FP-BSS: Full Privacy Blockchain-Based Self-Tallying Score Voting Protocol

No Author Given

No Institute Given

Abstract. Electronic voting has a wide range of applications in modern society. Self-tallying voting, which allows all voters to calculate the results themselves, is gaining popularity as it eliminates the need for a centralized authority. However, existing self-tallying voting schemes face challenges, such as the lack of a score voting mechanism and inadequate privacy protection. To address these issues, we propose FP-BSS, a full privacy, blockchain-based self-tallying score voting protocol. We extend the traditional binary voting paradigm to support score voting, enabling voters to rate each candidate within a specified range. Candidates with broad acceptability tend to win, while extreme candidates favored by a minority but opposed by the majority are less likely to win. Simultaneously, to guarantee ballot privacy both during and after the election, we propose a full privacy ballot encryption scheme leveraging blinding factors to ensure complete voter anonymity, and propose a new zero-knowledge proof to verify ballot legitimacy. We prove that FP-BSS achieves full voter privacy and meets all necessary security requirements. Performance results demonstrate that FP-BSS provides enhanced security while maintaining reasonable practicality.

Keywords: Blockchain · score voting · self-tallying · zero-knowledge proof · homomorphic encryption.

1 Introduction

With the development of science and technology, electronic voting is gradually gaining attention[1]. Electronic voting systems offer many advantages over traditional paper-based voting. For example, they eliminate geographical constraints, and the processes of tallying and verifying results are highly efficient.

However, traditional electronic voting systems also face security challenges. Systems that rely on a centralized authority to manage the voting process prevent voters from verifying whether their ballots are accurately recorded and counted, thereby undermining trust in the credibility of the system. Blockchain technology addresses the shortcomings of traditional electronic voting protocols. It is a distributed, decentralized ledger that records digital transactions securely and transparently [2,3], eliminating the need for a centralized authority. Permissioned blockchains enable multiple organizations to securely share data in a decentralized yet controlled environment. They offer greater openness than private

chains while providing stronger governance than public ones. This combination of features makes permissioned blockchains an ideal foundation for electronic voting systems. In recent years, a series of blockchain-based voting schemes have been proposed, enhancing security throughout the voting process [4].

However, existing blockchain-based electronic voting protocols suffer from various security limitations:

First, most existing systems focus only on binary voting, with limited exploration of more flexible voting paradigms. Lu et al. proposed a blockchain-based self-tallying protocol using a traceable ring signature [5]. Their approach only supports single binary voting rather than allowing voters to evaluate each candidate within a predefined range. This limitation can lead to situations where extreme candidates, those favored by a minority but strongly opposed by the majority win elections.

Second, existing voting schemes may compromise voter privacy. Miao et al. proposed a blockchain-based publicly traceable self-tallying voting protocol that uses time-lock puzzles to ensure time-limited privacy [6]. In this scheme, election results can only be computed after a fixed period elapses. However, ballot contents are revealed after the election, exposing the choices of the voters.

After investigating existing schemes, we propose FP-BSS, a full privacy, blockchain-based self-tallying score voting protocol that addresses the aforementioned problems. The contributions of this paper are summarized below:

(1) We propose a new full privacy ballot encryption scheme based on blinding factors, which ensures voter privacy both during and after the voting process. Specifically, we employ homomorphic encryption to encrypt ballots and then blind the ciphertext using blinding factors. This approach ensures ballot privacy while enabling self-tallying. We extend the traditional binary voting scheme to enable voters to evaluate candidates within a predefined range.
(2) We propose a new zero-knowledge proof for verifying the legitimacy of encrypted ballots. This proof ensures that: the correct blinding factor was used to encrypt the ballot, each score on a ballot lies within a predefined range, and the total sum of points sums to a predefined constant while preserving the complete confidentiality of the encrypted ballot.
(3) We implement FP-BSS, analyze its security, and compare it with existing self-tallying voting schemes. The results demonstrate that our protocol satisfies all requisite security requirements while providing comprehensive security guarantees. Performance evaluation confirms that our protocol maintains practical efficiency.

The remainder of the paper is organized as follows. Section 2 discusses related work on protecting privacy and security in electronic voting schemes. Section 3 introduces the preliminaries required for FP-BSS. Section 4 describes the system architecture and workflow of FP-BSS. Section 5 presents the concrete construction of FP-BSS. Section 6 provides a comprehensive security analysis of FP-BSS, followed by the implementation details and performance evaluation in Section 7. Finally, Section 8 presents the conclusion.

2 Related Work

In this section, we introduce works related to protecting privacy and security in electronic voting systems.

Traditional Electronic Voting Systems. With the advancement of information technologies, electronic voting systems have achieved significant gains in efficiency, but their security enhancements remain comparatively limited [8]. Traditional electronic voting systems usually rely on a centralized election authority to manage the voting process. The election results are announced solely by this centralized authority, which undermines trust in their credibility. This is because a centralized election authority could easily alter vote data, making participants unable to verify the authenticity of the election results. Moreover, centralized electronic voting systems are vulnerable to attacks, which can lead to system failures and privacy breaches.

Self-Tallying Voting Systems. Self-tallying voting eliminates the need for a centralized authority. All voters can collect all legitimate ballots and count the results themselves. In 2002, Kiayias et al. first proposed the concept of self-tallying [9]. Later, Hao et al. proposed a lightweight self-tallying voting scheme [10]; however, this scheme requires that all voters participate and cast valid ballots. If malicious voters refuse to participate, the final result cannot be computed. Self-tallying voting inherently faces abortive and adaptive issues. The abortive issue occurs if malicious voters do not cast their encrypted ballots, preventing computation of the final result. The adaptive issue arises because the last voter could calculate the vote result in advance.

Blockchain-Based Electronic Voting Systems. Although self-tallying protocols address centralization risks, their practical implementation faces challenges in ensuring universal verifiability. The emergence of blockchain has offered solutions to this problem. McCorry et al. proposed the first self-tallying voting protocol implemented on Ethereum [11]. However, this protocol did not resolve the abortive and adaptive issues and lacked support for score voting. Later, Khader et al. proposed a fair and robust protocol by adding a commitment round and a recovery round to address the problem of the last voter potentially calculating the result in advance [12], but it still suffered from the abortive issue. Then, Lin et al. proposed a complete self-tallying protocol to resolve the adaptive issue using Schnorr signatures [13], though it incurred additional overhead. In 2020, Li et al. proposed an electronic voting protocol with traceability using linkable group signatures [14], which enables tracing of malicious voters who vote twice. However, it requires the election authority to trace these voters, potentially compromising voter privacy. In 2022, Li et al. proposed a blockchain-based electronic voting protocol that solves the abortive issue [15], but the protocol only supports binary voting. To our knowledge, none of the existing protocols address all of the above problems simultaneously.

3 PRELIMINARIES

3.1 Score Voting

Score voting is a simple yet expressive decision making method where voters assign each candidate a score within a predefined range. Rather than merely choosing yes or no, voters can express their level of support for each candidate. The final result more accurately reflects the collective preferences of all voters. After voting concludes, the scores for each candidate are summed, and the candidate with the highest total score wins.

For example: In a score voting election with three candidates (Alice, Bob, Mary) and three voters, the first voter scores Alice 10, Bob 6, Mary 0; the second scores Alice 0, Bob 6, Mary 10; and the third scores Alice 6, Bob 6, Mary 4. While Alice and Mary each receive both perfect support and strong rejection from different voters, Bob wins with 18 total points (Alice: 16, Mary: 14) because he is broadly acceptable to all voters.

3.2 Time-Lock Puzzle

A Time-Lock Puzzle is a cryptographic technique that conceals information until a specific time period elapses [17]. It works by creating a computational puzzle designed to take a fixed duration to solve, even on powerful computers. This ensures that the data remains secret until the predetermined time passes.

However, time-lock puzzles do not support computations on hidden data. This limits their applicability. To address this limitation, Malavolta et al. introduced a method called the Homomorphic Time-Lock Puzzle (HTLP) [20]. In essence, HTLP is an enhanced version of the time-lock puzzle that enables computations on multiple encrypted puzzles using a predefined circuit, without requiring access to the hidden values within them. The final result of these computations is itself another puzzle.

3.3 Zero-Knowledge Proof of Knowledge

A zero-knowledge proof of knowledge is a cryptographic protocol that enables a prover to convince a verifier of possessing certain secret information without disclosing that information. The Σ-protocol is a common type of interactive zero-knowledge proof [22]. It satisfies the properties of completeness, special soundness, and special honest-verifier zero-knowledge. Any Σ-protocol can be transformed into a non-interactive version by replacing the random challenge with a hash of the commitment. The Fiat-Shamir heuristic is a technique for converting interactive proofs into non-interactive ones [23].

4 System Architecture

According to FP-BSS, the voting system consists of four core entities: **Election Authority**, **Voters**, **Observers**, and **Blockchain**. The system architecture is shown in Fig. 1.

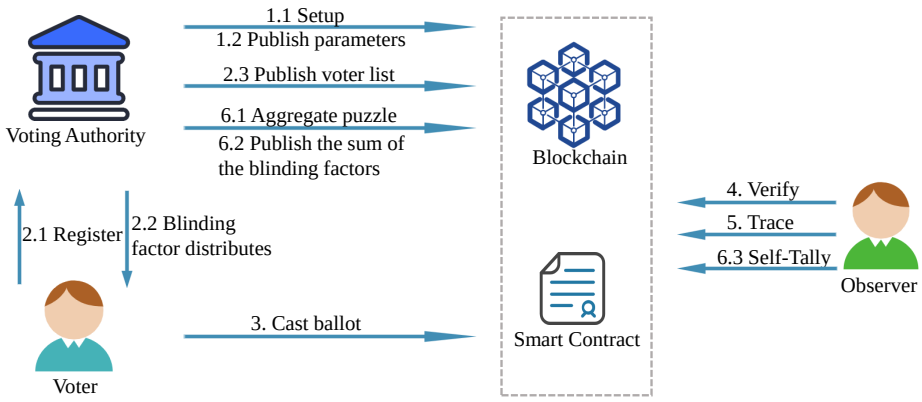


Fig. 1: System architecture

Election Authority (EA): The EA is responsible for election setup and publishes public voting parameters on the Blockchain. It handles voter registration and distributes blinding factors to legitimate voters.

Voters: A voter V_i registers with the EA to obtain a blinding factor, then shares their public key and corresponding proof on the Blockchain. During voting, V_i publishes the encrypted ballot and associated proof to the Blockchain to demonstrate ballot legitimacy.

Observers: An observer is any entity monitoring the election. Observers can: authenticate voter identities, verify encrypted ballot legitimacy, trace malicious voters attempting double voting, and independently compute final election results.

Blockchain: The Blockchain serves as an immutable ledger that permanently records: public keys of voters with corresponding proofs, and encrypted ballots paired with their validation proofs.

FP-BSS consists of three stages, the pre-voting stage including Setup and Registration algorithm, the voting stage including BallotCast algorithm and the post-voting stage including BallotVerfication, Trace and Self-Tally algorithm.

The notations used in this paper are summarized in Table 1.

Setup$(\lambda, eventID, t, m, R) \rightarrow (PP, \{C_j\}_{j=1}^m)$**:** Suppose there are m candidates. The EA selects security parameters λ, an event identifier $eventID \in \{0,1\}^*$, a time hardness parameter t, and a predefined total point R. It chooses a cyclic group G of prime order p with generator g. Let $N = \tilde{j} \times \tilde{k}$ be an RSA modulus where $\tilde{j}$ and $\tilde{k}$ are strong primes. Define hash functions $H : \{0,1\}^* \rightarrow G$ and $H_1 : \{0,1\}^* \rightarrow \mathbb{Z}_p^*$. The EA randomly selects $u \in \mathbb{Z}_N^*$, computes $T = u^{2^t}$, and publishes $PP = \{p, G, g, H, H_1, N, T, u, t, eventID\}$ and candidate list $\{C_j\}_{j=1}^m$ on the Blockchain.

Registration$(PP, VID_i) \rightarrow (pk_i, sk_i, b_i, n, VL)$**:** Each voter V_i registers with the EA by selecting random $x_i \in \mathbb{Z}_p^*$ as their private key sk_i, computing public key $pk_i = g^{x_i}$, and sending (pk_i, π_i, VID_i) to the EA. The EA verifies proof π_i; if valid, it assigns blinding factor f_i, selects random $s_i \in \mathbb{Z}_p^*$, computes $Y_i = g^{s_i}$

Table 1: Notations and Description

Notation	Description
n	Number of voters
m	Number of candidates
N	RSA integer
R	Total points allocated to all candidates
t	Time hardness parameter
$eventID$	Event Identifier
f_i	The blinding factor of V_i
α_i	The auxiliary blinding factor verification values
VL	Voter list
p_{ij}	The point allocate to C_j by V_i
$Ballot_{ij}$	Encrypted ballot of V_i for C_j
$Ballot_i$	Encrypted ballot generated by V_i
$NIZK_{ij}$	Zero-knowledge range proof of the ballot $Ballot_i$
$NIZK_i$	Zero-knowledge total proof of the ballot $Ballot_i$
σ_i	Signature signed by V_i for the ballot $Ballot_i$
p_j	Total point of the candidate C_j

and $\alpha_i = Y_i^{f_i}$, then adds (pk_i, π_i, α_i) to voter list VL. After all registrations, VL is published on the Blockchain.

BallotCast$(PP, PK_N, sk_i, \{p_{ij}\}_{j=1}^m) \rightarrow$ **(**$Ballot_i, NIZK_{Ballot_i}, \sigma_i$**):** V_i encrypts their ballot to obtain $Ballot_i$, generates zero-knowledge range proof $NIZK_{ij}$ and total proof $NIZK_i$ (see Section 5), then signs $Ballot_i$ as shown in Fig. 2 to produce signature σ_i. Finally, $(Ballot_i, NIZK_{ij}, NIZK_i, \sigma_i)$ is published on the Blockchain, which allows public verifiability of the ballot.

BallotVerification(PP**,** $(Ballot_i,\ NIZK_{Ballot_i})$**)** $\rightarrow \{0,1\}$**:** Any party verifies $Ballot_i$ by checking $NIZK_{ij}$ and $NIZK_i$ (see Section 5). For $\sigma_i = (D_1, C_N, K_N)$, validate whether $H(eventID \parallel Ballot_i \parallel D_0 \parallel D_1 \parallel r_N \parallel t_N) = \sum_{i=1}^n C_i$ holds. If any check fails, reject the ballot.

Trace(PP, PK_N**,** $(Ballot_i, \sigma_i)$**,** $(Ballot_i', \sigma_i')$**)** $\rightarrow \{pk_i\}$**:** Given two valid ballot-signature pairs, any party can determine if they come from the same voter. For σ_i, compute $q = H(eventID \parallel PK_N)$, $D_0 = H(eventID \parallel Ballot_i)$, $\eta_i = D_0 \cdot D_1^i$. Compute η_i' similarly for σ_i'. For $i \in \{1, 2, ..., n\}$, if only one of $\eta_i = \eta_i'$ is true, the public key of the malicious voter is pk_i.

Self-Tally($PP, \{Ballot_i\}_{i=1}^n$**)** $\rightarrow (p_1, \ldots, p_m)$**:** The EA runs Trace to identify malicious voters. Let ν be the set of legitimate voters. Aggregate valid ballots:

$$(a_j, b_j) = (\prod_{i \in \nu} a_{ij}, \prod_{i \in \nu} b_{ij}) \tag{1}$$

Then publish aggregate result (a_j, b_j) and $F_\nu = \sum_{i \in \nu} f_i$ on the Blockchain. After time t, any party computes:

$$c_j = (a_j / u^{F_\nu})^{2^t} \tag{2}$$

$$p_j = \frac{(b_j / c_j^N \mod N^2) - 1}{N} \tag{3}$$

The candidate scores are $(p_1, \ldots, p_m)$.

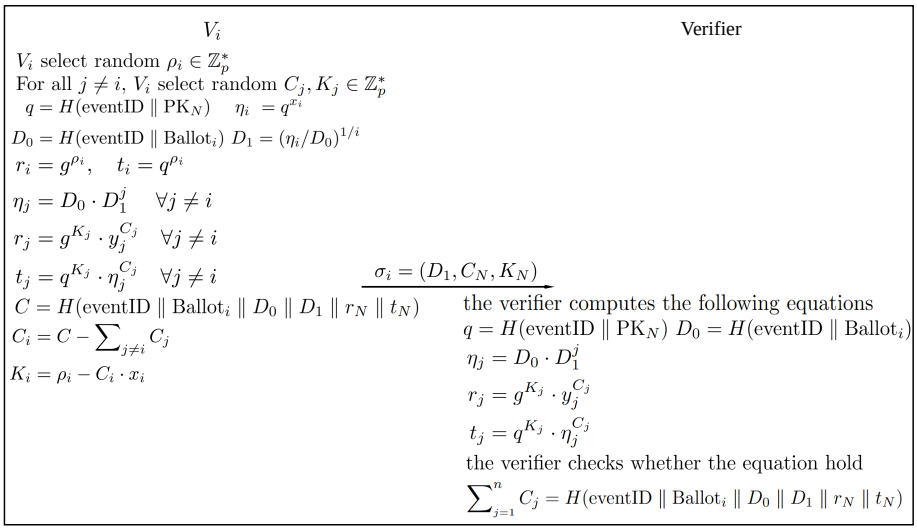


Fig. 2: Sign the encrypted ballot $Ballot_i$

5 Full Privacy Blockchain-Based Self-Tallying Score Voting Protocol

In this section, we present our new full privacy ballot encryption scheme and new zero-knowledge proof of encrypted ballot legitimacy.

5.1 A new full privacy ballot encryption scheme based on the blinding factor.

V_i allocates a point p_{ij} to candidate C_j according to the voting rules. To protect the privacy of the ballot, V_i encrypts the ballot with time-lock puzzles and blinding factors. To be specific, V_i selects $h_{ij} \in Z_p^*$ at random, and calculates a_{ij} and b_{ij} according to the following equations:

$$a_{ij} = u^{h_{ij}} \cdot u^{f_i} \tag{4}$$

$$b_{ij} = T^{h_{ij} \cdot N} \cdot (1+N)^{p_{ij}} \tag{5}$$

The point of V_i for the candidate C_j is $Ballot_{ij} = (a_{ij}, b_{ij})$.

5.2 A new range zero-knowledge proof.

V_i generates a non-interactive zero-knowledge range proof to prove that point p_{ij} belongs to $[0, R]$ and the correct blinding factor is used without revealing p_{ij} or the blinding factor h_{ij}. The details of range zero-knowledge proof are shown in Fig. 3. The specific definitions are provided below:

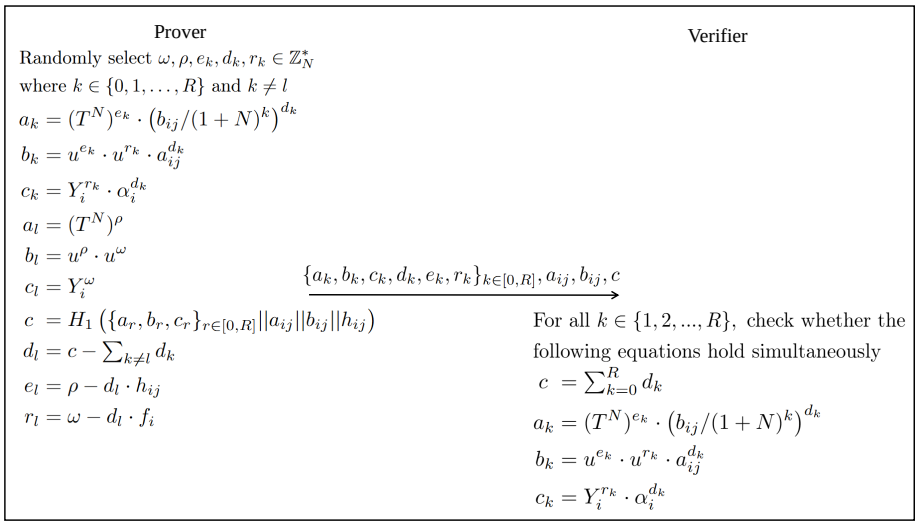


Fig. 3: Zero-knowledge range proof of the ballot $Ballot_i$

$NIZK_{ij}\{(h_{ij}, p_{ij}, f_i) : a_{ij} = u^{h_{ij}} \cdot u^{f_i} \wedge \alpha_i = Y_i^{f_i} \wedge (b_{ij} = T^{h_{ij}\cdot N} \vee b_{ij} = T^{h_{ij}} \cdot (1+N) \vee b_{ij} = T^{h_{ij}} \cdot (1+N)^2 \vee ... \vee b_{ij} = T^{h_{ij}} \cdot (1+N)^R)\}$
For $NIZK_{ij}$, the correctness of the following equations should be checked. $c = \sum_{k=0}^{R} d_k, a_k = (T^N)^{e_k} \cdot (b_{ij}/(1+N)^k)^{d_k}, b_k = u^{e_k} \cdot u^{r_k} \cdot a_{ij}^{d_k}, c_k = Y_i^{r_k} \cdot \alpha_i^{d_k}$. If one of these equations is not true, the verifier aborts.

5.3 A new totol zero-knowledge proof.

V_i also generates a non-interactive zero-knowledge total proof to prove that the total of the point p_{ij} equals R and the correct blinding factor is used without revealing p_{ij} or the blinding factor h_{ij}. The details of totol zero-knowledge proof are shown in Fig. 4. The specific definitions are provided below:
$NIZK_i\{((h_{i1}, h_{i2}, ..., h_{im}), (p_{i1}, p_{i2}, ..., p_{im}), f_i) : \prod_{j=1}^{m} a_{ij} = u^{\sum_{j=1}^{m} h_{ij}} \cdot u^{\sum_{j=1}^{m} s_i} \wedge \prod_{j=1}^{m} b_{ij} + (T^N)^{\sum_{j=1}^{m} h_{ij}} \cdot (1+N)^R \wedge \sum_{j=1}^{m} p_{ij} = R \wedge \alpha_i = Y_i^{f_i}\}$
For $NIZK_i$, the following equations should be checked. $c = H_1(\tilde{a}||\tilde{b}||\tilde{\alpha}||\prod_{j=1}^{m} a_{ij}||\prod_{j=1}^{m} b_{ij}||\alpha_i), \tilde{a} = (\prod_{j=1}^{m} a_{ij})^c \cdot u^{(\tilde{x_i}+\sum_{j=1}^{m}\alpha)}, \tilde{b} = (\prod_{j=1}^{m} b_{ij}/(1+N)^R)^c \cdot (T^N)^{\tilde{x_i}}, \tilde{\alpha} = \alpha_i^c \cdot Y_i^{\alpha}$. If one of these equations is not true, the verifier aborts.

5.4 Correctness of proofs

Proof. Correctness of "range" NIZK proof:

$$a_l = (T^N)^{\rho} = (T^N)^{e_l + d_l \cdot h_{ij}} = (T^N)^{e_l} \cdot ((T^N)^{h_{ij}})^{d_l} = (T^N)^{e_l} \cdot (b_{ij}/(1+N)^l)^{d_l},$$
$$b_l = u^{\rho} \cdot u^{\omega} = u^{e_l + d_l \cdot h_{ij}} \cdot u^{r_l + d_l \cdot f_i} = u^{e_l} \cdot u^{r_l} \cdot u^{d_l \cdot (h_{ij} + f_i)} = u^{e_l} \cdot u^{r_l} \cdot a_{ij}^{d_l},$$
$$c_l = Y_i^{\omega} = Y_i^{r_l + d_l \cdot f_i} = Y_i^{r_l} \cdot (Y_i^{f_i})^{d_l} = Y_i^{r_k} \cdot \alpha_i^{d_k}.$$

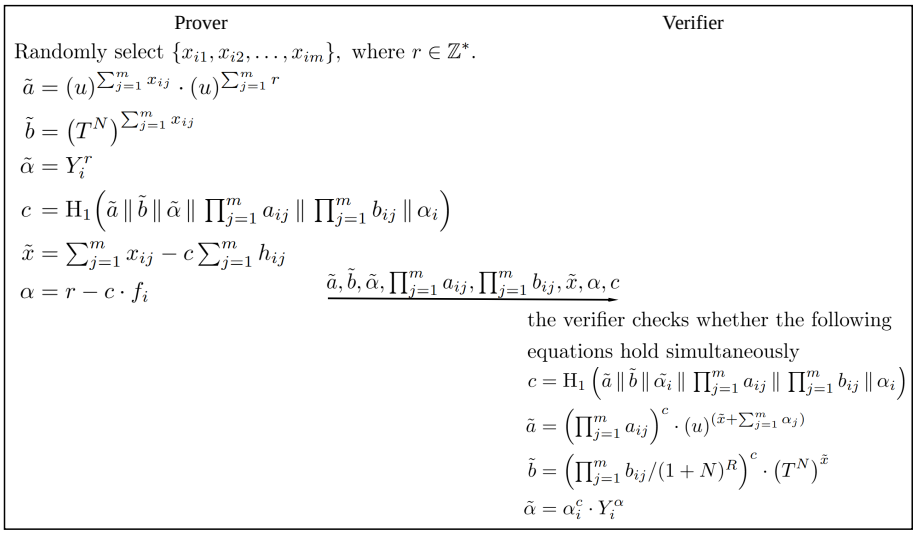


Fig. 4: Zero-knowledge total proof of the ballot $Ballot_i$

Proof. Correctness of "total" NIZK proof:

$$\begin{aligned}
\tilde{a} &= u^{\sum_{j=1}^{m} x_{ij}} \cdot u^{\sum_{j=1}^{m} r} = u^{\tilde{x} + c \cdot \sum_{j=1}^{m} h_{ij}} \cdot u^{\sum_{j=1}^{m} (\alpha + c \cdot f_i)} \\
&= (u^{\sum_{j=1}^{m} h_{ij} + \sum_{j=1}^{m} f_i})^c \cdot u^{\tilde{x} + \sum_{j=1}^{m} \alpha} = (\prod_{j=1}^{m} a_{ij})^c \cdot u^{\tilde{x} + \sum_{j=1}^{m} \alpha}, \\
\tilde{b} &= (T^N)^{\sum_{j=1}^{m} x_{ij}} = (T^N)^{\tilde{x} + c \cdot \sum_{j=1}^{m} h_{ij}} = ((T^N)^{\sum_{j=1}^{m} h_{ij}})^c \cdot (T^N)^{\tilde{x}} \\
&= (\prod_{j=1}^{m} b_{ij} / (1+N)^R)^c \cdot (T^N)^{\tilde{x}}, \\
\tilde{\alpha} &= Y_i^r = Y_i^{\alpha + c \cdot f_i} = (Y_i^{f_i})^c \cdot Y_i^{\alpha} = \alpha_i^c \cdot Y_i^{\alpha}.
\end{aligned}$$

Proof. Correctness of the vote result:

$$\begin{aligned}
p_j &= \frac{b_j / c_j^N \bmod N^2 - 1}{N} = \frac{\prod_{i=1}^{n} T^{h_{ij} \cdot N} \cdot (1+N)^{p_{ij}} / \prod_{i=1}^{n} T^{h_{ij} \cdot N} \bmod N^2 - 1}{N} \\
&= \frac{(1+N)^{\sum_{i=1}^{n} p_{ij}} \bmod N^2 - 1}{N} = \frac{1 + (\sum_{i=1}^{n} p_{ij}) \cdot N - 1}{N} = \sum_{i=1}^{n} p_{ij}.
\end{aligned}$$

6 Security Analysis

6.1 Security Properties

FP-BSS meets the following security properties.

Full privacy. Voter anonymity is preserved throughout the election lifecycle, including during ballot casting, tallying, and result publication, under the condition that no voter submits multiple ballots.

Time-limited privacy. During preset time t, ballots remain completely confidential against adversaries with arbitrary computational power.

Linkability. In a same voting event, two ballots cast by a same voter will be

linked. Legitimate single ballots remain computationally unlinkable even under coalition attacks.

Universal verifiability. Any observer can independently authenticate voter identities, validate encrypted ballots, and verify that all ballots are accurately included in the final tally.

Self-tallying. After the time-lock expiration. Any election observer can independently compute the final voting results. No trusted authority is required for result computation.

Robustness. The final election results cannot be manipulated by any party, including EA itself. Valid results remain computable and verifiable even if voters abstain or act maliciously."

6.2 Security Analysis

For the essential security requirements discussed above, we conduct an analysis of FP-BSS.

Full privacy: In the protocol, each ballot is encrypted with a time hardness parameter t and then blinded with a voter-specific blinding factor f_i. The ballot encryption takes the form:

$$Ballot_{ij} = (a_{ij}, b_{ij}) = (u^{h_{ij}} \cdot u^{f_i},\ T^{h_{ij} \cdot N} \cdot (1+N)^{p_{ij}}). \tag{6}$$

The blinding factor ensures persistent privacy of ballot contents. Crucially, even after the predefined time t elapses, the specific content of individual ballots remains protected due to the blinding operation. This guarantees end-to-end confidentiality of votes, preserving voter privacy throughout the process.

Time-limited privacy: Ballot encryption incorporates $T = u^{2^t}$, which requires at least t sequential squaring operations. This computation cannot be parallelized because each step depends on the previous result. When t is sufficiently large, real-time computation of T becomes computationally infeasible, thereby enforcing time-limited privacy.

Linkability: Each voter generates a signature and corresponding proof using their unique private key. This cryptographic binding ensures that ballots from distinct voters are computationally unlinkable, while simultaneously enabling detection of duplicate votes.

Universal verifiability: Voters publish encrypted ballots, range proofs $NIZK_{ij}$, total proofs $NIZK_i$, and signatures with proofs to the blockchain, enabling any observer to: verify ballot validity via $NIZK_{ij}$ and $NIZK_i$, authenticate signatures, and optionally recompute final results during self-tallying. This enables universal verification of both individual ballots and aggregated results.

Self-tallying: After the voting phase, any party can collect all ballots (a_{ij}, b_{ij}) and the aggregated blinding factor $F_\nu = \sum_{i \in \nu} f_i$, compute the aggregated values $(a_j, b_j) = \left(\prod_{i \in \nu} a_{ij},\ \prod_{i \in \nu} b_{ij}\right)$, then decrypt the tally through $c_j = (a_j / u^{F_\nu})^{2^t}$ and $p_j = \frac{(b_j / c_j^N \mod N^2) - 1}{N}$, eliminating the need for trusted third parties.

Robustness: Blockchain persistence ensures the immutability of cast ballots. Although EA publishes the final sum of blinding factors F_ν, it also includes all auxiliary verification values α_i in the VL. This design enables:

(1) Voters to cryptographically verify blinding factor consistency during ballot encryption by checking α_i against values published by the EA.
(2) Independent validation of the authenticity of F_ν during tallying through α_i using zero-knowledge principles.

This design enhances protocol robustness against potential EA misbehavior.

6.3 Comparison with Self-Tallying Voting Schemes

We compare FP-BSS with existing self-tallying voting schemes. Our analysis reveals that while current systems partially satisfy certain security requirements, each exhibits significant limitations. In contrast, our protocol robustly achieves comprehensive coverage of all essential security properties. Detailed comparisons are presented in Table 2, demonstrating that no existing scheme provides the complete security guarantees offered by our solution.

Table 2: Comparison of Self-Tallying Voting Schemes

	[5]	[6]	[7]	[13]	[15]	Ours
Full privacy	●	×	●	×	×	●
Time-limited privacy	●	●	×	●	×	●
Linkability	●	●	×	●	×	●
Universal verifiability	●	●	●	○	●	●
Robustness	×	●	●	●	○	●
Voting type	Binary Score Voting		Binary Choose-One		Choose-One Score Voting	

●: completely satisfy; ○: partly satisfy; ×: not satisfy

We analyze FP-BSS computation and communication costs per stage as detailed in Table 3. Registration requires each voter to generate a key pair with proof at cost of two exponentiations and one hash operation; BallotCast involves ballot encryption, proof generation, and signature creation with costs scaling linearly by candidate count, voter count and predefined total R, yielding computation cost $(9Rm+9m+5n+4)t_e+(5Rm+3m+3n)t_m+5t_h$ and communication cost $(3Rm+6m+4)|Z_N^*|+(Rm+3m+2)|Z_{N^2}^*|+(Rm+4m+2)|Z_p^*|+|G|$; The situation is similar for BallotVerification; Trace requires $2nt_e+2nt_m+2t_h$ computation with zero communication; Self-Tally computation costs by candidate count as $3mt_e+3mt_m$.

7 Performance Evaluation

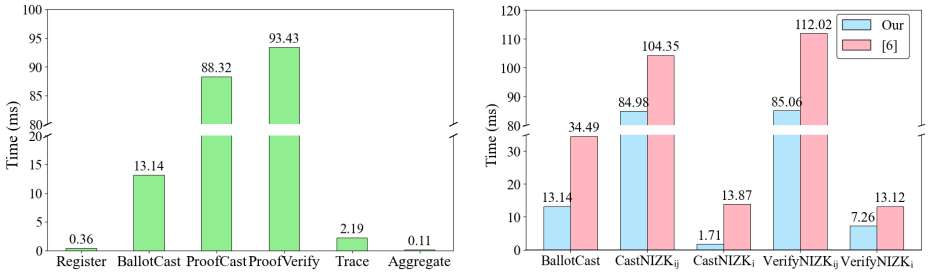
We implement FP-BSS and conduct a comprehensive performance evaluation. Our experiments included both off-chain and on-chain parts. Each experiment is

Table 3: Computation and Communication Cost

Phase	Computation Cost	Communication Cost
Registration	$2t_e + t_h$	$2\|Z_p^*\| + 2\|G\|$
BallotCast	$(9Rm + 9m + 5n + 4)t_e +$ $(5Rm + 3m + 3n)t_m + 5t_h$	$(3Rm + 6m + 4)\|Z_N^*\| + (Rm + 3m + 2)$ $\|Z_{N^2}^*\| + (Rm + 4m + 2)\|Z_p^*\| + \|G\|$
BallotVerification	$(6Rm + 6m + 6 + 5n)t_e +$ $3(Rm + m + n + 1)t_m + 4t_h$	–
Trace	$2nt_e + 2nt_m + 2t_h$	–
Self-Tally	$3mt_e + 3mt_m$	–

t_e: time of a modular exponentiation; t_m: time of a modular multiplication; t_h: time of hash calculation; $|G|$: size of element in G; $|Z_p^*|$: size of element in Z_p^*; $|Z_N^*|$: size of element in Z_N^*; $|Z_{N^2}^*|$: size of element in $Z_{N^2}^*$;

repeated 10 times with averaged results. The hardware environment we used is a PC with Intel(R) Core(TM) i7-8565U CPU @ 1.80GHz 1.99 GHz, 8 GB RAM and Ubuntu 20.04 system.

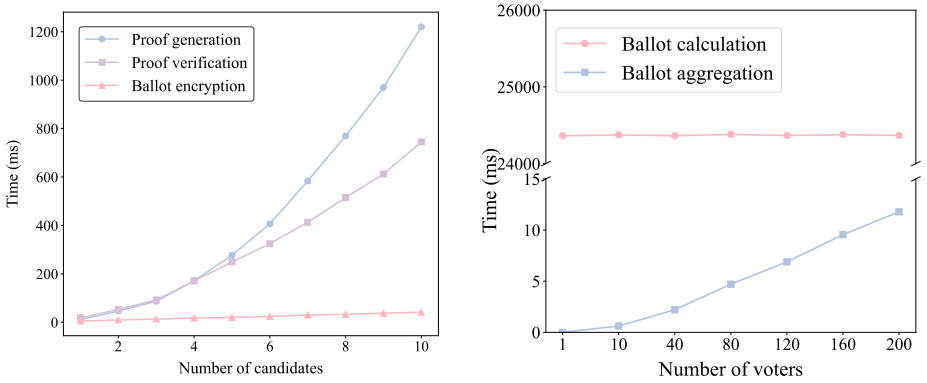


(a) Time cost of our protocol (b) Performance comparison

Fig. 5: Performance evaluation

(1) Off-chain Performance: We implement FP-BSS in Go and compile it using GoLand 2024. Each experiment is repeated 10 times with averaged results. We configure the system with $n = 3$ voters, $m = 3$ candidates, and time hardness parameter $t = 30,000,000$. Under this setting of the time hardness parameter, our system requires approximately 90 s to complete initialization. This time hardness parameter is adjustable, a larger value increases the time required to solve the time-lock puzzle, which ensures time-limited privacy. The protocol's time cost is shown in Fig. 5(a). Voters must register with the EA to obtain legal voting status, requiring 0.36 ms per registration. To cast a ballot, voters perform encryption requiring 13.14 ms. Additionally, voters generate range proofs, total proofs, and ballot signatures, with respective times of 84.98 ms, 1.71 ms, and 1.63 ms. Range proof generation constitutes the most computationally intensive operation and therefore consumes the most time. For ballot verification, a verifier must validate the corresponding proofs and signature, requiring 85.06 ms, 7.26 ms, and 1.11 ms respectively. Ballot tracing and aggregation require 2.19 ms and 0.11 ms respectively.

We present a performance comparison with [6], with results shown in Fig. 5(b). It can be observed that the scheme in [6] requires more time for ballot casting and verification. This efficiency improvement in our protocol arises from linking the random number used for ballot encryption to the order of the elliptic curve, which reduces computational costs for modular operations. Specifically, ballot casting in our protocol requires 13.14 ms, 84.98 ms, and 1.71 ms respectively, compared to 34.49 ms, 104.35 ms, and 13.87 ms for [6]. Similarly, ballot verification in our protocol requires 85.06 ms and 7.26 ms, compared to 112.02 ms and 13.12 ms for the referenced scheme.



(a) Time cost of cast and verification (b) Time cost of calculation

Fig. 6: Effects of different voters and candidates

We evaluate the time costs of ballot encryption, proof generation, and verification with different numbers of candidates. Based on actual voting application scenarios, we configure the system with up to 10 candidates and 200 voters to simulate realistic voting environments. As shown in Fig. 6(a), we set the number of candidates from 1 to 10, and as the number of candidates increases, the time required for these three steps also increases. We also evaluate the time costs of ballot aggregation and calculation with different numbers of voters. As shown in Fig. 6(b), we set the number of voters up to 200, and as the number of voters increases, the time required for aggregation also increases. When the number of voters is 200, the time required for ballot aggregation is 11.78 ms, and the time required for the calculation remains stable. This enables our system to maintain robust scalability even with progressively increasing voter populations.

(2) On-chain Performance: We deploy FP-BSS on a Hyperledger Fabric 2.2-based permissioned blockchain network, configured with two organizations and a single ordering node to evaluate its performance. Each experiment is repeated 10 times with averaged results. To better evaluate performance, we conduct comprehensive benchmarking of the smart contract using the Caliper framework under controlled experimental conditions. We use fixed load rate control at 5 transactions per second. We maintain consistency by following the off-chain set-

Table 4: Time Cost of Operations

Operation	Time Cost	Operation	Time Cost
BallotCast	0.143s	Trace	0.026s
Generate range proof	0.556s	Verify range proof	0.455s
Generate sum proof	0.053s	Verify sum proof	0.100s
Generate signature	0.029s	Verify signature	0.026s

tings mentioned above. As shown in Table 4, the BallotCast operation requires 0.143 s, while range proof generation, sum proof generation, and signature generation require 0.556 s, 0.053 s, and 0.029 s respectively. Range proof generation constitutes the most computationally intensive operation, consistent with prior analysis. The Trace operation completes in 0.026 s, while range proof verification, sum proof verification, and signature verification require 0.455 s, 0.100 s, and 0.026 s respectively. Similarly, range proof verification is the most complex operation, making it the most time-consuming verification task.

8 Conclusion

In this paper, we propose FP-BSS, a full privacy blockchain-based self-tallying score voting protocol. The protocol provides enhanced security. Specifically, we extend the conventional binary voting scheme to support score voting. We propose a novel full privacy ballot encryption scheme based on blinding factors to guarantee complete voter privacy. Also, we propose a new zero-knowledge proof system for encrypted ballot legitimacy verification. Our protocol satisfies the properties of full privacy, time-limited privacy, linkability, universal verifiability, and robustness. Experimental results from both on-chain and off-chain implementations demonstrate that our scheme maintains practical efficiency.

References

1. Y.-X. Kho, S.-H. Heng, J.-J. Chin, A review of cryptographic electronic voting, Symmetry 14 (5) (2022) 858.
2. S. Nakamoto, Bitcoin: A peer-to-peer electronic cash system, in: Decentralized Business Review, 2008, p. 21260.
3. Y. Yu, Y. Li, J. Tian, J. Liu, Blockchain-based solutions to security and privacy issues in the internet of things, IEEE Wirel. Commun. 25 (6) (2018) 12–18.
4. N. Kshetri and J. Voas, “Blockchain-enabled e-voting,” IEEE Softw., vol. 35, no. 4, pp. 95–99, Jul. 2018.
5. Yichao Lu, Huilin Li, Le Gao, Jiaxin Yu, Yong Yu, Hexing Su, Self-tallying e-voting with public traceability based on blockchain, Computer Standards & Interfaces, Volume 88, 2024, 103795, ISSN 0920-5489.
6. M. Miao, L. Tang, J. Li and X. Zhang, ”New Blockchain-Based Publicly Traceable Self-Tallying Voting Protocol,” in IEEE Transactions on Network Science and Engineering, vol. 11, no. 2, pp. 2034-2046, March-April 2024.

7. J. Huang, D. He, Y. Chen, M. K. Khan and M. Luo, "A Blockchain-Based Self-Tallying Voting Protocol With Maximum Voter Privacy," in IEEE Transactions on Network Science and Engineering, vol. 9, no. 5, pp. 3808-3820, 1 Sept.-Oct. 2022
8. T. Kohno, A. Stubblefield, A. D. Rubin, and D. S. Wallach, "Analysis of an electronic voting system," in Proc. IEEE Symp. Security Privacy, 2004, pp. 27–40.
9. A. Kiayias and M. Yung, "Self-tallying elections and perfect ballot secrecy," in Proc. Int. Workshop Public Key Cryptogr., 2002, pp. 141–158.
10. F. Hao, P. Y. Ryan, and P. Zieli ´nski, "Anonymous voting by two-round public discussion," IET Inf. Secur., vol. 4, no. 2, pp. 62–67, Jun. 2010.
11. P. McCorry, S. F. Shahandashti, and F. Hao, "A smart contract for boardroom voting with maximum voter privacy," in Financial Cryptography and Data Security, A. Kiayias, Ed., Cham, Switzerland: Springer, 2017, pp. 357–375.
12. D. Khader, B. Smyth, P. Ryan, F. Hao, A fair and robust voting system by broadcast, Lect. Note. Inform. (2012) 285–299.
13. Y. Lin, P. Zhang, Blockchain-based complete self-tallying E-voting protocol, in: 2019 Asia-Pacific Signal and Information Processing Association Annual Summit and Conference, 2019, pp. 47–52.
14. H. Li, Y. Li, Y. Yu, B. Wang, K. Chen, A blockchain-based traceable self-tallying E-voting protocol in AI era, IEEE Trans. Netw. Sci. Eng. 8 (2) (2020) 1019–1032.
15. Y. Li et al., "A blockchain-based self-tallying voting protocol in decentralized IoT," IEEE Trans. Dependable Secure Comput., vol. 19, no. 1, pp. 119–130, Jan./Feb. 2022.
16. L. Wang, H. Li, Y. Li, Y. Yu and X. Du, "Self-Tallying Voting with Blockchain in Wireless Network Environment," in IEEE Wireless Communications, vol. 31, no. 5, pp. 142-147, October 2024.
17. R.L. Rivest, A. Shamir, D.A. Wagner, Time-Lock Puzzles and Timed-Release Crypto, Massachusetts Institute of Technology, 2001.
18. A. Fiat and A. Shamir, "How to prove yourself: Practical solutions to identification and signature problems," in Proc. Conf. Theory Appl. Cryptographic Techn., 1986, pp. 186–194.
19. M. Mahmoody, T. Moran, and S. Vadhan, "Time-lock puzzles in the random oracle model," in Proc. Annu. Cryptol. Conf., 2011, pp. 39–50.
20. G. Malavolta and S. A. K. Thyagarajan, "Homomorphic time-lock puzzles and applications," in Proc. Annu. Int. Cryptol. Conf., 2019, pp. 620–649.
21. R. Cramer, I. Damgård, and B. Schoenmakers, "Proofs of partial knowledge and simplified design of witness hiding protocols," in Proc. Annu. Int. Cryptol. Conf., 1994, pp. 174–187.
22. I. Damgård, On σ-Protocols, in: Lecture Notes, University of Aarhus, Department for Computer Science, 2002, p. 84.
23. Manuel Blum, Paul Feldman, and Silvio Micali. 1988. Non-interactive zero-knowledge and its applications. In Proceedings of the twentieth annual ACM symposium on Theory of computing (STOC '88). Association for Computing Machinery, New York, NY, USA, 103–112.