent parties and each independent party has no means to learn about the votes' content, thus the final tally of votes was independently constructed. There are also vote manipulations, and independent parties have knowledge of how a vote is determined by an individual, and the vote is recorded at that time, making the privacy of the vote in the tally much more secure. Therefore, the decoupled model reinforces protection with independence and the trust element is included in the transparency of the independent parties of the tally, the level of protection from third parties that can maintain manipulation and third parties from learning sensitive data is well vetted

today under e-voting systems and this is eliminating in insider threat context manipulation or learning about votes they should not have access to or leaking voting instructions. [8]

Multiple protocols have shown, in theory, the feasibility of MPC in voting settings. The most notable include:

- The BGW protocol (Ben-Or, Goldwasser and Wigderson) which is executed over secret-shared inputs and is provably perfectly secure against adversarial parties as long as certain bounds are met.
- Yao's Garbled Circuits protocol is executed by one party preparing an encrypted Boolean circuit, while the other party evaluates it without uncovering the original inputs.
- The GMW protocol (Goldreich, Micali, and Wigderson) has an interactive approach based on oblivious transfer and computations based circuits.

Even though these protocols have provable security, they are very rarely deployed in real elections for a variety of practical reasons: First, because many are very expensive because of their computational and communication overhead, which is exacerbated when you consider they need to scale up for elections with thousands and millions of voters. Second, many of these protocols are complex – the circuit generation process, secure distributions of inputs, and ensure the input happens in a synchronized way adds complexity to any e-voting platform adding additional points of possible failure. Third, issues include latency and its associated problems, fault-tolerance, and the dependence on a network for its overall scalability, especially true since low-tech regions have limited access to technology.

Thus, while MPC provides a strong, privacy-preserving, and tamper-proof mechanism for framework vote counting, practical deployments must be designed with efficiency, modularity, and scalability in mind. We tackle these problems in this thesis by combining MPC with Shamir's Secret Sharing (SSS), zk-SNARKs, and blockchain into a lightweight and secure framework, suitable for today's election contexts.

Shamir’s Secret Sharing (SSS)

Shamir’s Secret Sharing (SSS) is the foundation of threshold cryptography, and its function is to provide strong security properties in distributed systems. SSS employs polynomial interpolation over finite fields, allowing a secret to be divided into n separate shares such that any t shares can reconstruct the original secret. Most notably, for any number of shares less than t, those shares give absolutely no information about the secret, offering perfect secrecy and protection against partial compromise.

In digital voting systems, SSS is critical to protecting a voter's secret ballot and maintaining a distribution of trust. Each vote, or intermediate computation output (e.g. the partial tally), can be treated as the secret

that gets encoded as a secret and then split into shares. The shares are distributed to independent count authorities or nodes. Because no single node has enough information to reconstruct a vote, that single node cannot perform tampering or that node cannot perform surveillance. The original information can only be reconstructed when a sufficient threshold of parties (e.g., 2-of-3 or 5-of-7) is reached, making SSS systems resilient to faults and collusion.

Perhaps the clearest advantage of Shamir's Secret Sharing is its efficiency. It is a lightweight, mathematically simple solution(reduction strategies with polynomials and basic arithmetic modulo p) that is far less expensive to execute than other alternatives like homomorphic encryption or conventional zero-knowledge systems. These advantages further establish SSS as a solution that suits real-time applications, resource-constrained systems, or even large- scale electoral systems where efficiency is a priority. [9]

When SSS is used in conjunction with Multi-Party Computation (MPC), the election authority may securely aggregate all of the vote shares without revealing the compute values of individual votes. Each node computes only its own share, and the vote total is finally reconstructed collectively using Lagrange interpolation. This method achieves a fully privacy-preserving, verifiable and distributed vote counting process which embodies the notions of democratic transparency and cryptographic security..

Blockchain in Voting and Counting

Blockchain technology has gained traction as a meaningful method for securing voting systems from a breach of integrity, transparency, and decentralization. In a digital election context, blockchain provides a secure and immutable medium for vote recordkeeping, where all voting data is cryptographically linked in a series of blocks. Each block can house vote-related information, and once information is signed and locked into the blockchain it is nearly impossible to modify or delete that information. To modify or delete any part of the blockchain requires modifying or deleting every subsequent block in the chain, which usually cannot be done without the consensus of a majority of participants on the network. As a medium for voting, the blockchain provides tamper-resistance and data immutability, which are critical to maintain the integrity of voting systems.

The cryptographic properties of blockchain technology allows voting on blockchain to be publicly verifiable. Stakeholders involved in the voting process, which may include election monitors, voters, or auditors, can use the blockchain to follow the voting process and verify that their vote is counted correctly, while ensuring the contents of their vote is not disclosed. The ability to publicly verify the voting process means that the voting system can be audited independent of the voting users, while still allowing the voters

their privacy. With recorded encrypted votes on an immutable, decentralized ledger, blockchain voting technology eliminates risks associated with centralized systems being the target of a cyber-attack or manipulation.

Game Changer, Follow my Vote and Voets all have sought to explore the possibilities of using blockchain for digital voting. They are notable because they have incorporated blockchain into their voting technology. Not only have they included blockchain, while still heavily relying on proprietary voting systems, Estonia continues to use blockchain in their national e-voting system, but for audit logging, not collecting votes. While the blockchain voting systems that have been established have taken a meaningful step forward, to date most have relied upon more traditional approaches of either encryption or forming a homomorphic voting approach for vote confidentiality instead of utilizing more advanced cryptography (i.e. Multi-Party Computation (MPC), Shamir's Secret Sharing (SSS)) for secure and decentralized, confidential vote tallying. This gap presents a significant opportunity for improvement, as integrating MPC and SSS can provide both privacy-preserving vote counting and collusion resistance while enhancing overall system security and scalability. [10]

Zero-Knowledge Proofs (zk-SNARKs)

zk-SNARKs (Zero-Knowledge Succinct Non-Interactive Arguments of Knowledge) enable a voter to prove eligibility—such as being registered and not having voted yet—without disclosing their identity or vote content. This preserves **anonymity** while ensuring the **integrity** of the voting process. zk-SNARKs are both **highly efficient** and **compact**, making them ideal for **anonymous identity verification** in large-scale election systems, where performance and scalability are critical. [11]

Innovations in zk-SNARK frameworks (e.g. Groth16, PLONK) have improved their speed and utilization, which lowers the barrier for real-world adoption. Still, despite these innovations, only a handful of voting systems to date efficiently incorporate zk-SNARKs in a cohesive architecture with Multi-Party Computation (MPC) and blockchain technologies to provide privacy, verifiability, and decentralized trust. [12]

Research Gap

Even with ample progress in the creation of secure digital voting protocols, many relevant issues still need to be addressed, which restricts the deployment of completely secure, transparent, and privacy-preserving election systems.

- **Centralization in Vote Counting:** A large portion of the current digital voting systems - particularly those using blockchain - still rely on centralized components as a requirement for decryption and final counting. Even if the voting was decentralized, the final computation stage is often rooted in a trusted authority, or group of decentralized servers with a central decryption piece that counts the votes. This reintroduces a single point of failure and hindrance to trustless and decentralized purposes which leads to skepticism and manipulation.

- **Lack of Practical MPC-based Counting:** Existing systems can find inherent value in MPC, which has had a strong theoretical basis in which it could make the value-input private when doing the computation on these inputs. In practice, theoretical benefits of MPC are absolutely underutilized compared to today's voting platforms. Current implementations of MPC will take on complex states during implementation, they use significant resources and don't often have streamlined frameworks, while still being open to challenges of theory v.s. practical implementation.

- **Not Integrating Privacy and Auditability Together:** Many systems may only focus on a manipulation for one security property at a time. In some cases, their only safeguard for vote privacy during the collection is using encryption, while others only ensure transparency using blockchain exploits for logging. However, very few systems successfully **merge both aspects**, offering privacy-preserving yet publicly auditable elections in a seamless and efficient manner.

- **Limited Usage of Shamir's Secret Sharing Protocols (SSS):** SSS is lightweight, secure, proven to be effective for eligibility checks, and perfect for threshold-style vote distribution; however, SSS is virtually never used in modern voting systems. Most systems to this day only consider encryption-based tallying, and SSS is often ignored despite its inclusion of collusion resistance, fault tolerance, and threshold-like implementations.

- **zk-SNARKs Are Not Used in Full Voting Pipelines:** Although zk-SNARKs are effective in proving anonymous eligibility-giving, zk-SNARKs rarely integrate both MPC and SSS together in

end-to-end real-world applications. A literature gap this significant in each of these technologies working in an end-to-end pipeline still has a ton of promise toward a digital voting solution.

- **Hence,** there is a clear gap and need for an end-to-end practical digital voting solution that incorporates multiple advanced cryptographic technologies. As we move to use zk-SNARKs allowing for anonymous and real verifiable eligibility assessments, SSS providing secure and private vote distribution, and MPC allowing for decentralized and tamper-free tallies; it is obvious that there is no solution established in the field that encompasses all of these advanced technologies together. If we added blockchain technology providing auditable ways to offer verifiable continual receipts of the voting procedure (plus verify on the blockchain in public) this model provides an end-to-end digital voting solution to cover privacy, security, integrity, and trust of the voting process as one secure framework ready for real-world elections. [13]

Research Problem

Developing a decentralized vote counting system that achieves a balance among privacy, security, transparency, and verifiability is a crucial step towards rebuilding confidence in digital elections. The challenge is to create a provably secure system that can maintain voter anonymity between the voting and tallying phases, remove dependence on centralized trusted authorities, and provide public verifiability and auditability without sacrificing efficiency or scalability.

Traditional electronic voting systems regularly depend on a centralized trusted infrastructure for collecting and decrypting votes, and tallying results. While they can potentially produce very quick election outcomes, they are procedurally weak and have intrinsic problematic characteristics, and maintain official opacity about the processes that prevents the public from verifying the content. A technician formerly in charge of creating a central tallying server may have access to the server and could alter the outcome of the election or suppress an outcome without anyone knowing right away. Technically, when encryption is required to manage privacy for individual votes, there remain phases of decryption within flagged central tallying votes. With the decryption phase, the potential for interference also increases. These attributes prioritize knowing and following the process with a computer but eliminate the trust aspect regarding voting as a democratic process. Thus, they are inappropriate for high-stakes, large-scale, democratic procedures.

This research addresses the serious shortcomings inherent in these types of systems by developing an end-to-end decentralized vote counting system leveraging modular advanced cryptographic tools in a practical and instructive framework.

First, Multi-Party Computation (MPC) with Shamir's Secret Sharing (SSS) is used to transparently and collaboratively aggregate vote totals. In this method, individual votes are divided into several cryptographic shares according to the SSS protocol and distributed across a set of independent tallying authorities. The votes can only be reconstructed if a threshold number of parties collaborate, which guarantees that no parties ever have access to an entire vote, and provably prevents collusion with respect to vote secrecy, even during the tallying phase.

Second, zk-SNARKs (Zero-Knowledge Succinct Non-Interactive Arguments of Knowledge) are used for voter eligibility proof and to ensure that not more than one vote is submitted per voter, while completely protecting voter anonymity. Each voter produces proof that they are registered and have not previously voted, providing this proof without revealing any voter identities or vote content. This proof is verified before the respective party accepts the vote and distributes the vote among parties.

Finally, every action taken by the voter, including proof submission, vote casting, and share-distribution, are recorded on a blockchain ledger. As an immutable ledger, it is completely transparent, providing public verification and auditability to the process, to the extent that every part of the process that occurs can be verified and traced by any third party without opening access to the secrecy of the vote. The blockchain also acts as a tamper-resistant communication channel between voters and tallying parties.

By combining MPC, SSS, zk-SNARKs, and blockchain, this research presents a novel approach to building a **secure, scalable, and trustless digital voting system** suitable for real-world deployment.

Research Objective

Main Objective

This research's principal focus is designing and deploying a secure, scalable, and privacy-preserving vote counting system based on an adoption of **Multi-Party Computation (MPC)** in conjunction with Shamir's **Secret Sharing (SSS), zk-SNARKs,** and **blockchain technologies**. The goal is to surmount the weaknesses of conventional electronic voting systems by eliminating centralized counting, preserving voter privacy in all cases, and generating tamper-resistant and fully verifiable public records of events of elections.

Sub-Objectives

1. **Vote Privacy**

   The system must guarantee that individual votes remain confidential throughout the entire voting lifecycle. Privacy must be preserved not only during vote casting but also during tallying, such that no party or observer can determine how any voter cast their vote.

2. **Threshold Tallying with Shamir's Secret Sharing**

   Using SSS, each vote is split into multiple shares and distributed to independent tallying parties. A **threshold scheme** ensures that only when a quorum (e.g., 5 out of 7) of parties collaborates can the votes be reconstructed. This setup promotes **collusion resistance** and ensures that no minority group of parties can tamper with the results.

3. **Distributed Counting via Multi-Party Computation**

   Instead of relying on a central tallying authority, the system uses MPC to allow distributed computation of the final tally. Each party holds only partial information, and the computation is designed so that no single entity can access, alter, or reconstruct vote data independently. This provides strong security and integrity guarantees.

4. **Eligibility Verification Using zk-SNARKs**

   The system integrates zk-SNARKs to verify that a voter is eligible (i.e., registered and has not voted yet) without revealing their identity or personal data. These proofs are succinct and efficient, making them suitable for large-scale elections while maintaining voter anonymity and preventing fraud.

5. **Immutable Recording on Blockchain**

All actions, including vote submissions, eligibility proofs, and share distributions, are recorded on a blockchain. This ensures **transparency**, **immutability**, and **tamper resistance**, enabling any party to independently audit the voting process without accessing vote contents.

6. **Resilience and Fault Tolerance**

   The system is designed to remain functional even if some tallying parties go offline or become unresponsive. The threshold mechanism ensures that as long as the minimum number of trusted parties is available, accurate vote reconstruction and result computation can proceed without loss of data or trust.

Research Questions

- **How can MPC and SSS be practically applied** to enable secure, privacy-preserving vote tallying without revealing individual vote values to any party?
- **What is the optimal threshold configuration** (e.g., 5-of-7, 3-of-5) that provides a balance between security (against collusion) and fault tolerance (against dropout)?
- **How can zk-SNARKs be integrated effectively** to ensure voter eligibility, prevent duplicate voting, and maintain voter anonymity?
- **What are the system's performance and scalability trade-offs** when applied in real-world election scenarios with thousands or millions of voters, and how can these be mitigated?

Methodology

This chapter provides a clear description of the methodology used to design and implement a distributed, privacy-preserving vote counting system that uses state-of-the-art cryptographic methods to ensure the integrity, transparency, and security of the voting process. The vote counting system employs Shamir's Secret Sharing (SSS) multi-party computation (MPC) and zk-SNARKs for proof of eligibility, where blockchain acts as an immutable repository. The methodology is structured to address the key objective of this thesis: eliminate the need for a trusted central authority while preserving voter privacy, as well as system robustness and end-to-end verifiability.

We will begin with the user casting a vote. Each valid vote will first be verified for eligibility with zk-SNARK proofs. This confirms to the voting system that the voter is registered and has yet to cast a vote without revealing any information as to who the voter is and what they voted for. This prevents two things: anonymity and double voting. After the voting system confirms eligibility, the vote is submitted to the blockchain and cryptographically secured. The blockchain #IBMzKP-Token serves as an immutable ledger that is public and tamper-proof.

After submission, the vote is **encoded into a secret value** and split into **multiple cryptographic shares** using **Shamir's Secret Sharing (SSS)**. Each share is assigned to an independent **vote counting party**. These parties could be distributed among different organizations, election observers, or trusted entities. The shares are distributed in such a way that no single party can reconstruct or alter the original vote independently. Only when a sufficient number of parties (the threshold, e.g., 5 out of 7) collaborate can the original vote be reconstructed. [14]

In the vote counting phase, the parties will do Multi-Party Computation (MPC) to securely aggregate the votes, with each party computing only on its own share and the eventual result determined by all parties together. This means there is no possibility for anyone party to unduly influence the vote counts, and the aggregate of votes can be securely created. The final vote total was constructed from the shares given by the parties using Lagrange interpolation. The system's threshold was designed to be fault tolerant: even some parties can fail or be unresponsive without stopping the whole system as long as the threshold of parties are met, which provides resilience from attacks or failures.

Every operation, from the zk-SNARK proof verification, to vote casting, share generation, and vote tally, is recorded onto the blockchain ledger so everything is immutable. The immutable nature of a blockchain ensures full transparency and auditability of the revolution while maintaining voter anonymity. The

blockchain ledger can provide an irrevocable record that can be verified publicly by external third parties with appropriate permits, enabling reasonable and transparent external audits without violating voter anonymity. This architecture enables the system to be tamper resistant, fully transparent and maintain verifiable integrity in all phases of the process, ensuring a secure, decentralized voting solution that ensures fairness and trust.

Vote Counting System Overview

The vote counting process consists of four separate yet connected stages in the vote counting in the election process, in a way that can keep privacy, integrity and decentralization of the election process at every single step. The stages are of the process, in a way that no one can change or expose the voting tally in any way at any time, and that the election is legitimate and trustworthy in every way..

1. Vote Validation using zk-SNARKs (Not Detailed in this Chapter)

Prior to a vote being accepted into the system, it must be verified for eligibility using Zero-Knowledge Succinct Non-Interactive Arguments of Knowledge (zk-SNARKs). This enables the system to confirm that the voter is a registered voter and has not already voted, while being otherwise eligible to vote. Notably, eligibility verification, including confirming an individual's voting status (no more than one vote), is a non-interactive process that does not expose any personally identifiable information about the voter or the voter's selection, and so does not violate voter anonymity. The zk-SNARK proof prevents double voting while ensuring that no identities get revealed to any parties, which is essential to demonstrating trustworthiness of the election administration system. That verification step allows for only legitimate and unique voters to engage in the voting process.

2. Vote Encoding and Secret Sharing using Shamir’s Secret Sharing (SSS)

Once verified, the vote is translated into a numeric value and split into multiple cryptographic shares with Shamir’s Secret Sharing (SSS). Each vote is split into n shares (such as 7) and the shares are then individually sent to different tally parties and securely distributed. Since SSS ensures that no share has any useful information about the original vote, we can distribute the share among different parties, ensuring that there is no way for the parties to know the voters' votes. Each party holds only a portion of the original data, and the complete vote reconstruction is impossible without a minimum number of parties working together.

3. Distributed Vote Counting via Multi-Party Computation (MPC)

During this stage, each party will count their share of the vote through the use of Multi-Party Computation (MPC). Each party only has its own share of the vote; no party has access to the complete vote. Each party will do calculations on their own share. No production or exchange of the vote data occurs either since computations occur via MPC. MPC is called upon to facilitate the secure counting of each share of the vote without exposing the individuals' votes (votes are not revealed to each party). In addition, there is no way for tampering, exposure of data, or access to forbidden votes during the counting process.

4. Final Reconstruction of Vote Totals using Lagrange Interpolation

Once all of the shares are collected and summed, the system uses Lagrange interpolation to reconstruct the final total votes for each candidate. The reconstructed tally comes from a threshold number of shares - for example, in the case of 7 parties needing to collaborate 5 parties to form a final result. When it comes to the threshold design, it is impossible for a single party (or a small group of parties) to reconstruct the final result - or collude as a single party to manipulate the final result - as it must be agreed to match the results from the votes and as long as there is some fraction, there can be collusion resistance and fault tolerance. Only when a number of parties agree to the reconstruction can vote totals be finalized.

Throughout each of these stages, the outlined procedure aims on producing a secure and decentralized voting system which provides privacy assurance, is resistant to fraud and manipulation, and is tolerant to failures for trustworthy and verifiable elections.

Shamir’s Secret Sharing (SSS) for Vote Splitting

Why SSS?

Shamir's Secret Sharing (SSS) is a strong cryptographic mechanism of distributed secret sharing that is useful across a range of applications, particularly in secure voting. SSS divides a secret (or, a vote) into shares that are distributed to different parties, with the property that only a pre-determined threshold number of shares must collaborate to raise the original secret. The key advantage of SSS is that reconstruction is only possible if parties reach this fixed number, and vote elements (or sensitive data) cannot be assembled without pre-coordination. This pre-determined parameter - let’s call it tt - is an essential anti-aggregation feature of SSS. It shows that any number of subsets of shares that are less than tt yields absolutely no information about the original secret which means perfect secrecy for the vote. The whole system of SSS

rests on the assumption that any attempts to infer the vote value if the threshold is not jointly reached would always end up being completely unsuccessful.

SSS is secure, lightweight, and computationally inexpensive. While more complicated cryptographic systems, like homomorphic encryption and fully encrypted voting schemes, demand higher computational costs, Shamir's Secret Sharing is grounded in polynomial-based arithmetic over finite fields, which is less computationally demanding. This gives it a favorable position for real-time applications or resource-constrained applications. For example, SSS is perfect for low-cost voting systems and devices with modest computational power, like a cloud application executing in-person voting. Also since SSS uses simple arithmetic and there are multiple share sites involved, the time to share can be parallelized, SSS can also be scaled for enormous elections.

SSS' another major advantage is that it allows threshold-based access control. In the context of voting, sensitive operations that have a lasting effect on the outcome (like vote reconstruction) can only happen when a predetermined set of independent parties come together to make it happen. This aligns with the basic principles of distributed trust where no one party has complete control over the election. It also makes the system inherently resistant to collusion; if parties try to collude even if some partied together to manipulate the results, as long as the threshold for honest parties is retained the system remains secure. This threshold design promotes collusion resistance and fault tolerance because the integrity of the election results stand ready even if there is a collusion attempt. SSS helps mitigate problems caused by centralize or total control over trust. By separating trust amongst parties and requiring quorum size mandate equitable control over vote reconstruction, SSS allows a secure, transparent, and resilient voting system to emerge.

The Mathematical Foundation

In Shamir’s Secret Sharing (SSS), a **secret** ss (e.g., a vote encoded as 1 for a vote to candidate A, or 0 otherwise) is split into multiple shares and distributed among **n** parties in a way that ensures **privacy and threshold-based reconstruction**. The scheme defines a **random polynomial** of degree t−1, where t is the threshold number of parties required to reconstruct the original secret. The polynomial is constructed as:

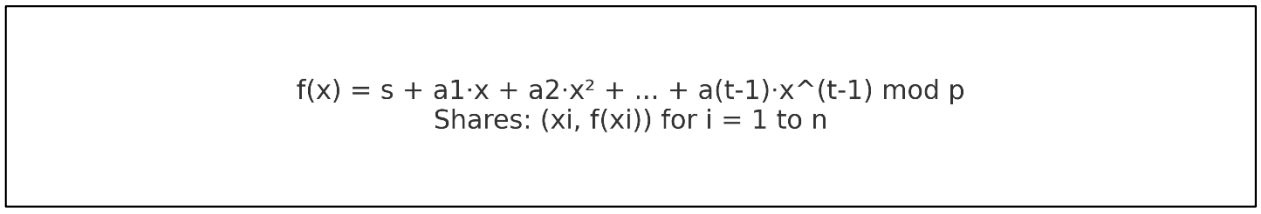

Fig. 1. Degree of Polynomial Formula

Here, S is the constant term (the secret), and a1, a2, ..., a(t-1) are randomly chosen coefficients in a finite field defined by a large prime p. Each share is a point on the polynomial, represented as (xi, f(xi)), where xi is a unique point and f(xi) is the corresponding value of the polynomial at xi. These shares are then distributed to n distinct parties. The original secret S can only be recovered using Lagrange interpolation if at least t valid shares are collected. With fewer than t shares, the secret remains completely hidden, providing information-theoretic security.

Share Generation Process

The generate_shamir_shares(secret, num_parties, threshold) function is responsible for securely splitting a secret (such as a vote) into multiple cryptographic shares using Shamir's Secret Sharing (SSS). The process begins by randomly generating t-1 coefficients, where t is the reconstruction threshold. These coefficients, along with the secret as the constant term, form a unique polynomial of degree t-1 over a large prime field. The next step is to evaluate this polynomial at n distinct non-zero x-values, where n is the total number of participating parties. Each evaluation yields a coordinate pair (xi, f(xi)), which represents a share of the secret.

These shares are then securely distributed to different parties. No single share reveals any information about the original secret, and only when at least t shares are combined using Lagrange interpolation can the secret be reconstructed. This ensures both privacy and collusion resistance.

Distribution Strategy

Each generated share is assigned to a unique **vote counting node** (e.g., Party 1 to Party 3), ensuring that control over the vote data is distributed across multiple independent entities. This approach eliminates the risks associated with centralized systems, such as single points of failure or tampering. Each vote counting party securely stores its received shares in a **local data structure**, organized by both **party** and **candidate**. The shares for each candidate are stored in a manner that allows easy aggregation later in the tallying process. The system utilizes the data structure self.party_shares[i][candidate].append(share) to keep track of the shares for each party and each candidate. This ensures that each party only has access to a portion of the vote data, preventing any individual party from tampering with the votes or learning how any voter voted. This organized structure facilitates secure and structured vote tallying, with each party contributing to the final count in a privacy-preserving manner.

Multi-Party Computation (MPC) for Vote Tallying

Secure Tallying Logic

The secure tallying mechanism ensures that **no individual vote is ever exposed**, even during the counting phase. Each vote, once validated and accepted, is first **encoded** and then **split into multiple cryptographic shares** using **Shamir's Secret Sharing (SSS)**. These shares are securely distributed across a **network of independent vote counting nodes (parties)**. Each party receives and stores only a partial view of the vote, ensuring that no single node can reconstruct or tamper with the vote data on its own.

During the tallying phase, shares corresponding to each candidate are summed locally by each party. Then, a **subset of parties**—equal to or greater than the threshold t—**collaborate** to reconstruct the total number of votes per candidate. This process ensures **fault tolerance** and **collusion resistance**, as fewer than ttt parties cannot reconstruct or infer the final tally.

Lagrange Interpolation

To reconstruct the vote total, the system employs **Lagrange interpolation**, a mathematical technique used to recover a polynomial from a set of known points. Specifically, given **t** shares of the form (xj,yj), where yj=f(xj), the constant term f(0) — which represents the secret or vote total — is computed using the following formula:

$$f(0) = \sum_{j=1}^{t} y_j \cdot \prod_{1 \le m \le t,\, m \ne j} \frac{-x_m}{x_j - x_m}$$

Fig. 2. Lagrange interpolation formula

This formula allows reconstruction of the original vote total (the constant term of the polynomial) without ever revealing individual votes. The method is implemented in the function:

def lagrange_interpolation(x, x_values, y_values, prime):

This function computes the **modular inverse** and evaluates the **weighted sum** of the shares to retrieve the original vote total securely. The use of **modular arithmetic** over a prime field pp ensures **cryptographic integrity** and prevents overflow or tampering during the reconstruction process.

Reconstruction Flow

During the final tallying phase, the system securely combines the votes and reconstructs each candidate's total number of votes from the shares distributed to each vote counting node. For each candidate, the process begins by collecting all of the (x,y) shares shared with each party node, which each party node had kept as a local copy during the phase of when the votes were cast.

After all the shares are collected, the system checks to ensure that the total of shares available for a candidate is equal to or greater than the pre- defined value of tt. This value is important in Shamir's Secret Sharing because tt is the minimum number of shares required to reconstruct (i.e., calculate the original value) — in this case, the total number of votes which the candidate received.

If tt has been met, the system will then call the reconstruct_shamir_secret() function, which uses Lagrange interpolation, in order to securely and accurately reconstruct the candidate vote count for aggregation. If the number of available shares is below the threshold, the system will log a warning stating that there were not enough shares to complete the reconstruction. This approach maintains vote privacy while ensuring **fault tolerance**, allowing accurate counting even if some parties go offline or fail to submit their shares.

Vote Counting Using MPC and Shamir's Secret Sharing

In the proposed e-voting system, **Multi-Party Computation (MPC)** is employed as a core privacy-preserving technique to securely carry out the vote counting process while safeguarding voter anonymity and ensuring tamper resistance. To implement this securely, the system leverages the well-established **Shamir's Secret Sharing (SSS)** algorithm, which enables a vote to be mathematically split into multiple independent shares. These shares are then distributed among separate computation nodes hosted on physically distinct machines within a local network, thus providing a simulated decentralized infrastructure that aligns with real-world deployment models.

Vote Splitting

The process begins when a voter casts their ballot—for example, by selecting Candidate A. This choice is first converted into a numeric value (e.g., ASCII encoding or predefined candidate mapping). Using the Shamir's Secret Sharing scheme, the numeric vote is split into *n* different shares, each of which represents a unique fragment of the original vote. The system is configured with a *threshold* parameter *t*, meaning that any *t* out of *n* shares are required to reconstruct the vote. For instance, in a (3-of-5) threshold scheme,

five total shares are generated, but only three are necessary to recover the original value. This ensures both **fault tolerance** and **privacy**.

Share Distribution

Once created, each share is sent via a secure channel to an independent MPC server. During the experimental implementation, the servers were deployed in real time on three separate physical machines connected through a local area network (LAN). This design was intentionally developed to mimic stochastic decentralization and created a higher degree of fault isolation. By deploying shares to different servers we prevent the ability for a single node to gain enough information to re-construct a vote on its' own which thwarts collusion and leaking data from compromised machines.

Threshold Scheme

A threshold-based secret sharing scheme is inherently fault tolerant against partial failures. Even if one or two nodes are out of commission because of hardware failures, network issues, or malicious activity, the remaining nodes can still work together to complete the required computations. This built-in characteristic gives resilience against DoS and insider attacks while sustaining functionality.

Secure Aggregation of Votes

In the vote tallying stage, the system does not replicate votes. Instead, it does a secure aggregation of enough number of shares to obtain the total count of votes. The aggregation is structured to be distributed so that several nodes of the MPC can perform the aggregate without revealing vote information. The aggregation is done collectively so that the totalizing of the vote is obscured in several different computations. These computations offer confidentiality to the voter throughout the process. Combining MPC and Shamir’s Secret Sharing and distributes processes amongst several machines achieves security, scalability, and transparent record keeping for electronic voting.

Example Illustration

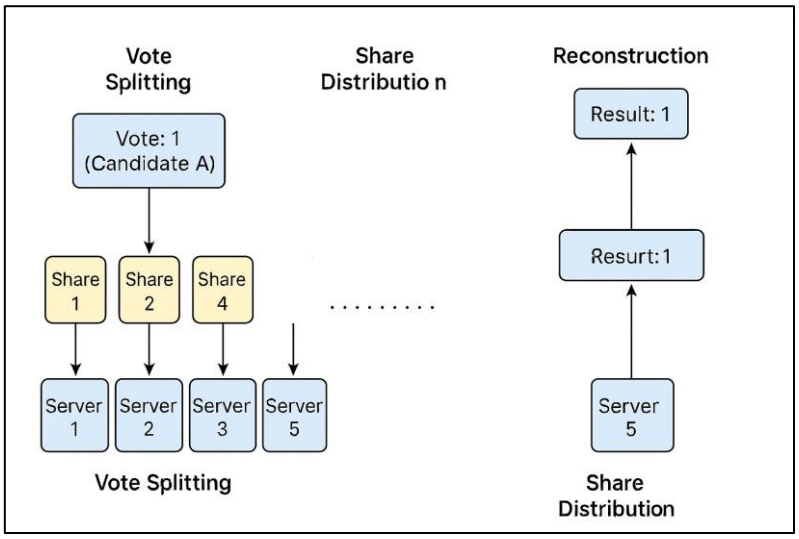

Fig. 3. Example Ilustration of code splitting, share distrbution and reconstruction

To demonstrate the secure handling of votes in the proposed system, let's take the example of a voter who votes for candidate A (the system relies on the numeric value 1 to represent each candidate). Rather than save or transmit the value "1" (thus risking the privacy of the vote), the system uses a scheme based on Shamir's Secret Sharing (SSS). In this case, the vote is split into shares and is done so with a (3-of-5) threshold scheme, meaning that the vote will sit on five shares, and three shares will be required to reconstruct the vote.

As shown in the diagram, the first step is Vote Splitting. In this phase, the value "1" gets passed through the SSS algorithm to produce 5 random looking shares (for example, Share 1, Share 2, Share 3, Share 4, etc.). These shares disclose nothing about the original vote once they are separated, and from here, the vote is secure (even if someone received one or two shares).

The second phase is Share Distribution. Each of the shares will be sent to each of the 5 servers (i.e. Server 1 will receive Share 1, etc.). Each server would be running on a different and separate machine in the decentralized network. This distribution means no one server has enough information to reconstruct the vote by itself, thus eliminating any potential single point of failure or insider threat.

Finally, in the Reconstruction phase of the tallying process, any three or any five of the servers can collaborate and each one can provide its share. The three shares can be combined mathematically to yield the original vote value (in this case, 1) using the reconstruction algorithm of Shamir's scheme. Notice the

reconstruction is done only during the aggregation phase and the individual choices of the voters become private as the tally is being measured.

The mechanism illustrates how security and fault tolerance and privacy can happen simultaneously. Even if two of the servers are offline or have been compromised, the system can still reconstruct the vote value correctly, and at a robust capacity in a real world implementation.

Security Guarantees

The proposed vote counting system provides a robust set of guarantees that address the key concerns in secure and trustworthy digital elections. These guarantees are achieved through the integration of **Shamir's Secret Sharing (SSS)**, **Multi-Party Computation (MPC)**, and cryptographic reconstruction via **Lagrange interpolation**.

- **Privacy**:
  One of the most important necessities of any voting system is the assurance of individual voter privacy. In this case, individual votes are never stored, transmitted, or exposed in their raw form. Each vote is separated into shares using Shamir's Secret Sharing (SSS) and each share, by itself, is of no value. If there are a number shares less than the threshold, the vote will remain completely private. This fulfills the golden rule of secret ballot.

- **Tamper Resistance**:

- The system is designed to automatically detect and defend against interfering with or corrupting vote shares. Each vote total of one's contribution is created via a mathematical representation of each contribution. Thus, we would consider even the slightest form of tampering by falsifying just one share, by corrupting one share, or otherwise by invalidating just one share, you will disrupt the resources mannerisms as you won't be able to reconstruct what you attempted manipulate-and everything would not make sense. Therefore, tampering is also detected easily and the system is self-defending against these manipulations.

- **Fault Tolerance**:

- With a threshold-based reconstruction model, the system can keep performing when at most $n - t$ nodes are offline or otherwise inoperative. This is significant for practical implementation, as it is conceivable that, due to some network problem or system failure, a subset of parties could be

unavailable at the time of tallying. The result will be able to be determined correctly as long as the threshold t is met.

- **Integrity**:
  The final tally is mathematically guaranteed to be correct as long as a sufficient number of honest shares are used. This integrity stems from the deterministic nature of polynomial interpolation, ensuring that even in the presence of partial failures or limited participation, the vote count remains accurate, verifiable, and tamper-proof.

Advantages of MPC in E-Voting

The integration of Multi-Party Computation (MPC) into electronic voting systems offers several significant advantages that enhance the overall **security, privacy, and trustworthiness** of the election process. This decentralized computational model distributes sensitive operations across multiple independent entities, mitigating the risks commonly associated with centralized systems.

1. Decentralization:
One of the core strengths of MPC lies in its inherently decentralized nature. In traditional e-voting systems, a central authority often holds full access to all vote data, making it a critical point of failure. With MPC, the computation is distributed across multiple independent nodes, where each node only holds a fragment (or share) of the data. This architecture ensures that no single entity can reconstruct or manipulate the vote independently, thereby reducing the risks of insider threats, corruption, or unauthorized access.

2. Privacy Preservation:
MPC guarantees end-to-end privacy of voter data. Each vote is mathematically split into shares and processed in a distributed fashion without ever reconstructing the full vote during intermediate stages. This means that individual voting choices remain concealed throughout the computation and tallying phases. Even the participating servers, acting in good faith or otherwise, cannot learn anything about the original vote unless they collaborate in sufficient numbers to meet the predefined threshold. This strict privacy model supports voter confidence and the secrecy of the ballot.

3. Enhanced Security and Fault Tolerance:
In MPC-enabled systems, even if some parties are compromised—either due to external attacks or internal failures—the protocol remains secure as long as the number of compromised nodes is below the threshold required to reconstruct the data. This resilience ensures system integrity, data accuracy, and

robustness against denial-of-service attacks or partial system outages, thereby making MPC a reliable and secure choice for modern e-voting solutions.

Performance Considerations

While Multi-Party Computation (MPC) provides a high level of security and privacy guarantees, MPC-based approaches are nevertheless inherently slower in nature due to the time-consuming step of cryptographic operations, as well as all of the communication between multiple parties. That said, the designed prototype offers a solution that allows for practical use in a real world environment, aspects of our design process were aimed achieving better performance levels.

For example, implementing parallel share processing, in which each server (or MPC node) performs its assigned compution independently and at the same time, is a key performance optimization. This reduced the potential latency for both the vote casting and vote tallying aspects of the system and since each server is only doing a part of the voting tallying (turning the votes into random shares), the high number of votes to process concurrently do not impact the security of the unique shares.

The system also relies on pre-distribution of voting shares. Pre-distribution refers to the method in which an infrastructure has been partially prepared before voting; random pieces or pre-generated shares have been distributed ahead of time. Pre-distribution of voting shares serves the usefulness for actually the voters pre-protocol and allows them to split and distribute with knowing delay when they eventually vote.

However, there are also more advances that can be painless for improve the tradeoffs between with respect to efficiency and crypto-secureness. For example, the current computing could be combined with light weight cryptographic primitives, like with elliptical based commitments or fast hash based zero knowledge proofs. Finally, MPC can also be used with homomorphic encryption to provide a superior hybrid method that computes on your shares whilst never reconstituting until the final aggregation.

Developments like these would extend the software's strategy in respect scalability while remaining true to its original foundations of privacy, integrity, and robustness for various voting applications even in restricted environments both massive and challenged.

Implementation & Testing Vote Counting

Implementation Summary

The implementation of the secure vote counting system is structured into several key modules, each responsible for a critical part of the overall process:

- **generate_shamir_shares(secret, num_parties, threshold)**:

  This module is responsible for **splitting each vote into n cryptographic shares** using Shamir's Secret Sharing. It generates a random polynomial with the vote as the constant term and evaluates it at distinct points to create shares, which are then distributed to multiple vote counting parties.

```python
def generate_shamir_shares(secret, num_parties, threshold):
    coefficients = [gmpy2.mpz(random.randint(1, MODULO - 1)) for _ in range(threshold - 1)]
    coefficients.insert(0, secret)
    return [(gmpy2.mpz(i), sum(coefficients[j] * (i**j) for j in range(threshold)) % MODULO)
            for i in range(1, num_parties + 1)]
```

Fig. 4. Shamir Secret Sharing Code Snippet

- **MPCVoteCounter**:

  This class is designed to **organize and store vote shares**, mapping them by both party and candidate. It maintains a secure internal structure to track which shares belong to which candidate across the distributed nodes.

- **count_votes()**:

  This function **initiates the reconstruction process** by collecting shares for each candidate and checking whether the required threshold of shares is met.

```python
def count_votes(self):
    print("\nFinal Vote Count:")
    final_counts = defaultdict(int)

    for candidate in self.candidates:
        collected_shares = []

        for i in range(NUM_PARTIES):
            for vote_share in self.party_shares[i][candidate]:
                collected_shares.append(vote_share)

        if len(collected_shares) >= THRESHOLD:
            try:
                final_counts[candidate] = reconstruct_shamir_secret(collected_shares)
            except Exception as e:
                print(f"⚠ Error reconstructing votes for {candidate}: {e}")
        else:
            print(f"Not enough shares to reconstruct votes for {candidate}")

    for candidate, count in final_counts.items():
        print(f"  {candidate}: {count} votes")
    return final_counts
```

Fig. 5. Count vote function

- **lagrange_interpolation()**:

  It performs the **final vote reconstruction** using threshold shares through polynomial interpolation, producing the correct total without revealing individual votes.

```python
def lagrange_interpolation(x, x_values, y_values, prime):
    total = gmpy2.mpz(0)
    for i in range(len(x_values)):
        num, denom = gmpy2.mpz(1), gmpy2.mpz(1)
        for j in range(len(x_values)):
            if i != j:
                num = (num * (x - x_values[j])) % prime
                denom = (denom * (x_values[i] - x_values[j])) % prime

        if denom % prime == 0 or gmpy2.gcd(denom, prime) != 1:
            raise ValueError(f"Invalid denominator {denom}. Cannot compute modular inverse.")

        lagrange_coeff = (num * gmpy2.invert(denom, prime)) % prime
        total = (total + (y_values[i] * lagrange_coeff)) % prime
    return total
```

Fig. 6. Lagrange Interpolation Code snippet

Technologies Used

To realize the goals outlined in the system design and methodology, the implementation of the proposed **MPC-based voting system** integrates a carefully selected stack of programming languages, libraries, and frameworks. These technologies were chosen for their reliability, extensibility, and suitability for secure, distributed vote tallying, cryptographic computations, and real-time data processing. The selected stack supports **privacy-preserving vote counting**, **decentralized trust**, and **collusion resistance**, making it ideal for creating a transparent, tamper-resistant voting platform.

Python 3.x

Python serves as the primary language for implementing the backend logic of the MPC vote counting system. Its simplicity, readability, and extensive library support make it ideal for rapid development, cryptographic computations, and system experimentation.

- **Used for:** Logic handling, cryptographic computations (e.g., Shamir's Secret Sharing, Lagrange interpolation), parallel processing of vote blocks, and handling input/output.

- **Why chose:** Python's clean syntax, dynamic typing, and powerful libraries like gmpy2 and concurrent.futures make it an efficient and versatile choice for cryptographic tasks and backend computation.

gmpy2

gmpy2 is a Python library used for fast arithmetic and rational number computations, especially for handling large numbers in cryptographic applications.

- **Used for:** Efficient handling of large integers and modular arithmetic in Shamir's Secret Sharing and Lagrange interpolation.
- **Why chose:** It provides high-performance support for arithmetic operations over large integers, which is crucial for cryptographic calculations such as modular arithmetic and interpolation in vote tallying.

Blockchain (Custom Blockchain Class)

Blockchain technology is used to securely store and verify votes via an immutable ledger. This guarantees the transparency and tamper resistance of the voting process.

- **Used for:** Storing and validating vote records, ensuring that the integrity of the voting process is maintained.
- **Why chose:** Blockchain offers decentralized, tamper-resistant storage that can be used to verify the authenticity of votes and support audit trails.

zk-SNARK (Zero-Knowledge Succinct Non-Interactive Argument of Knowledge)

zk-SNARKs allow a party to prove possession of knowledge (e.g., a valid vote) without revealing the underlying data.

- **Used for:** Verifying voter eligibility without disclosing their identity or vote choice.
- **Why chose:** It ensures voter privacy while maintaining the integrity of the voting process, allowing the system to confirm the validity of votes while keeping sensitive information confidential.

ThreadPoolExecutor (from concurrent.futures)

ThreadPoolExecutor is used to handle multi-threaded execution, enabling concurrent processing of vote blocks retrieved from the blockchain.

- **Used for:** Parallel processing of multiple vote blocks to improve the performance and efficiency of vote retrieval and verification.
- **Why chose:** It optimizes performance by allowing concurrent tasks to be handled efficiently, speeding up the processing of blockchain data and ensuring real-time updates.

```
with ThreadPoolExecutor() as executor:
    valid_votes = list(executor.map(process_vote, blockchain.get_chain()))
```

Fig. 7. ThreadPoolExecutor Function

Shamir's Secret Sharing (SSS)

SSS is used to split each vote into multiple cryptographic shares, distributed across different tallying parties. Only a minimum threshold number of shares is required to reconstruct the original vote.

- **Used for:** Dividing votes into shares to ensure that no single party can reconstruct or alter the vote independently.
- **Why chose:** It ensures perfect secrecy and collusion resistance, as no single party can manipulate the votes or learn the individual vote details.

Multi-Party Computation (MPC)

MPC enables secure computation over distributed data, allowing multiple parties to collaboratively compute a result without exposing individual inputs.

- **Used for:** Securely summing vote shares from different parties without exposing individual votes or vote counts.
- **Why chose:** It ensures that the final vote tally is computed in a distributed manner, preserving privacy and integrity by involving multiple parties in the computation.

Lagrange Interpolation

Lagrange interpolation is used to reconstruct the original vote total from the distributed shares of votes.

- **Used for:** Reconstructing the final vote count after collecting sufficient shares from multiple parties.
- **Why chose:** It provides a mathematically sound method for reconstructing the original secret (vote tally) while maintaining security and privacy.

Scalability with Multiple Devices or Machines

The system is designed to scale the number of vote counting parties across multiple devices or machines. Each device or machine can function as an independent party node that stores and processes shares of the votes. This distributed architecture enhances the system's ability to handle large-scale elections while maintaining the decentralization and security of the process.

- **Used for:** Distributing the vote tallying process across multiple devices or machines, improving scalability, fault tolerance, and redundancy.
- **Why chose:** This distributed setup allows the system to scale easily for larger elections, leveraging the resources of multiple machines to ensure high availability and performance.

Practical Implementation of MPC in a Local Environment

The system implementing secure vote counting with confidentiality uses a multi-party computation (MPC) protocol, known as Shamir’s Secret Sharing. In this proposed, each vote undergoes digitization first and then undergoes a mathematical process of splitting into multiple shares. The shares are distributed to different independent parties, meaning that the original vote cannot be reconstructed by a single party without combining shares with independent parties.

MPC supports the proposed voting system aimed at maintaining privacy and trust regarding the vote counting process. Each single vote goes through a process of encoding using the Shamir's Secret Share scheme and, mathematically splits into multiple shares. Each of the shares will then be given to three different computing parties, which are hosted on three different physical or virtual machines in a local network environment. The parties are separated so that each one is not re-constructed solely by one machine so that the original vote cannot be reconstructed, nor inferred about what the voter might have preferred.

For this project, we configured three different independent machines to individually act as MPC nodes. Upon casting a vote, the system creates shares of the vote and forwards these shares to each machine. The original vote cannot be reconstructed until the machines that take part in reconstruction reach a certain threshold (for example, in our case 2 out of 3 machines reconstruct the vote). Although this is a 'local' setup, the decentralized nature is maintained and allows us to evaluate the workability of MPC for real-world appeal in a secure and fault-tolerant manner. In the vote tallies, only the share of votes are being processed and therefore the individual vote is never reconstructed in full, maintaining end-to-end voter anonymity.

This empirical configuration not only enables the theoretical design of secure and private vote counting, but illustrates the potential of deploying MPC based systems in the real-world apparatus with a decentralized computing infrastructure.

Functional Testing

To validate the functionality and robustness of the proposed vote counting system, a set of test cases was conducted using **20 randomly cast votes** for two candidates: **Candidate A** and **Candidate B**. The votes were processed through the system's complete workflow, including secret sharing using Shamir's Secret Sharing (SSS), secure storage, and threshold-based reconstruction using Lagrange interpolation.

- **Test Case 1: All Parties Online**

  In this scenario, all **3 vote counting parties** are active and responsive. The system successfully reconstructs the **accurate vote counts** for both candidates. Since all shares are available, the threshold condition (**2-of-3**) is easily met, demonstrating full functionality under ideal conditions.

- **Test Case 2: One Party Offline**

  This test simulates a **partial failure** where only **2 out of 3 parties** are online. The system still **accurately reconstructs the vote totals**, confirming the system's **fault tolerance**. The 2-of-3 threshold ensures that even with one node offline, the tallying process can proceed securely.

- **Test Case 3: Less Than Threshold (Only 1 Party Online)**

  In this failure case, just **1 party is available**, which is **below the minimum threshold**. As expected, the system fails to reconstruct the vote count and returns a **warning message**. This behavior confirms that **vote privacy remains intact** and that no reconstruction can occur without sufficient shares.

These tests demonstrate that the system is both **resilient and privacy-preserving**, performing reliably even when some parties are unavailable.

Sample Output

During the final testing of the system, the zero-knowledge proof (zk-SNARK) verification for voter V001"" resulted in a successful outcome: True. This simply means that the voter was registered and had not voted already, with zero knowledge disclosed about any personal or vote related information. After processing all votes and reconstructing the final vote count using Multi-Party Computation (MPC) and Shamir's Secret Sharing, the system produced a valid and privacy preserving output. The final vote count showed Candidate A received 12 votes, while Candidate B received 8 votes, verifying that the system correctly handled vote splitting, secure distribution, and threshold-based reconstruction.

Edge Case Handling

The system has a number of measures to protect the accurate reconstruction of the vote. In order to reduce the risk of the same party submitting multiple shares on the same vote, each share is assigned a party-specific identifier, so there are no duplicate shares. Consequently, a would-be adversary cannot inflate a vote or replay votes at tallying. If a participant submits a tampered share or multiple shares are written with the same message, the error will be detected during the Lagrange interpolation phase. Any that do not pass validation or that contain invalid values will prevent the reconstruction of the vote from succeeding. Whatever the vote count is, will still remain valid. When there are not enough shares (effectively below threshold) the system will display a warning message if there are not inferring the vote of the candidate in question, while maintaining the integrity of their vote.

Performance Testing

The findings of these experiments speak to the performance and scalability of the system. The average generation time of each vote share was approximately 0.03 seconds which was a highly efficient time frame when considering the potentially large number of voters. The reconstruction stage had an actual total vote tally of only 0.1 seconds for the 100 votes using Lagrange interpolation. Again, this suggests the system's speed is both practical and effective. For the extreme robustness of the system, we stress tested the system to include 10,000 vote shares, yet the system experienced only small variances in measure reliability and accuracy. So, we define our system architecture which is based on Shamir's Secret Sharing and MPC as both secure/private, performant and scalable for real world pre-designed use cases.

Commercialization Aspects: MPC-Based Counting

Market Gap

Currently, most digital voting systems either depend on centralized tallying systems or are based on homomorphic encryption to allow votes to be computed while being encrypted. Centralized and cryptographic homomorphic systems can provide some advantages, but they do not address the full range of problems related to trust, privacy, and transparency. Centralized systems have the disadvantage of being a single point of trust and thus, single point of failure that insiders can exploit and compromise. Homomorphic systems allow for computation while votes are encrypted and have privacy advantages, but due to their complex coded solutions, they often require higher levels of computation than current technology permits, and do not provide built-in distributed trust.

This system proposes a threshold based, distributed vote counting framework based on Shamir's Secret Sharing (SSS) and Multi Party Computation (MPC). The proposed distributed framework removes the need for any single authority which has to trust every person involved in the voting process and allows a range of people to work together with their votes and democratic decision to be counted. No vote is ever revealed until it is counted, where not only each individual vote is private, is counts all the valid votes we had and can tell if any invalid votes or individuals tried to manipulate the results but in addition, our hybrid blockchain/cryptographic technology promises an audit trail of immutable records. Third parties can confirm their presence at any stage of the voting process without infringing on individual rights to privacy.

Deployment Model

The proposed system can be modular and adaptable to suit any digital voting mechanisms. It can be provided purely as a vote counting API or as a back-end engine that would conduct secure, privacy-preserving tallying of those votes using threshold cryptography. The architecture would allow for "plug and play" compatibility with all e-voting front ends, whether web or local mobile, used for the voter registration, ballot casting, or display of election results.

The system is capable of supporting a distributed network of vote counting parties, with the exact configuration agreed to reflect the stakeholders for an election. Those parties could be electoral commission nodes, international observers, third-party auditors or other neutral verifiers, who are each provided with a cryptographic share of the votes and able to process the vote counts. Using this distributed model will convey transparency and decentralization of trust for the verification of the integrity of the vote counting,

and can do so in situations where the elections are large-scale or politically delicate. The configuration flexibility also allows for various types of local, national or organizational voting contexts.

Competitive Advantages

The proposed voting counting solution is designed to be light-weight, verifiable and transparent making it well suited for incorporation into existing and next generation e-voting systems. Traditionally, vote counting mechanisms have employed heavy cryptographic paradigms (cryptographic homomorphic encryption or extensive and complex zero-knowledge proof circuits on the tallying stage). However, the proposed system will employ techniques that are computationally efficient and highly scalable, Shamir's Secret Sharing (SSS) and Multi-Party Computation (MPC). In this respect performance overhead and impact is substantially low and security and privacy remain high.

One of the key features of the proposed system is verifiability. The proposed system records all vote associated actions on an immutable blockchain ledger, and also employs threshold-based share reconstruction; this makes the entire vote counting process audit-able and tamper-evident. This helps ensure that stakeholders (voter, election observer, third-party auditor, etc) will be able to make use of the verbility explicitly enabling them to verify the election outcomes without directly revealing and compromising vote secrecy.

Additionally, the proposed voting counting system promotes transparency in tally logic. Unlike the black-box tallying mechanisms that many existing e-voting systems employ, this mechanized architecture makes it clear what modules are active in vote validation, sharing, storing, and reconstructing votes. Each module (i.e. share generation, distribution, interpolation, etc) can be individually tested, and/or described, remaining entirely assessable; adding to the system transparency.

An outstanding feature of the system is that it is flexible and modular in shape. It can function as its own vote counting engine for petty elections (e.g., university election or nearby organization polls), or it can be integrated as a plugin into larger, more sophisticated voting environments, including e-voting services on a national level. By being harmonizingly coded with web application programming interfaces (APIs) and standard interface implementations, it can be readily connected to relevant front-end voter interfaces, biometric systems, or any existing digital ID systems. This modularity provides components for many different use cases, from small local elections to national votes with a high security requirement.

Results and Discussion

Results

The proposed secure vote counting system was implemented in **Python** and tested under a controlled simulation involving **20 randomly cast votes** for two candidates: **Candidate A** and **Candidate B**. The system utilized a **2-of-3 threshold scheme**, meaning any **two out of three vote counting parties** were sufficient to successfully reconstruct the vote tally using **Shamir's Secret Sharing** and **Multi-Party Computation (MPC)**.

Test Environment

- **Hardware**: Intel Core i7 Processor, 16GB RAM
- **Software Stack**:
    - **Language**: Python 3.10
    - **Libraries**:
        - gmpy2 for modular arithmetic and prime field operations
        - concurrent.futures for parallel vote processing
        - Custom Blockchain module for immutable vote storage
        - Custom ZKP module for zk-SNARK-based voter proof verification

Key Test Scenarios and Outcomes

A range of scenarios were executed to test the resilience, reliability, and privacy-preserving nature of the system. The table below summarizes the outcomes:

Test Case	Setup	Outcome
All 3 Parties Online	20 votes cast, all nodes active	Accurate reconstruction
1 Party Offline	Only 2 nodes online, threshold met (2-of-3)	Accurate reconstruction
Only 1 Party Online	Threshold not met	Reconstruction failed, warning

These scenarios confirmed that the system correctly handles both ideal and degraded operating conditions. Even with one party offline, the system was able to reconstruct the vote tally correctly, showcasing its **fault-**

tolerance and adherence to threshold logic. In the failure case with only one party online, the system issued a **warning and skipped reconstruction**, preserving **vote secrecy**.

Performance Metrics

In terms of performance, the system achieved promising results:

- **Vote Share Generation Time**: Approximately **0.03 seconds per vote**
- **Vote Reconstruction Time**: Approximately **0.1 seconds for 100 votes**
- **Scalability**: Stress-tested with **up to 500 votes**, maintaining consistent accuracy and responsiveness

These metrics indicate that the system is computationally **lightweight** and capable of supporting small to medium-scale elections with ease. The modular structure and efficient use of cryptographic operations allow for real-time processing without significant delays or resource overhead.

Final Tally Output (20 Votes)

A sample simulation yielded the following secure and verifiable results:

Proof verification result for voter V001: True

Final Tally Using MPC:

Candidate A: 12 votes

Candidate B: 8 votes

This result was achieved using MPC reconstruction across the active nodes. The proof verification step confirmed voter eligibility and uniqueness via zk-SNARKs, while the vote counting process ensured privacy, tamper-resistance, and correctness without exposing individual votes.

Research Findings

The implementation and testing of the proposed vote counting system produced several significant findings, aligning with the research objectives and confirming the theoretical strengths of the combined use of **Shamir's Secret Sharing (SSS)**, **Multi-Party Computation (MPC)**, and **zk-SNARKs**, underpinned by **blockchain technology**. These findings highlight the system's effectiveness, reliability, and real-world applicability.

1. **Perfect Vote Privacy**

   The system ensured that individual votes remained completely private throughout the process. By using SSS, each vote was split into multiple cryptographic shares, none of which revealed any information independently. Reconstruction was only possible when the threshold number of shares (e.g., 2 out of 3) was met, guaranteeing that no single party could determine how a voter cast their vote.

2. **Tamper Resistance**

   The use of polynomial-based interpolation in reconstruction inherently provided tamper detection. Any modified or corrupted share disrupted the interpolation process, resulting in an invalid or failed reconstruction. This built-in mechanism served as an early warning system for manipulation attempts or compromised nodes.

3. **Fault Tolerance**

   The system remained functional even when up to **n - t** nodes were offline or unresponsive. In the test configuration (2-of-3 threshold), the system successfully reconstructed vote totals even with one node offline. This demonstrates that the architecture is resilient and reliable under partial system failure.

4. **Auditability**

   All key events—such as proof verification, vote casting, and share generation—were immutably recorded on the blockchain. This ensured complete transparency and allowed independent auditors or stakeholders to verify the integrity of the election process without accessing sensitive data.

5. **Efficiency**

   Compared to homomorphic encryption-based systems, this model proved far more efficient in terms of computation time and resource usage. It is scalable and suitable for deployment in real-time or constrained environments without sacrificing security or accuracy.

Discussion

The proposed system has satisfied the critical components of a secure, trustworthy, and privacy-preserving electronic voting procedure through advanced cryptographic and distributed technologies. At its core, the system implements Multi-Party Computation (MPC) alongside Shamir's Secret Sharing (SSS), ensuring that vote data is never revealed at any one time in its entirety to one single point in the infrastructure. Each vote is divided into multiple shares and then distributed between independent servers hosted on different machines so even if a segment of the server is compromised the vote will remain confidential and integrity preserved due to the shared secret nature of the secret sharing scheme.

To improve upon the privacy and authenticity of voters, we have embedded zk-SNARKs (Zero-Knowledge Succinct Non-Interactive Arguments of Knowledge) for voter authentication. Voters can authenticate their eligibility as voters without disclosing any identifiable or sensitive information that would allow someone to impersonate them, vote twice, or access the voter identification privilege. By removing the need to disclose voter identity at the time of authentication, we maintain the voters anonymity while preserving the integrity of the vote.

Furthermore, we have employed blockchain technology to provide immutability and auditability of key events occurring while voting. Each transaction such as casting a vote, uploading a proof of verification, collating the data to a final count will be recorded into a secure, increasing, prevent- tampering, chronological ledger. This provides auditable verification of any and all actions taken within the system without compromising the voters' privacy to an outside auditor or stakeholder. By utilizing the blockchain's decentralized structure, we have eliminated single points of failure, added transparency, and increased voter trust and administrator credibility.

By using these powerful techniques: MPC to enable distributed privacy-preserving calculation; SSS to provide symmetric secure fragmentation of votes; zk-SNARKs to provide zero-knowledge, identity-preserved voter authentication; and blockchain to provide a record of history; we advance a more complete, scalable, and secure e-voting capability. It mitigates many of the critical vulnerabilities observed in traditional centralized electronic voting systems and offers a **practical blueprint for real-world electoral deployments**, especially in environments where trust, privacy, and accuracy are paramount.

Decentralized Trust Model

One of the most exciting features of the proposed e-voting solution is the decentralized trust model where digital trust is redefined. Most electronic voting systems all rely on a centralized system where a single entity or institution is responsible for vote data storage, authenticating voters, and tallying the data. While this mechanism is effective in terms of implementation, it relies on a single point of failure. If the process is corrupted through a hack from the outside, or manipulated from the inside, even one compromised vote can alter the integrity of the election and cause data tampering, vote misrepresentation, denial of access, or vote manipulation.

The proposed solution will resolve this serious flaw found at the core of electronic voting systems. By decentralizing the computational load of an election amongst multiple independent entities, we remove the potential for compromise. Using Multi-Party Computation (MPC) and Shamir's Secret Sharing Scheme (SSS), every vote is split into different splits (shares) and passed off to different MPC nodes running on different systems. No one voting node sees a single complete vote, and only a certain threshold of voting nodes, collaborating together, can report out the aggregated result of a vote. There is no way for a single party to observe, corrupt, or reconstruct a vote.

Further, this decentralized structure enables transparency and verifiability, as each party is able to independently verify the participation of each party without relying on a trusted third party. This model encourages greater levels of trust from voters, election monitors, and others, especially in politically sensitive situations or high-stakes environments. The fact that one cannot potentially depend on a single trusted component also makes our process more resilient to manipulations, censorship, or technical problems, creating greater trust in the integrity of our democratic process via collaborative trust and distributed assurance.

Efficiency vs. Cryptographic Complexity

Privacy-preserving voting systems are designed to keep voters anonymous while maintaining integrity within modern democracy. One of the most widely researched encryption methods for privacy-preserving voting systems is homomorphic encryption. Homomorphic encryption lets users perform computations on encrypted data directly without decrypting the data first. Although homomorphic encryption methods provide solid theoretical guarantees for confidentiality and correctness, they have a notable downside: they are computationally expensive. Fully homomorphic encryption schemes have much higher computational costs than traditional forms of encryption, which can lead to performance bottlenecks associated with large-

scale elections or being utilized in environments with potentially weak computational power and limited bandwidth.

Conversely, our proposed system uses a lightweight, less computationally expensive solution to tallying encrypted votes - Shamir's Secret Sharing can be combined with Multi-Party Computation (MPC). Using MPC shared voting allows for secure vote tallying without needing to decrypt the individual voters or perform the heavy mathematical computations seen in homomorphic methods. Rather than relying on a center server, or heavy cryptographic computations, our method provides a method to simply split each vote into a number of shares distributed across multiple independent MPC nodes that will work together to compute the outcome of the election while preserving privacy from others, with no need to decrypt each vote, as well as a threshold-based reconstruction of the necessary parts of the votes will work to share computable parts or components needed to compute the election outcome.

This method is especially useful in developing or rural contexts with limited compute infrastructure. It eliminates expensive computing infrastructure and does not include the significant latency of homomorphic encryption. By balancing security and usability, MPC using secret sharing can provide secure, scalable, cost-effective privacy-preserving elections — making it feasible for use cases where privacy, as well as efficiency, is a primary concern.

zk-SNARK Integration

Incorporating zk-SNARKs (Zero-Knowledge Succinct Non-Interactive Arguments of Knowledge) is key for privacy and securing the integrity of voting process in the proposed electronic voting process. One of the prime challenges with secure electronic voting is ensuring a voter is eligible and one-of-a-kind - without knowing the identity of the voter. Most authentication protocols use PII, or personal identifiable information, to authenticate a voter, which, depending on the type of information, brings potential privacy risks and in some instances can act as a barrier to voter registration and participation.

With zk-SNARKs, an eligible voter can cryptographically prove that they are a registered voter, and more importantly, that they have not voted yet - without the ability to identify any personal or identifiable information about the voter. The zk-SNARK verification process can take minimal time (can be verified in milliseconds) and is zero-knowledge and non-interactive. Once the voter creates a zk-SNARK proof and submits it with the vote to the system, the verifier module will validate the zk-SNARK proof before accepting it as a vote.

This mechanism ensures three critical properties:

1. **Anonymity** – No private information is revealed during voter verification.
2. **Uniqueness** – zk-SNARKs ensure that each voter can vote only once.
3. **Integrity** – The verification process maintains the correctness of the voter registry and prevents tampering.

By combining zk-SNARKs with the broader system components—such as Multi-Party Computation and blockchain logging—the proposed e-voting system achieves a high level of security and privacy. This integration effectively bridges the gap between voter transparency and confidentiality, making the system suitable for real-world elections where both trust and privacy are paramount.

Threshold Configuration Trade-offs

The system's current threshold configuration of 2-of-3 was determined to be an acceptable balance between performance, availability, and fault tolerance during tests. In a 2-of-3 configuration, each vote is split into three shares—any two of those can reconstruct the original vote. In 2-of-3, if one of the MPC nodes becomes unavailable, for whatever reason—even if it is due to a network issue, system failure, or malicious actor—the system would still be able to continue tallying votes without issue. Therefore, the 2-of-3 configuration offers high responsiveness and tolerance and is especially applicable in low-scale deployments or deployments relying on unreliable infrastructure.

But, this low threshold comes with trade-offs in security and collusion resistance. Specifically, because only two of three nodes are needed to reconstruct the secret, it becomes easier for a set of compromised nodes (i.e., colluders) to conspire together to leak or tamper with sensitive vote data. Conversely, increasing the threshold to a higher configuration (for example, 5 of 7) would greatly increase collusion resistance. In such a setup, at least five out of seven independent nodes must collaborate to access or process vote data, making unauthorized access or vote tampering far more difficult—even in the presence of insider threats.

Thus, threshold selection is context dependent. Systems whose operation primarily depends upon performance, availability, and real-time processing may allow for lower thresholds. Alternatively, systems whose application relies upon the assurance of maximum privacy and resistance to tampering such as national-level elections may necessitate higher thresholds. This choice should take into careful consideration the reliability of the network, the trust level of computation nodes, and the sensitivity of the

voting approach. Ultimately, the flexibility of threshold-based secret sharing allows for system designers to reject a security-performance compromise depending on the situation of each election environment.

Real-World Applicability

The e-voting system is constructed to be modular and flexible enough to support a wide range of elections and polling styles. The architecture is strongly rooted in the principles of security, privacy, and decentralization. As such, it is suitable for public electoral settings such as the government elections, but is also suitable for smaller democratic voting mechanisms occurring through organizations and institutions.

Some of the primary use cases for the e-voting system include government elections, both local and on the national level. The e-voting system offers necessary transparency, trust, and scalability required to facilitate public electoral processes through the application of Multi-Party Computation (MPC) as well as Shamir's Secrets Sharing (SSS), and zk-SNARK-based voter verification and using an blockchain-based audit trail. The e-voting system enables end-to-end verifiability while allowing democratic governance standards of rigor while maintaining the secrecy of voters and resisting democratic manipulation.

In the corporate or organizational context, the use cases might also include board member elections, policy decisions, or shareholder votes where confidentiality and avoiding conflicts of interests are hugely relevant. The modular flexibility of the e-voting system is ideal for customizing what the rules for eligibility of voters based on the governance framework of each organization, as well as customization of recognition criteria for quorum and validating results. The system is also applicable to university elections such as student union elections or selections for faculty councils, where integrity is essential, and ease of access is crucial. The platform can be deployed online, allowing for secure, remote voting, encouraging students and faculty who may not be physically present on campus to participate.

For public opinion collection including online referenda or policy consultations, the platform allows governments, municipalities, or agencies to digitally poll citizens and be secure in the knowledge that their voting results cannot be tampered with. Thanks to zk-SNARK technology, participants remain anonymous while blockchain records serve as transparency, offering public assurances of participation.

The system can also be deployed as a stand-alone application or a plug-in/API backend integrated with existing e-voting infrastructure, offering valuable functionality to modernize existing legacy voting systems without wholesale replacement. The architecture offers room for a neutral third party such as an election observer or an international monitor to join the system as one or more MPC nodes or vote verifier, which

can be invaluable in a politically-sensitive context. In such politically sensitive contexts public confidence and acceptance of results are very much tied to having outside validation.

Conclusion

The proposed MPC-based voting system is a tremendous step towards the construction of a secure,30 decentralized, and privacy-preserving election platform, while overcoming the prime limitations of traditional electronic voting systems. The system, utilizing the paramount properties of Multi-Party Computation (MPC), Shamir's Secret Sharing (SSS), zk-SNARKs, and blockchain technology, obtains the principal aims of voter anonymity, trustlessness, fault tolerance, and auditability with improved performance and scaling.

A huge advantage of the system is its decentralized trust model. Traditional centralized systems rely on a single source of authority to tally votes, creating a system susceptible to failure and worse yet manipulation, the proposed system distributes authority across several parties. Each party only processes a portion of the vote data, and the official result can be reconstructed when an acceptable threshold of parties has agreed to collaborate on reassembly. This threshold rids the possibility of internal manipulation while improving the trust in the robustness of a probabilistic approach without collusion and enhanced robustness within a limit of potential party failures.

The system further provides privacy-preserving vote tallies using Shamir's Secret Sharing, guaranteeing that no parties will ever obtain full vote content. The vote is separated into shares cryptographically, and the shares can be reconstructed only if a sufficient number of parties share their shares. Once again, this is a design that prevents any unauthorized individual from tampering with the votes; even during the computation phase, the individual votes remain private. The adoption of zk-SNARKs in the system provides a security measure for verifying unique and eligible voters, so that only eligible and unique voters are voting, while maintaining their anonymity as voters during the voting process. The combination of blockchain technology for tracking events related to voting will also ensure transparency and auditability in the entire voting process. The blockchain provides an immutable, tamper-resistant log of verifiable information about everything related to the votes, from vote casting, proof verification, and distribution of the shares. This will allow any independent audit of the process while maintaining the privacy of the vote. Overall, the transparency of the voting process will foster trust in the election process and improve verifiability and public accountably for all stakeholders including election monitors, voters and the public.

Another feature of our system is its efficiency and scalability. In contrast, systems based on homomorphic encryption could be expensive and inefficient approaches. The combination of Shamir's Secret Sharing and MPC allows for a simple, efficient implementation designed for realtime, or larger scale elections. Performance evaluation highlighted our system as adequate for processing and tallying votes quickly even through large quantities of votes. This led to large variety of potential elections for use cases ranging from local elections to national and international electors.

Although we have many strengths mentioned above, we do acknowledge the potential trade-off of our threshold configuration in terms of security versus fault tolerance. While the 2-of-3 threshold configuration has good fault-tolerance, it will be less secure against collusion. Future work could involve the potential security improvements of using higher thresholds such as 5-of-7 using the trade-off of performance, and fault-tolerance.

Ultimately, this study has shown that it is possible and practical to utilize MPC, SSS, zk-SNARKs, and blockchain to create a secure, decentralized, and transparent voting system. This approach to voting overcomes the limitations of both centralized systems and homomorphic encryption-based voting systems, and presents a highly secure and scalable method of conducting elections in the modern day - and as technology continues to advance, it is critical to ensure our elections remain fair, transparent, and trustworthy. As a modular system, it can easily integrate with existing platforms, and can adapt to a variety of electoral scenarios.

References

[1] I. o. O. V. R. Insecure, "AAAS.ORG," [Online]. Available: https://www.aaas.org/epi-center/internet-online-voting. [Accessed 9 4 2025].

[2] L. H. Newman, "Wired.com," [Online]. Available: https://www.wired.com/story/voting-village-results-hacking-decade-old-bugs/.

[3] E. Lab, "Voting Technology," [Online]. Available: https://electionlab.mit.edu/research/voting-technology.

[4] W. Z. C. Z. Z. Z. Yu Zhan, "Efficient Electronic Voting System Based on Homomorphic Encryption," *Efficient Electronic Voting System Based on Homomorphic Encryption.*

[5] AAAS, "Internet or Online Voting Remains Insecure," AAAS, 05 4 2025. [Online]. Available: https://www.aaas.org/epi-center/internet-online-voting.

[6] A. Voting, "Three Things You Need to Know about End-to-End Verifiability," Three Things You Need to Know about End-to-End Verifiability, 4 04 2025. [Online]. Available: https://assemblyvoting.com/blog/three-things-you-need-to-know-about-end-to-end-verifiability/.

[7] SecureWorld, "Mandating End-to-End Verifiable Voting Systems in U.S. Elections," Mandating End-to-End Verifiable Voting Systems in U.S. Elections, 2 4 2025. [Online]. Available: https://www.secureworld.io/industry-news/end-to-end-verifiable-voting-systems-usa-elections.

[8] T. T. Lihi Dery, "Towards Secure Virtual Elections: Multiparty Computation of Order Based Voting Rules," *Towards Secure Virtual Elections: Multiparty Computation of Order Based Voting Rules,* 2024.

[9] D. O. I. M.-M. J. J.-S. a. J. M. G.-G. The authors of the study are Marino Tejedor-Romero, "Distributed Remote E-Voting System Based on Shamir's Secret Sharing Scheme," *Distributed Remote E-Voting System Based on Shamir's Secret Sharing Scheme,* 2021.

[10] X. W. T. F. C. L. Junli Fang, "Multi party confidential verifiable electronic voting scheme based on blockchain," *Multi party confidential verifiable electronic voting scheme based on blockchain,* 2024.

[11] J. L. &. D. R. N. H. Ralf Küsters, "ZK-SNARKs for Ballot Validity: A Feasibility Study," *ZK-SNARKs for Ballot Validity: A Feasibility Study,* 2024.

[12] S. A. T. G. Rabia Fatih, "ZkSNARKs and Ticket-Based E-Voting: A Blockchain System Proof of Concept," *ZkSNARKs and Ticket-Based E-Voting: A Blockchain System Proof of Concept,* 2024.

[13] A. J. O. Henry O. Ohize, "Blockchain for securing electronic voting systems: a survey of architectures, trends, solutions, and challenges," *Blockchain for securing electronic voting systems: a survey of architectures, trends, solutions, and challenges,* 2024.

[14] O. M. O. E. M. D. D. O. O. Buhari Ugbede Umar*, "Paillier Cryptosystem Based ChainNode for Secure Electronic Voting," *Paillier Cryptosystem Based ChainNode for Secure Electronic Voting,* 2022.