A Comparative Analysis of SNARKs and Bulletproofs in Voting Systems

Matthew Richard (20474497, 23brlv@queensu.ca)

ABSTRACT

Elections are the cornerstone of modern democracies, yet they are vulnerable to countless security threats, such as duplicate and invalid voting. This paper investigates the use of zero-knowledge proofs (ZKPs) to enhance the security of voting systems, specifically focusing on SNARKs and Bulletproofs. Both schemes are implemented in a Rust-based voting system to address these vulnerabilities. The results show that SNARKs offer faster proof generation and more efficient memory usage, while Bulletproofs, which do not require a trusted setup, perform better in verification but have larger proof sizes. Overall, the results suggest that both SNARKs and Bulletproofs are valid solutions for improving security in election voting systems.

1 INTRODUCTION

Since ancient civilization, voting has served as the foundation in which society comes together to make decisions [1]. From ancient Roman and Greek senates, to the nationwide elections today, democracy is characterized by voting [1]. In our modern society, we frequently face voting cycles for our primary levels of government; municipal, provincial, and federal [1]. From decisions about local schools and healthcare to updating national policies, the results of elections determine the direction of our society; every vote shapes our future [5].

In our current day and age, elections utilize a hybrid voting approach, allowing for in person ballot submission, and online vote submission [5]. While this hybrid approach makes voting more accessible to society as a whole, it exposes elections to a larger range of security threats [2, 7]. Two critical vulnerabilities that stem from elections are duplicate voting and invalid voting [2, 7]. Without thorough verification, malicious actors could cast multiple votes, skewing the results of an election [2, 7]. Moreover, constraints that are unenforced can result in votes being submitted for invalid parties, leading to the vote being invalid [2, 7]. These vulnerabilities entirely threaten the integrity of elections, which places a great importance on security in voting systems.

To mitigate the security vulnerabilities of elections, this paper explores the use of zero-knowledge proofs (ZKP) in a voting system. Specifically, it examines the application of SNARKs and Bulletproofs to address issues like duplicate voting, invalid votes, and voter privacy. This paper focuses on the design and implementation of a ZKP voting system in Rust, and compares the performance of SNARKs and Bulletproofs. With this focus in mind, I first provide an overview of ZKP, SNARKs, and Bulletproofs, then discuss the design and implementation of a ZKP voting system, present the performance results, and discuss findings from the completed work.

Overall, this paper aims to demonstrate the validity and effectiveness of zero-knowledge proofs as a solution to security vulnerabilities in election systems.

2 BACKGROUND

To support the design and implementation of a secure voting system, this section outlines the cryptographic concept of zero-knowledge proofs. It begins with an overview of zero-knowledge proofs, followed by focused explanations of SNARKs and Bulletproofs, the two proof systems compared in this paper.

2.1 Zero-knowledge Proofs

Zero-knowledge proofs (ZKPs), first proposed by authors Shafi Goldwasser, Silvio Mical, and Charles Rackoff in their 1985 paper, "The Knowledge Complexity of Interactive Proof-Systems", are a cryptographic method where one party, the prover, convinces another party, the verifier, that a statement is true without revealing anything about the statement aside from its validity [4, 6, 12]. ZKPs are characterized by three principles: completeness, soundness, and zero-knowledge [4, 12]. Completeness is the principle that if a statement is true, a prover can always convince the verifier of the statements truth [4, 12]. Conversely, soundness guarantees that if a statement is false, a prover cannot convince the verifier that the statement is true [4, 12]. Most importantly, zero-knowledge is the foundational idea that if a statement is true, the verifier does not receive any additional information from the prover aside from the fact of the statements truth [4, 12]. These principles together create the ZKP cryptographic method that enforces security and privacy [4, 12].

To better understand that idea of ZKP, a widely used example is that of 'Ali Baba's Cave', stemming from the 1989 paper "How to Explain Zero-Knowledge Protocol to Your Children" by Quisquater et. al [9, 10]. In this example, there is a cave that forms a loop [10]. As such, there are two paths one can follow from the entrance of the cave, left or right [10]. Connecting the two paths and completing the loop is a door that requires a secret word to open [10]. Additionally, there are two characters, Peggy the prover who claims to know the secret word, and Victor the verifier who wants to verify that Peggy knows the secret word [10]. Peggy enters the cave and randomly chooses a path to follow, while Victory waits outside [10]. Victor then tells Peggy which path she must exit from [10]. If Peggy knows the doors secret word, she can come out of the correct path 100% of the time [10]. If she does not know the password, she can only leave from the path she entered, and will be correct 50% of the time [10]. If this process is repeated many times, and Peggy always exits the correct path, Victor can conclude that Peggy does in fact know the secret word, without knowing the word himself [10]. This example, while seemingly simple, perfectly demonstrates the core principles of zero-knowledge proofs discussed earlier; completeness, soundness, and zero-knowledge.

This cryptographic concept has grown significantly in popularity since its inception, and continues to be a foundational method for privacy-preserving applications, largely due to its many benefits. The privacy-preserving nature of ZKP is a universal solution

to many problems in the modern day [6, 12]. ZKPs have been applied to transactional applications as they can keep transaction amounts, sender, and receiver information private [6, 12]. Similarly, ZKPs have been used in identity verification systems to verify that someone is who they say they are, without exposing their sensitive information to anyone [6, 12]. From this overview, it is evident how ZKPs can be applied to a voting system to enhance the privacy of elections.

2.2 Types of Zero-knowledge Proofs

Zero-knowledge proofs are classified into two different categories, interactive and non-interactive [11]. Interactive ZKPs are proofs that require multiple rounds of communication between a prover and verifier [11]. The aforementioned 'Ali Baba's Cave' scenario demonstrates an interactive ZKP. Non-interactive ZKPs are proofs in which the prover generates a single proof for the verifier, and no further communication is required [11]. As a result of their simplicity, non-interactive ZKPs are more commonplace, which can be seen in the widespread use of SNARKs, STARKs, and Bulletproofs [11].

This paper focuses on the implementation of SNARKs and Bulletproofs.

2.3 SNARKs

Succinct Non-Interactive Arguments of Knowledge (SNARKs) are very popular non-interactive ZKP [8]. As the name suggests, SNARKs are characterized by their succinctness [8]. In other words, SNARKs are most known for having small proofs and fast verification times.

To perform the ZKP functionality, SNARKs convert the statement that is to be verified as true into an arithmetic circuit [8]. Next, the circuit is converted into a Quadratic Arithmetic Program (QAP) which is a formula that represents the constraints of the statement that is being verified [8]. What sets SNARKs apart from other proofs is its requirements of a trusted setup [8]. In the trusted setup, a Common Reference String (CRS) is generated which contains parameters gathered from the use of elliptic curve cryptography [8]. This CRS is used by the prover to create a proof of the initial statement [8]. After the proof is generated, the verifier uses pairing operations to validate the proof [8]. With both the proof being generated and verified, the SNARKs process is complete.

SNARKs have many advantages. Notably, the succinct nature of SNARKs ensures a small proof size which directly results in fast verification times [8]. This makes SNARKs an ideal proof scheme for systems with limited computational resources [8]. However, SNARKs do have disadvantages stemming from their use of a trusted setup process [8]. If the setup is compromised, the parameters can be reused to create valid proofs for fake statements, resulting in the ZKP system being untrustable [8].

In later sections, their practical performance and trade-offs will be examined in a voting system.

2.4 Bulletproofs

Proposed in 2018 by Bünz et al, Bulletproofs are another non-interactive ZKP that has quickly solidified itself as a staple of ZKP method [3]. This proof scheme is characterized by its use of range proofs and lack of a trusted setup [3].

To perform ZKP functionality, Bulletproofs begin with the prover generating a Pedersen commitment, which is a cryptographic technique that allows the prover to commit to a value while keeping it hidden [3]. An important property of the commitment is that it hides the committed value and cannot be modified in the ZKP process [3]. The commitment is then tested against a range proof, which ensures that the committed value falls within an allowed range [3]. Bulletproofs do so by converting the value into bits, and representing the range constraint as an inner production relation [3]. Then, an inner product argument is applied recursively to verify the constraint [3]. To generate the proof, public parameters called generators, created from elliptic curve cryptography, and the committed value are combined [3]. Finally, the proof verification is done by the verifier using the committed value and the generated proof [3]. Combined, this process allows for a zero-knowledge proof and verification.

The most notable advantage that Bulletproofs offer are its lack of a trusted setup [3]. This removes the vulnerabilities unique to a comprised trusted setup [3]. Additionally, the use of range proofs allows for easier implementation of numerical based proofs [3]. The biggest drawbacks of Bulletproofs are a larger proof size, and increased verification time [3]. These disadvantages draw questions regarding the limitations of the scalability of Bulletproofs.

The performance of Bulletproofs in a voting system will be further explored and measured in later sections.

3 METHODOLOGY

To achieve the goal of performing a comparative analysis of SNARKs and Bulletproofs in a voting system, a voting environment was developed using Rust. The Rust libraries, arksworks and bulletproofs, were utilized for the implementation of SNARKs and Bulletproofs, respectively. This section thoroughly details the implementation of both SNARKs and Bulletproofs, along with the experimental setup used to evaluate the performance of the proofs.

3.1 SNARKs Implementation

As previously discussed, a major component of SNARKs ZKPs are the circuits. In the implementation, a circuit was designed to verify that a voter's unique identifier belongs to a valid voter that has not voted before, and that their vote is for a valid predefined party. To ensure that a voter has not already voted, the circuit checks to see if the voter's identifier has already been recorded in the system for a previous vote. Furthermore, the circuit verifies that a valid party is being voted for by checking the submitted party identifier against a valid range of parties.

To finalize the SNARKs implementation, a trusted setup was performed to generate the parameters for the designed circuit. These parameters are then used by the prover and verifier to prove and verify the circuit using arkwork's builtin functions.

Through the arkworks library, we are able to directly mirror the SNARKs ZKP process that was previously outlined.

3.2 Bulletproofs Implementation

Bulletproofs were previously outlined with an emphasis placed on the use of range proofs. In the implementation, a range proof was designed to ensure that a voter's choice falls within the valid

Table 1: Performance Comparison of SNARKs and Bulletproofs in a Voting System

	SNARKs	Bulletproofs
Average Proof Generation Time	34.16 ms	275.36 ms
Average Verification Time	70.20 ms	39.94 ms
Average Proof Size	192 bytes	488 bytes
Memory Usage	8 KB	92 KB

range of party identifiers. The range proof guarantees that the vote corresponds to a valid party without revealing the specific choice made by the voter. To generate the range proof, the implementation creates an encrypted commitment of the vote. The range proof is then verified to confirm that the vote is both valid and within the predefined range of acceptable party identifiers.

Unlike SNARKs, Bulletproofs does not require a trusted setup. This allowed for a more simple implementation of the proof generation and verification.

Using the functionality of the bulletproofs library, the proof and verification process of Bulletproofs was simple to implement.

3.3 Experimental Setup

To evaluate the performance of both ZKP schemes, a simulation of a voting scenario was carried out. Four parties were initialized, Liberal, Conservative, NDP, and Green, mirroring the four most popular Canadian political parties. In each test, a set of voters were created each with a unique identifier created from a SHA-256 hash of the voter name and a blinding factor. Each voter was used to submit a vote to a different party, ensuring that proofs and verifications work for all parties. To test the proof schemes ability to prevent duplicate voting, an additional voter was created and attempted to submit two different votes.

The implementation of both SNARKs and Bulletproofs followed this experimental setup, with the only exceptions being the parts of the ZKP process that are unique to each proof. Following this setup ensured that each proof scheme was tested under the same conditions, allowing for accurate and valid comparisons.

3.4 Metrics Tracked

To achieve the goal of performing a comparative analysis of SNARKs and Bulletproofs in a voting system, it was imperative that metrics were recorded. The perform metrics measured for comparison were proof generation time, proof verification time, proof size, and memory usage.

To track both the proof generation and verification times, the built-in Rust time functions were used. The time started right before the proof generation and verification processes, and were measured immediately after the processes were finished. Next, the size of each proof was calculated by serializing the proof and measuring the resulting bit length. Lastly, memory usage was tracked by taking a measurement of the memory usage before the vote submission, proof generation, and verification, and then comparing it to the memory consumed after the process.

Each metric recorded was a calculated average from the proof generation and verification of five different votes that were completed in series, ensuring in a valid comparison. Overall, these performance metrics provide useful insight into the trade-offs of both SNARKs and Bulletproofs in a voting domain.

4 RESULTS

The results of the performance of SNARKs and Bulletproofs in a voting system are shown in Table 1. These results showcase distinct performance differences between SNARKs and Bulletproofs.

SNARKs demonstrate a greater efficiency in proof generation by 8x (34.16 ms vs. 275.36 ms). However, the verification times have the opposite results, with Bulletproofs being nearly 2x faster in verification time (70.20 ms vs. 39.94 ms). SNARKs once again beat Bulletproofs when it comes to proof sizes, as they are 2.5x smaller (192 bytes vs 488 bytes). In the last metric, SNARKs show an 11.5x smaller amount of memory usage (8 KB vs 92 KB).

These results suggest SNARKs are better suited for a large scale voting system with millions of votes that prioritizes compact proofs and fast proof verification time. On the other hand, Bulletproofs prove to be a good option for settings in which trust is not required and proof setup is not a major concern.

While the performance of SNARKs is overall more efficient in the voting domain, the vulnerabilities introduced by the trusted setup cannot be forgotten. To make SNARKs completely viable for elections, further security measures would need to be implemented.

The choice between these ZKP schemes ultimately depends on specific system requirements. Overall, each zero-knowledge proof scheme has their own unique advantages and disadvantages but are both valid approaches for improving election security.

5 CONCLUSION

This paper has explored the application of zero-knowledge proofs (ZKPs) in enhancing the security of voting systems under the lense of a comparative analysis of SNARKs and Bulletproofs. Through the implementation of both proof schemes in a Rust-based voting system, we have demonstrated how these ZKPs address vulnerabilities in elections such as invalid votes, duplicate voting, and voter privacy as a whole.

The performance analysis outlined the efficiency and trade-offs of both SNARKs and Bulletproofs. SNARKs have a faster proof generation, as well as more efficient memory usage. As a result, SNARKs appear to be better for large scale voting systems, such as federal elections, in which there are millions of votes. Despite their excellent performance, their use of a trusted setup creates new vulnerabilities that must be accounted for in a real-world ZKP voting system application. On the other hand, Bulletproofs do not required a trusted-setup and had a better verification performance. However, their large proof size and lengthy proof generation time bring into question the scalability of Bulletproofs. Overall, both

SNARKs and Bulletproofs are a valid ZKP solution for improving the security of voting systems in elections.

Future work could explore enhancing the security of SNARKs by implementing solutions to the vulnerabilities introduced by the trusted setup. Additionally, combining zero-knowledge proofs with identity verification could add an extra layer of security against voter fraud. Lastly, implementing this ZKP voting system in a real-world environment with a user interface is a natural progression to fully test its practical use.

On the whole, as hybrid voting systems become increasingly prevalent, further research into election security through techniques like zero-knowledge proofs will be vital for preserving the integrity of democracy.

REFERENCES

[1] [n. d.]. Election | History, Polls, Results, Date, & Facts | Britannica — britannica.com. https://www.britannica.com/topic/election-political-science. [Accessed 05-04-2025].

[2] Matt Bishop and Sean Peisert. 2012. Security and Elections. *IEEE Security Privacy* 10, 5 (2012), 64–67. https://doi.org/10.1109/MSP.2012.127

[3] Benedikt Bünz, Jonathan Bootle, Dan Boneh, Andrew Poelstra, Pieter Wuille, and Greg Maxwell. 2018. Bulletproofs: Short Proofs for Confidential Transactions and More. In *2018 IEEE Symposium on Security and Privacy (SP)*. 315–334. https://doi.org/10.1109/SP.2018.00020

[4] Uriel Feige, Amos Fiat, and Adi Shamir. 1988. Zero-knowledge proofs of identity. *Journal of Cryptology* 1, 2 (01 Jun 1988), 77–94. https://doi.org/10.1007/BF02351717

[5] Alan Gerber, Gregory Huber, David Doherty, and Conor Dowling. 2014. Why People Vote: Estimating the Social Returns to Voting. *British Journal of Political Science* 46 (10 2014), 1–24. https://doi.org/10.1017/S0007123414000271

[6] Siqi Liu. 2022. Privacy Protection Revolution: Zero-knowledge Proof. *2022 International Conference on Data Analytics, Computing and Artificial Intelligence (ICDACAI)* (2022), 394–397. https://api.semanticscholar.org/CorpusID:254737078

[7] Muharman Lubis, Mira Kartiwi, and Sonny Zulhuda. 2016. Election fraud and privacy related issues: Addressing electoral integrity. *2016 International Conference on Informatics and Computing (ICIC)* (2016), 227–232. https://api.semanticscholar.org/CorpusID:14105581

[8] Anca Nitulescu. [n. d.]. zk-SNARKs: A Gentle Introduction. ([n. d.]), 50.

[9] Jean-Jacques Quisquater, Myriam Quisquater, Muriel Quisquater, Michaël Quisquater, Louis Guillou, Marie Annick Guillou, Gaïd Guillou, Anna Guillou, Gwenolé Guillou, and Soazig Guillou. 1990. How to Explain Zero-Knowledge Protocols to Your Children. In *Advances in Cryptology — CRYPTO' 89 Proceedings*, Gilles Brassard (Ed.). Springer New York, New York, NY, 628–631.

[10] Tabacaru Robert, Anghel Florin, Asandoaiei David, and Simion Emil. 2023. The challenges of proving solvency while preserving privacy. Cryptology ePrint Archive, Paper 2023/079. https://eprint.iacr.org/2023/079

[11] Chainalysis Team. [n. d.]. Introduction to Zero-Knowledge Proofs - Chainalysis — chainalysis.com. https://www.chainalysis.com/blog/introduction-to-zero-knowledge-proofs-zkps/#ZKP-types. [Accessed 05-04-2025].

[12] Justin Thaler. 2022. Proofs, Arguments, and Zero-Knowledge. *Foundations and Trends® in Privacy and Security* 4, 2–4 (2022), 117–660. https://doi.org/10.1561/3300000030