Abstract—Voting is a means to agree on a collective decision based on available choices (e.g., candidates), where participants agree to abide by their outcome. To improve some features of e-voting, decentralized blockchain-based solutions can be employed, where the blockchain represents a public bulletin board that in contrast to a centralized bulletin board provides extremely high availability, censorship resistance, and correct code execution. A blockchain ensures that all entities in the voting system have the same view of the actions made by others due to its immutability and append-only features. The existing remote blockchain-based boardroom voting solution called Open Vote Network (OVN) provides the privacy of votes, universal & End-to-End verifiability, and perfect ballot secrecy; however, it supports only 2 choices and lacks recovery from stalling participants.

We present BBB-Voting, an equivalent blockchain-based approach for decentralized voting such as OVN, but in contrast to it, BBB-Voting supports 1-out-of-\(k\) choices and provides robustness that enables recovery from stalling participants. We make a cost-optimized implementation using an Ethereum-based environment respecting Ethereum Enterprise Alliance standards, which we compare with OVN and show that our work decreases the costs for voters by 13.5% in normalized gas consumption. Finally, we show how BBB-Voting can be extended to support the number of participants limited only by the expenses paid by the authority and the computing power to obtain the tally.

I. INTRODUCTION

Voting is an integral part of democratic governance, where eligible participants can cast a vote for their representative choice (e.g., candidate or policy) through a secret ballot. The outcome of voting announces a tally of votes. Voting is usually centralized and suffers from a single point of failure that can be manifested in censorship, tampering, and issues with availability of a service. Blockchain is an emerging decentralized technology that provides interesting properties such as decentralization, censorship-resistance, immutability of data, correct execution of code, and extremely high availability, which can be harnessed in addressing the existing issues of e-voting. A few blockchain-based e-voting solutions have been proposed in recent years, mostly focusing on boardroom voting [1, 2, 3, 4] or small-scale voting [5, 6, 3].

Decentralization was a desired property of e-voting even before invention of blockchains. For example, (partially) decentralized e-voting that uses the homomorphic properties of El-Gamal encryption was introduced by Cramer et al. [7]. It assumes a threshold number of honest election authorities to provide the privacy of vote. However, when this threshold is adversarial, it does not protect from computing partial tallies, making statistical inferences about it, or even worse—the vote choices of participants. A solution that removed trust in tallying authorities was for the first time proposed by Kiayias and Yung [8] in their privacy-preserving self-tallying boardroom voting protocol. A similar protocol was later proposed by Hao et al. [9], which was later extended to a blockchain environment by McCorry et al. [1] in their Open Vote Network (OVN). An interesting property of OVN is that it requires only a single honest voting participant to maintain the privacy of the votes. However, OVN supports only two vote choices (based on [9]), assumes no stalling participants, and requires expensive on-chain tally computation (limiting its scalability). The scalability of OVN was improved by Seifelnasr et al. [5], but retaining the limitation of 2 choices and missing robustness.

Our goal is to build a remote boardroom voting protocol that resolves these limitations and enables a straightforward extension to support scalability. Therefore, we introduce BBB-Voting, a blockchain-based boardroom voting system providing 1-out-of-\(k\) voting, while additionally offering a mechanism for resolution of faulty participants. Alike OVN, BBB-Voting also provides the maximum privacy of votes in the setting that outputs the full tally of votes (as opposed to tally-hiding protocols [10, 11]). Both OVN and BBB-Voting require the authority whose role is limited to registering participants and shifting the phases of the protocol. The communication between the participants and the blockchain is semi-synchronous; i.e., each participant is expected to execute certain actions within a given time frame. When all registered participants submit their votes, the result can be tallied by anybody and the correctness of the result is verified by the blockchain.

Contributions. We make the following contributions.

1) We present BBB-Voting, an approach for remote end-to-end verifiable privacy-preserving self-tallying 1-out-of-\(k\) boardroom voting on the blockchain (see § IV). In detail, we start with the voting protocol proposed by Hao et al. [9] that provides a low bandwidth requirements and computational costs but is limited to 2 vote choices. We extend this protocol to support \(k\) choices utilizing the 1-out-of-\(k\) proof verification proposed by Kiayias and Yung [8]. We accommodate this approach to run on the blockchain with Turing-complete smart contract capability, enabling on-chain zero-knowledge proof verification of blinded votes (and other proofs).
ii) We incorporate a robustness approach [12] into our protocol, which enables us to eliminate (even reoccurring) stalling (i.e., faulty) participants and thus finish voting without restarting the protocol (see § IV-B).

iii) We make two implementations, one based on discrete logarithm problem (DLP) for integers modulo $p$ and the second one based on the elliptic curve DLP. For both implementations we propose various optimizations reducing the costs imposed by the blockchain platform (§ V-A). Due to the optimizations, our implementation (with elliptic curve DLP) increases the number of participants fitting a single block by 9% in contrast to OVN [1] under the same assumptions, while it decreases the costs for voters by 13.5%.

iv) We outline a scalability extension of our work, enabling the number of participants to be limited only by the expenses paid by the authority to register participants and compute their multi-party computation (MPC) keys as well as the computing power to obtain the tally. For demonstration purposes, we evaluate its utility in the context of the voting that is a magnitude greater than the boardroom voting (i.e., up to 1000 participants) while preserving almost the same per-participant costs paid by the authority as without this extension (see § VI-A).

II. PRELIMINARIES

In this section, we describe voting terminology. We assume that the reader is familiar with blockchains and smart contracts.

An involved party refers to any stakeholder of the voting process and it covers all participants and the authority. A voting protocol is expected to meet several properties. A list of such properties appears in the works of Kiayias and Yung [8], Groth [13], and Cramer et al. [7].

1) Privacy of Vote: ensures the secrecy of the ballot contents [8]. Hence, a participant’s vote must not be revealed other than by the participant herself upon her discretion (or through the collusion of all remaining participants). Usually, privacy is ensured by trusting authorities in traditional elections or by homomorphic encryption in some decentralized e-voting solutions (e.g., [8], [9], [1], [5], [6]).

2) Perfect Ballot Secrecy: is an extension of the privacy of the vote. It implies that a partial tally (i.e., prior to the end of voting) is available only if all remaining participants are involved in its computation.

3) Fairness: ensures that a tally may be calculated only after all participants have submitted their votes. Therefore, no partial tally can be revealed to anyone before the end of the voting protocol [8].

4a) Universal Verifiability: any involved party can verify that all cast votes are correct and they are correctly included in the final tally [8].

4b) End-to-End (E2E) Verifiability: The verifiability of voting systems is also assessed by E2E verifiability [14], which involves cast-as-intended, recorded-as-cast, and tallied-as-recorded verifiability [15].

5) Dispute-Freeness: extends the notion of verifiability. A dispute-free [8] voting protocol contains built-in mechanisms eliminating disputes between participants; therefore, anyone can verify whether a participant followed the protocol. Such a scheme has a publicly-verifiable audit trail that contributes to the reliability and trustworthiness of the scheme.

6) Self-Tallying: once all the votes are cast, any involved party can compute the tally. Self-tallying systems need to deal with the fairness issues (see (3) above) because the last participant is able to compute the tally even before casting her vote. This can be rectified with an additional verifiable dummy vote [8].

7) Robustness (Fault Tolerance): the voting protocol is able to recover from faulty (stalling) participants, where faults are publicly visible and verifiable due to dispute-freeness [8]. Fault recovery is possible when all the remaining honest participants are involved in the recovery.

8) Resistance to Serious Failures: Serious failures are defined as situations in which voting results were changed either by a simple error or an adversarial attack. Such a change may or may not be detected. If detected in non-resistant systems, it is irreparable without restarting the entire voting [16].

III. SYSTEM MODEL & OVERVIEW

Our system model has the following actor/components: (1) a participant ($P$) who votes, (2) a voting authority ($VA$) who is responsible for validating the eligibility of $Ps$ to vote, their registration, and (3) a smart contract ($SC$), which collects the votes, acts as a verifier of zero-knowledge proofs, enforces the rules of voting, and verifies the tally.

A. Adversary Model

The adversary $A$ has bounded computing power, is unable to break used cryptographic primitives, and can control at most $t$ of $n$ participants during the protocol, where $t < n - 2$ and $n > 3$. Any $P$ under the control of $A$ can misbehave during the protocol execution. $A$ is also a passive listener of all communication entering the blockchain network but cannot block it or replace it with a malicious message since all transactions sent to the blockchain are authenticated by signatures of $Ps$ or $VA$. Finally, $VA$ is only trusted in terms of identity management, i.e., it performs identity verification of $Ps$ honestly, and neither censor any $P$ nor register any spoofed $P$. Nevertheless, no other trust in $VA$ is required.

IV. BBB-VOTING SCHEME

BBB-Voting scheme provides all properties mentioned in § II. Similar to OVN [1], BBB-Voting publishes the full tally at the output and uses homomorphic encryption to achieve privacy of votes and perfect ballot secrecy. In detail, we extend the protocol of Hao et al. [9] to support $k$ choices utilizing the 1-out-of-$k$ proof verification proposed by Kiayias and Yung [8], and we accommodate this approach to run on the blockchain. Additionally, we extend our protocol to support the robustness, based on Khader et al. [12], which enables the protocol to recover (without a restart) from faulty participants who did not submit their votes. As a consequence, robustness increases the resistance of our protocol to serious failures.

A. Base Variant

The base variant of BBB-Voting does not involve a fault recovery and is divided into five stages: registration (identity verification, key ownership verification, enrollment at $SC$), a setup (an agreement on system parameters, submission of ephemeral public keys), pre-voting (computation of MPC keys), voting (vote packing, blinding, and verification), and tally phases. All faulty behaviors of $Ps$ and $VA$ are subject to deposit-based penalties. In detail, $P$ who submitted her ephemeral key (in the setup phase) and then has not voted
within the timeout will lose the deposit. To achieve fairness, \( VA \) acts as the last \( P \) who submits a "dummy vote" with her ephemeral private key\(^1\) after all other \( P \)'s cast their vote (or upon the voting time expiration).

1) **Phase 1 (Registration)** \( VA \) first verifies the identity proof of each \( P \). For decentralized identity management (IDM), the identity proof is represented by the verifiable credentials (VC)\(^2\) signed by the issuer, while in a centralized IDM the identity proof is interactively provided by a third-party identity provider (e.g., Google). First, \( VA \) verifies the issuer’s signature on the identity proof. Next, \( VA \) challenges \( P \) to prove (using her VC) that she is indeed the owner of the identity. Further, each \( P \) creates her blockchain wallet address (i.e., the blockchain public key (PK)) and provides it to \( VA \). The \( VA \) locally stores a bijective mapping between a \( P \)'s identity and her wallet address.\(^3\) Next, \( VA \) enrolls all verified \( P \)'s by sending their wallet addresses to \( SC \).

2) **Phase 2 (Setup)** \( P \)'s agree on system parameters that are universal to voting – the parameters for voting are publicly visible on \( SC \) (deployed by \( VA \) in a transaction).

Therefore, any \( P \) may verify these parameters before joining the protocol. Note the deployment transaction also contains the specification of timeouts for all further phases of the protocol as well as deposit-based penalties for misbehavior of \( VA \) and \( P \)'s. The parameters for voting are set as follows:

1) \( VA \) selects a common generator \( g \in \mathbb{F}_p^* \). The value of \( p \) is chosen to be a safe prime, i.e., \( p = 2 \cdot q + 1 \), where \( q \) is a prime. A safe prime is chosen to ensure the multiplicative group of order \( p - 1 = 2 \cdot q \), which has no small subgroups that are trivial to detect.\(^4\) Let \( n < p - 1 \).

2) Any participant \( P_i \) is later permitted to submit a vote \( \{v_i | i \in \{1, 2, ..., k\} \} \) for one of \( k \) choices. This is achieved by selecting \( k \) independent generators \( \{f_1, ..., f_k\} \) in \( \mathbb{F}_p^* \) (one for each choice). These generators for choices should meet a property described by Hao et al.\(^5\) to preclude having two different valid tallies that fit Eq. 4:

\[
\sum_{i=1}^{k} a_i f_i \equiv 0 \pmod{p} \quad \text{(for choice } k \text{)}
\]

where \( m \) is the smallest integer such that \( 2^m > n \) (the number of participants).

**Ephemeral Key Generation & Committing to Vote.** Each \( P_i \) creates her ephemeral private key as a random number \( x_i \in \mathbb{F}_p^* \) and ephemeral public key as \( g^{x_i} \). Each \( P_i \) sends their ephemeral public key to \( SC \) in a transaction signed by her wallet, thereby, committing to submit a vote later.\(^6\)

Furthermore, \( P_i \) sends a deposit in this transaction, which can be retrieved back after the end of voting. However, if \( P_i \) does not vote within a timeout (or does not participate in a fault recovery (see § IV-B)), the deposit is lost, and it is split to the remaining involved parties. \( P \)'s who do not submit their ephemeral keys in this stage are indicating that they do not intend to vote; the protocol continues without them and they are not subject to penalties. Finally, each \( P \) obtains (from \( SC \)) the ephemeral public keys of all other verified \( P \)'s who have committed to voting. Ephemeral keys are one-time keys, and thus can be used only within one run of the protocol to ensure privacy of votes (other runs require fresh ephemeral keys).

3) **Phase 3 (Pre-Voting)** This phase represents multiparty computation (MPC), which is run to synchronize the keys among all \( P \) and achieve the self-tallying property. However, no direct interaction among \( P \) is required since all ephemeral public keys are published at \( SC \). The MPC keys are computed by \( SC \), when \( VA \) triggers the compute operation via a transaction. The \( SC \) computes and stores the MPC key for each \( P_i \) as follows:

\[
h = g^{y_i} = \prod_{j=1}^{i-1} g^{x_j} / \prod_{j=i+1}^{m} g^{x_j},
\]

where \( y_i = \sum_{j<i} x_j - \sum_{j>i} x_j \) and \( \sum_{j<i} x_j y_i = 0 \) (see Hao et al.\(^5\) for the proof). While anyone can compute \( g^{y_i} \), to reveal \( y_i \), all \( P \) \( P \) must either collude or solve the DLP for Eq. 2. As the corollary of Eq. 2, the protocol preserves

\(^{1}\)Privacy for a dummy vote is not guaranteed since it is subtracted.

\(^{2}\)Note that the address of \( P \) must not be part of identity proof - avoiding \( VA \) to possess a proof of identity to blockchain address mapping (see § VII-B).

\(^{3}\)We use modular exponentiation by repeated squaring to compute \( g^x \mod p \), which has a time complexity of \( O((\log x) \cdot (\log p^2)) \)\(^18\).

\(^{4}\)In contrast to OVN\(^1\) (based on the idea from \( [9] \)), we do not require \( P_i \) to submit ZKP of knowledge of \( x_i \) to \( SC \) since \( P_i \) may only lose by submitting \( g^{x_i} \) to which she does not know \( x_i \) (i.e., a chance to vote + deposit).
Fig. 2: ZKP of set membership for 1-out-of-\( k \) choices.

4) Phase 4 (Voting) In this phase, each \( P_i \) blinds and submits her vote to \( SC \). These steps must ensure the recoverability of the tally, vote privacy, and well-formedness of the vote. Vote privacy is achieved by multiplying the \( P_i \)'s blinded key with her vote choice. The blinded vote of the participant \( P_i \) is

\[
B_i = \begin{cases} 
    g^{x_i \cdot y_i} \cdot f_1 & \text{if } P_i \text{ votes for choice 1}, \\
    g^{x_i \cdot y_i} \cdot f_2 & \text{if } P_i \text{ votes for choice 2}, \\
    g^{x_i \cdot y_i} \cdot f_k & \text{if } P_i \text{ votes for choice } k.
\end{cases}
\]

The \( P \) sends her choice within a blinded vote along with a 1-out-of-\( k \) non-interactive zero-knowledge (NIZK) proof of set membership to \( SC \) (i.e., proving that the vote choice \( \in \{1, \ldots, k\} \)). We modified the approach proposed by Kiayias et al. [8] to the form used by Hao et al. [9], which is convenient for practical deployment on existing smart contract platforms. The verification of set membership using this protocol is depicted in Fig. 2, where \( P_i \) is a prover and \( SC \) is the verifier. Hence, \( SC \) verifies the correctness of the proof and then stores the blinded vote. In this stage, it is important to ensure that no re-voting is possible, which is to avoid any inference about the final vote of \( P \) in the case she would change her vote choice during the voting stage. Such a re-voting logic can be enforced by \( SC \), while user interface of the \( P \) should also not allow it. Moreover, to ensure fairness, \( VA \) acts as the last \( P \) who submits a dummy vote and her ephemeral private key.

5) Phase 5 (Tally) When the voting finishes (i.e., voting timeout expires or all \( P \)'s and \( VA \) cast their votes), the tally of votes received for each of \( k \) choices is computed off-chain by any party and then submitted to \( SC \). When \( SC \) receives the tally, it verifies whether Eq. 4 holds, subtracts a dummy vote of \( VA \), and notifies all \( P \) about the result. The tally is represented by vote counts \( c_{ij}, \forall i \in \{1, \ldots, k\} \) of each choice, which are computed using an exhaustive search fitting

\[
\prod_{i=1}^{n} B_i = \prod_{i=1}^{n} g^{x_i \cdot y_i} f = g^{\sum_{i=1}^{n} x_i y_i} f = f_1^{c_{11}} f_2^{c_{22}} \ldots f_k^{c_{kk}}. \quad (4)
\]

The maximum number of attempts is bounded by combinations with repetitions to \( \binom{n+k-1}{k-1} \). Although the exhaustive search of 1-out-of-\( k \) voting is more computationally demanding in contrast to 1-out-of-2 voting [1], [9], this process can be heavily parallelized. See time measurements in § V-B.

B. Variant with Robustness

We extend the base variant of BBB-Voting by a fault recovery mechanism. If one or more \( P \)'s stall (i.e., are faulty) and do not submit their blinded vote in the voting stage despite committing in doing so, the tally cannot be computed directly. To recover from faulty \( P \)'s, we adapt the solution proposed by Khader et al. [12], and we place the fault recovery phase immediately after the voting phase. All remaining honest \( P \)'s are expected to repair their vote by a transaction to \( SC \), which contains key materials shared with all faulty \( P \)'s and their NIZK proof of correctness. \( SC \) verifies all key materials with proofs (see Fig. 3), and then they are used to invert out the counter-party keys from a blinded vote of an honest \( P \) who sent the vote-repairing transaction to \( SC \). Even if some of the honest (i.e., non-faulty) \( P \)’s would be faulty during the recovery phase (i.e., do not submit vote-repairing transaction), it is still possible to recover from such a state by repeating the next round of the fault recovery protocol. For this reason, we use only 1-out-of-2 voting [1], [9], and this process can be heavily parallelized. See time measurements in § V-B.

V. IMPLEMENTATION & EVALUATION

We selected the Ethereum-based environment for evaluation due to its widespread adoption and open standardized architecture (driven by the Enterprise Ethereum Alliance [19]), which is incorporated by many blockchain projects. We implemented \( SC \) components in Solidity, while \( VA \) and \( P \) components were implemented in Javascript as testing clients of the truffle project. Executing smart contracts over blockchain, i.e., performing computations and storing data, has its costs. In Ethereum Virtual Machine (EVM), these costs are expressed by the level of execution complexity of particular instructions, referred to as gas. In this section, we analyze the costs imposed by our approach, perform a few optimizations, and compare the costs with OVN [1]. In the context of this work, we assume...
In the ECC, in contrast to the version with affine coordinates. L ZKP requires three affine in contrast to the \( kP \) can be pre-computed off-chain, while \( kP \) is computed on-chain. However, to leverage the full potential of the doubled simultaneous multiplication, one must have the expression \( kP + lQ \), which is often not the case. In our case, we modified the check at \( SC \) to fit this form. Alike the vote submission, this optimization can be applied in vote repair. We depict the performance improvement brought by this optimization as series “ECC-Jacobi (smal)” in Fig. 5.

(4) Pre-Computation of Modular Inversions. Each affine transformation in the vote submission contains one operation of modular inversion – assuming previous optimizations, ZKP verification of one item in 1-out-of-\( k \) ZKP requires three affine transformations (e.g., for \( k = 5 \), it is 15). Similarly, the ZKP verification of correctness in the repair vote requires two affine transformations per each faulty participant submitted. The modular inversion operation runs the extended Euclidean algorithm, which imposes non-negligible costs. However, all modular inversions can be pre-computed off-chain, while only their verification can be made on-chain (i.e., modular multiplication), which imposes much lower costs. We depict
the impact of this optimization as “ECC-Jacobi (smul+modi)” series in Fig. 5 and “…modi…” series in Fig. 4. In the result, it has brought 5% savings of costs in contrast to the version with the simultaneous multiplication.

B. Tally Computation

In Tab. I, we provide time measurements of tally computation through the entire search space on 1 core vs. all cores of the i7-10510U CPU laptop.8 We see that for $n \leq 100$ and $k \leq 6$, the tally can be computed even on a commodity PC in a reasonable time. However, for higher $n$ and $k$, we recommend using a more powerful machine or distributed computation across all PEs. One should realize that our measurements correspond to the upper bound, and if some ranges of tally frequencies are more likely than others, they can be processed first – in this way, the computation time can be significantly reduced. Moreover, we emphasize that an exhaustive search for tally computation is not specific only to our scheme but to homomorphic-encryption-based schemes providing perfect ballot secrecy and privacy of votes (e.g., [8], [9], [1]).

C. Cost Comparison

In Tab. II, we made a cost comparison of BBB-Voting (using ECC) with OVN [1], where we assumed two choices and 40 participants (the same setting as in [1]). We see that the total costs are similar but BBB-Voting improves $P$’s costs by 13.5% and $VA$’s cost by 0.9% even though using more complex setting that allows 1-out-of-k voting. We also emphasize that the protocol used for vote casting in BBB-Voting contains more operations than OVN but regardless of it, the costs are close to those of OVN, which is mostly caused by the proposed optimizations.9 Next, we found that OVN computes tally on-chain, which is an expensive option. In contrast, BBB-Voting computes tally off-chain and $SC$ performs only verification of its correctness, which enables us to minimize the cost of this operation. Another gas saving optimization of BBB-Voting in contrast to OVN (and Hao et al. [9]) is that we do not require voters to submit ZKP of knowledge of $x_i$ in $g^{x_i}$ during the registration phase to $SC$ since $P_i$ may only lose by providing incorrect ephemeral public key $g^{x_i}$ – she might lose the chance to vote and her deposit. Finally, we note that we consider the deployment costs of our $SC$ equal to 4.8M units of gas; however, our $SC$ implementation contains a few auxiliary view-only functions for a pre-computation of modular inverses, with which, the deployment costs would increase to 7.67M due to code size. Nevertheless, these operations can be safely off-chained and we utilized them on-chain only for simplicity.

VI. DISCUSSION OF EXTENSIONS

In this section, we discuss the extensions addressing the scalability and performance limitations of BBB-Voting.

A. Scalability Limitation & Extension

The limitation of BBB-Voting (like in OVN) is a lack of scalability, where the block gas limit might be exceeded with a high number of PEs. Therefore, we primarily position our solution as boardroom voting; however, we will show in this section that it can be extended even to larger voting. Our voting protocol (see § IV-A) has a few platform-specific bottlenecks. The first bottleneck is in the setup phase, where $VA$ submits the wallet addresses of all PEs to SC in a single transaction, which might exceed the block gas limit when the number of participants $n > 201$. The second bottleneck occurs in the pre-voting phase, where $VA$ calls the function of $SC$ to compute all MPC keys; exceeding the block gas limit occurs when the number of participants $n > 135$. The next bottleneck occurs in the voting phase, where voters submit their blinded votes together with 1-out-of-$k$ ZKP, exceeding the block gas limit when the number of choices $k > 7$. The last bottleneck occurs in the fault recovery phase and the block gas limit is exceeded when the number of simultaneously faulty participants $f > 9$.

| Gas Paid by | OVN | BBB-Voting |
|-------------|-----|------------|
| Deployment of Voting $SC$ | $VA$ 3.78M | 4.8M |
| Deployment of Cryptographic $SC$s | $VA$ 2.44M | 2.15M |
| Enroll voters | $VA$ 2.38M | (1.22M+0.93M) |
| Submit Ephemeral PK | $P$ 0.76M | 0.15M |
| Cast Vote | $P$ 2.50M | 2.72M |
| Tally | $VA$ (or $P$) 0.75M | 0.39M |
| Total Costs for $P$ | 3.26M | 2.87M |
| Total Costs for $VA$ | 9.35M | 9.27M |

Tab. II: A normalized cost comparison of BBB-Voting with OVN for $n = 40$ and $k = 2$.

---

8In some cases we estimated the time since we knew the number of attempts.
9To verify 1-out-of-$k$ ZKP in vote casting, BBB-Voting computes $5 \cdot k$ multiplications and $3 \cdot k$ additions on the elliptic curve – i.e., 10 multiplications and 6 additions for $k = 2$. In contrast, OVN computes only 8 multiplications and 5 additions for $k = 2$. 

Fig. 5: Vote submission and vote repair (i.e., fault recovery) with various optimizations.
mitigations. The security of our scheme relies on well-known cryptographic primitives under their standard assumptions.

A. Properties of Voting

1) Privacy in BBB-Voting requires at least 3 Ps, out of which at least 2 are honest (see § IV-A). Privacy in BBB-voting is achieved by blinding votes using ElGamal encryption [31], whose security is based on the decisional Diffie-Hellman assumption. Unlike the conventional ElGamal algorithm, a decryption operation is not required to unblind the votes. Instead, we rely on the self-tallying property of our voting protocol. The ciphertext representing a blinded vote is a tuple \((c_1, c_2)\), where \(c_1 = g^{xy} \cdot f\) and \(c_2 = g^y\), where the purpose of \(c_2\) is to assist with the decryption. Decryption involves computing \((c_2)^{-2} \cdot c_1\) to reveal \(f\), which unambiguously identifies a vote choice. As a result, the blinding operation for participant \(P_i\) in Eq. 3 is equivalent to ElGamal encryption involving the computation of \(c_1\) but not the decryption component \(c_2\). Furthermore, the blinding keys are ephemeral and used exactly once for encryption (i.e., blinding) of the vote within a single run of voting protocol\(^{11}\) – i.e., if the protocol is executed correctly, there are no two votes \(f_1\) and \(f_m\) encrypted with the same ephemeral blinding key of \(P_i\), such that

\[
\frac{(g^{xy} \cdot f_1)}{(g^{xy} \cdot f_m)} = \frac{f_1}{f_m},
\]

from which the individual votes could be deduced. For the blockchain-specific privacy analysis, see also § VII-B.

2) Ballot Secrecy. It is achieved by blinding the vote using ElGamal homomorphic encryption [7], and it is not required to possess a private key to decrypt the tally because of the self-tallying property \((g^{\sum y} = 1\). Therefore, given a homomorphic encryption function, it is possible to record a sequence of encrypted votes without being able to read the votes choices. However, if all \(P_s\) are involved in the recovery of a partial tally consisting of a recorded set of votes, these votes can be unblinded (as allowed by ballot secrecy). Even a subset of \(n - 2 P_s\) who have already cast their votes cannot recover a partial tally that reveals their vote choices because of the self-tallying property \((g^{\sum y} = 1\) has not been met.

3) Fairness. If implemented naively, the last voting \(P\) can privately reveal the full tally by solving Eq. 4 before she casts her vote since all remaining blinded votes are already recorded on the blockchain (a.k.a., the last participant conundrum). This can be resolved by \(VA\) who is required to submit the final dummy vote including the proof of her vote choice, which is later subtracted from the final tally by \(SC\).\(^{13}\)

4a) Universal Verifiability. Any involved party can check whether all recorded votes in the blockchain are correct and are correctly included in the final tally [8]. Besides, the blinded votes are verified at \(SC\), which provides correctness of its execution and public verifiability, relying on the honest majority of the consensus power in the blockchain (see § VII-B).

4b) E2E Verifiability. To satisfy E2E verifiability [2]: (I) each \(P\) can verify whether her vote was cast-as-intended and

---

\(^{10}\)E.g., for \(n = 10000, k = 2\) (and \(k = 4\), it takes 0.15s (and \(\sim 4\)h) to obtain the tally on a commodity PC with 8 cores, respectively.

---

VII. Security Analysis

We first analyze security of BBB-Voting with regard to the voting properties specified in § II. Next, we analyze blockchain-specific security & privacy issues and discuss their

---

\(^{11}\)As a consequence, BBB-Voting can be utilized in a repetitive voting [32] with a limitation of a single vote within an epoch.

\(^{12}\)Note that at least 2 Ps are required to be honest (see § III-A).

\(^{13}\)Note that if \(VA\) were not to execute this step, the fault recovery would exclude \(VA\)’s share from MPC keys, and the protocol would continue.
recorded-as-cast, (II) anyone can verify whether all votes are tallied-as-recorded. BBB-Voting meets (I) since each $P$ can locally compute her vote choice (anytime after) and compare it against the one recorded in the blockchain. BBB-Voting meets (II) since $SC$ executes the code verifying that the submitted tally fits Eq. 4 that embeds all recorded votes.

(5) Dispute-Freeness. Since the blockchain acts as a tamper-resistant bulletin board (see § VII-B), and moreover it provides correctness of code execution (i.e., on-chain execution of verification checks for votes, tally, and fault recovery shares) and verifiability, the election remains dispute-free under the standard blockchain assumptions about the honest majority and waiting the time to finality.

(6) Self-Tallying. BBB-Voting meets this property since in the tally phase of our protocol (and anytime after), all cast votes are recorded in $SC$; therefore any party can use them to fit Eq. 4, obtaining the final tally.

(7) Robustness (Fault Tolerance). BBB-Voting is robust since it enables to remove (even reoccurring) stalling $Ps$ by its fault recovery mechanism (see § IV-B). Removing of stalling $Ps$ involves $SC$ verifiability of ZKP submitted by $Ps$ along with their counter-party shares corresponding to stalling $Ps$.

(8) Resistance to Serious Failures. The resistance of BBB-Voting to serious failures relies on the integrity and append-only features of the blockchain, which (under its assumptions § VII-B) does not allow the change of already cast votes.

B. Blockchain-Specific Aspects and Issues

In the following, we focus on the most important blockchain-specific aspects and issues related to BBB-Voting.

(1) Bulletin Board vs. Blockchains. The definition of a bulletin board [8] assumes its immutability and append-only feature, which can be provided by blockchains that moreover provide correct execution of code. CAP theorem [33] enables a distributed system (such as the blockchain) to select either consistency or availability during the time of network partitions. If the system selects consistency (e.g., Algorand [34]), BFT-based blockchains such as [35], it stalls during network partitions and does not provide liveness (i.e., the blocks are not produced) but provides safety (i.e., all nodes agree on the same blocks when some are produced). On the other hand, if the system selects availability (e.g., Bitcoin [36], Ethereum [37]), it does not provide safety but provides liveness, which translates into possibility of creating accidental forks and eventually accepting one as valid. Many public blockchains favor availability over consistency, and thus do not guarantee immediate immutability. Furthermore, blockchains might suffer from malicious forks that are longer than accidental forks and are expensive for the attacker. Usually, their goal is to execute double-spending or selfish mining [38], violating the assumptions of the consensus protocol employed – more than 51% / 66% of honest nodes presented in PoW / BFT-based protocols. To prevent accidental forks and mitigate malicious forks in liveness-favoring blockchains, it is recommended to wait for a certain number of blocks (a.k.a., block confirmations). Another option to cope with forks is to utilize safety-favoring blockchains (e.g., [34], [35]).

Considering BBB-Voting, we argue that these forks are not critical for the proposed protocol since any transaction can be resubmitted if it is not included in the blockchain after a fork. Waiting for the time to finality (with a potential resubmission) can be done as a background task of the client software at $Ps$’ devices, so $Ps$ do not have to wait. Finally, we emphasize that the time to finality is negligible in contrast to timeouts of the protocol phases; therefore, there is enough time to make an automatic resubmission if needed.

(2) Privacy of Votes. In BBB-Voting, the privacy of vote choices can be “violated” only in the case of unanimous voting by all $Ps$, assuming $A$ who can link the identities of $Ps$ (approximated by their IP addresses) to their blockchain addresses by passive monitoring of network traffic. However, this is the acceptable property in the class of voting protocols that provide the full tally of votes at the output, such as BBB-Voting and other protocols (e.g., [1], [8], [9], [39], [4]). Moreover, $A$ can do deductions about the probability of selecting a particular vote choice by $Ps$. For example, in the case that the majority $m$ of all participants $n$ voted for a winning vote choice, then $A$ passively monitoring the network traffic can link the blockchain addresses of $Ps$ to their identities (i.e., IP addresses), and thus $A$ can infer that each $P$ from the group of all $Ps$ cast her vote to the winning choice with the probability equal to $\frac{m}{n} > 0.5$. However, it does not violate the privacy of votes and such an inferring is not possible solely from the data publicly stored at the blockchain since it stores only blinded votes and blockchain addresses of $Ps$, not the identities of $Ps$. To mitigate these issues, $Ps$ can use anonymization networks or VPN services for sending transactions to the blockchain. Moreover, neither $A$ nor $VA$ can provide the public with the indisputable proof that links $Ps$’ identity to her blockchain address.

(3) Privacy of Votes in Larger Voting. The privacy issue of unanimous and majority voting (assuming $A$ with network monitoring capability) are less likely to occur in the larger voting than boardroom voting since the voting group of $Ps$ is larger and potentially more divergent. We showed that BBB-Voting can be extended to such a large voting by integrity-preserving batching in § VI-A. We experimented with batching up to 1000 $Ps$, which is a magnitude greater voting than the boardroom voting. We depict the gas expenses paid by $VA$ (per $P$) in Fig. 6, where we distinguish various batch sizes. In sum, the bigger the batch size, the lower the price per $P$.

VIII. RELATED WORK

In this section, we briefly survey existing paradigms in e-voting and describe a few blockchain-based e-voting approaches. In particular, we focus on remote voting approaches (with sufficient specification), which we compare in Tab. III.

E-Voting Paradigms. Utilization of mix-nets that shuffle the votes to break the map between $Ps$ and their votes was proposed by Chaum [40]. Benaloh and Fischer [41] were among the first who showed a paradigm shift from anonymizing $Ps$ to providing privacy of the vote. Cramer et al. [7] present a model where all votes are sent to a single combiner, utilizing homomorphic properties of the ElGamal cryptosystem [31]. Using bulletin board and zero-knowledge proofs allow their protocol to be universally verifiable. The work of Kiayias and Yung [8] converts this scheme into a self-tallying combiner supporting 1-out-of-2 choices; further, the authors outline an extension of their base protocol to support 1-out-of-k choices. Hao et
al. [9] improve upon the self-tallying protocol by proposing a simple general-purpose two-round voting protocol for 1-out-of-2 choices with low bandwidth requirements and computational costs. Khader et al. [12] take it a step further by adding fairness and robustness properties. Groth [13] introduces an anonymous broadcast channel with perfect message secrecy leveraged in his voting protocol that is simpler and more efficient than [8]. However, it requires sequential voting, where each voter has to download a fresh state of the bulletin board before voting.

Zagorski et al. [42] propose Remotegrity that is based on Scanintegrity [43] ballots mailed to voters, allowing them remotely vote and verify that their ballots were correctly posted to the bulletin board and at the same time providing protection against malware in clients. Another direction (e.g., [10], [11]) focuses on the tally-hiding [44], [45] property that enable to reveal only the best $m$ candidates.

**Location.** Voting systems can be classified by the physical location where the vote is cast. Some schemes allow P's to submit a vote from their devices (i.e., remote voting), e.g., [46], [47], [42], and blockchain-based [48], [49]. Others systems require voting to be carried out at a designated site (a.k.a., supervised voting), e.g., [43], [50], [51], [52], and blockchain-based [53], [54], [55].

**A. Blockchain-Based E-Voting**

We extend the categorization of (remote) blockchain-based voting [4], and we focused on smart contract-based systems.

**(1) Voting Systems Using Smart Contracts.** McCorry et al. [1] proposed OVN, a self-tallying voting protocol (basing on [9]) that provides vote privacy and supports two vote choices. OVN is implemented as Ethereum SC and is suitable for boardroom voting. It does not provide robustness and expensive tally computation is made by SC. In contrast, BBB-Voting performs only tally verification in SC, while it is computed off-chain. Similar approach basing on [9] was proposed by Li et al. [3], who further provided robustness from [12]. Seifelnasr et al. [5] aimed to increase the scalability of OVN by off-chaining tally computation and registration at $VA$ in a verifiable way. Due to the higher costs imposed by storing data on SC, they compute the Merkle tree of voter identities and store only its root hash at SC. Their approach requires only a single honest $P$ to maintain the protocol’s security by enabling her to dispute the incorrect tally submitted to SC. The scalability technique proposed in this paper is orthogonal to us, and it can be combined with our techniques (see § VI-A) to optimize on-chain costs. Yu et al. [4] employ ring signature to ensure that the ballot is from one of the valid choices, and they achieve scalability by linkable ring signature key accumulation. Their approach provides receipt-freeness under the assumption of trusted $VA$. However, due to receipt-freeness, this approach does not provide E2E verifiability. Killier et al. [39] present an E2E verifiable remote voting scheme with two vote choices. The authors employ threshold cryptography for achieving robustness using a scheme similar to Shamir secret sharing. However, it supports only integers up to a 256 bits (i.e., a size of the EVM word), which is far below a minimal secure length. Matile et al. [56] proposed a voting system providing cast-as-intended (but neither E2E nor universal) verifiability. Their system uses ElGamal encryption based on DLP with integers modulo $p$. Since existing blockchains support natively only up to 256-bit security for this DLP, the authors create the custom blockchain with sufficient security. Dagher et al. [6] proposed BroncoVote, a voting system that preserves vote privacy by homomorphic encryption (i.e., Paillier cryptosystem) – the authors off-chain all cryptographic operations to a trusted server (without verifiability), which introduces a vulnerability. Kostal et al. [57] propose voting system, in which $VA$ serves as a trusted key generator that distributes private keys for homomorphic encryption to voters, enabling them to resolve robustness issues at the cost of putting a trust into $VA$.

**(2) Voting Systems Using Cryptocurrency.** Zhao and Chan [58] propose a privacy-preserving voting system with 1-out-2 choices based on Bitcoin, which uses a lottery-based approach with an off-chain distribution of voters’ secret random numbers with their ZKPs. The authors use deposits incentivizing $P$s to comply with the protocol; however, a malicious P can sabotage the voting by refusing to vote or vote in a wrong order. Tarasov and Tewari [59] proposed a voting system based on Zcash. The privacy of vote is ensured by the z-address that preserves unlinkability. The correctness of the voting is guaranteed by the trusted $VA$ and the candidates. If $VA$ is compromised, double-voting or tracing the source of the ballot (violating privacy) is possible. Liu and Wang [60] propose a conceptual voting approach based on blind signatures with 2 vote choices. Blockchain is utilized only for (auditable) sending of the messages among parties. However, despite using blind signatures, $P$s send their vote to blockchain in plain-text.

| Approach | Privacy of Vote | Perfect Ballot Secrecy | Fairness | Self-Tallying | Robustness | Uses Blockchain | E2E Verifiability | Uni. Verifiability | Open Source | Choices |
|----------|----------------|------------------------|----------|--------------|------------|----------------|-------------------|-------------------|-------------|---------|
| Hao et al. [9] | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | 2 |
| Khader et. [12] | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | 2 |
| Kayiwas and Yung [8] | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | 2/k |
| McCorry et. [1] (OVN) | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | 2 |
| Seifelnasr et al. [5] | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | 2 |
| Li et al. [3] | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | k |
| Baudron et al. [61] | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | k |
| Groth [13] | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | k |
| Adida [47] (Helios) | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | k |
| Matile et al. [56] (CaV) | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | k |
| Killer [39] (Protonum) | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | 2 |
| Dagher et al. [6] | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | k |
| (BroncoVote) | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | k |
| Kostal et al. [57] | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | k |
| Zagorski et al. [42] | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | k |
| (Remotegrity) | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | k |
| Yu et al. [4] | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | k |
| BBB-Voting | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | k |

Tab. III: A comparison of various remote voting protocols. *Assuming a trusted $VA$.

IX. Conclusion

In this paper, we proposed BBB-Voting, a 1-out-of-$k$ blockchain-based boardroom voting solution that supports fault tolerance. We made two variants of full implementation on EVM: one based on ECC DLP and the other one based on DLP for integer modulo $p$. We showed that only ECC variant is feasible in the real settings of public blockchains. We performed several cost optimizations and discussed further improvements concerning costs and scalability, where we made a proof-of-concept implementation with up to 1000 voters. Finally, we compared our solution with OVN and results indicate that BBB-Voting reduces the costs for voters and the authority by 13.5% and 0.9%, respectively.
REFERENCES

[1] P. McCorry et al., “A Smart Contract for Boardroom Voting with Maximum Voter Privacy,” in Financial Cryptography. Springer, 2017, pp. 357–375.

[2] S. Panja and B. K. Ray, “A secure end-to-end verifiable e-voting system using zero knowledge based blockchain,” ICMR Cryptology ePrint Archive, vol. 2018, p. 466, 2018.

[3] Y. Li et al., “A blockchain-based self-tallying voting protocol in decentralized IoT,” IEEE TDSC, pp. 1–1, 2020.

[4] B. Yu et al., “Platform-independent secure blockchain-based voting system,” in International Conference on Information Security. Springer, 2018, pp. 369–386.

[5] M. Seifelnasr et al., “Scalable open-vote network on ethereum,” in Financial Cryptography and Data Security, ser. LNCS, vol. 12063. Springer, 2020, pp. 436–450.

[6] G. Dagher et al., “BrontoVote: Secure Voting System using Ethereum’s Blockchain,” in ICICSP’18. SCITEPRESS, 2018, pp. 96–107.

[7] R. Cramer et al., “A secure and optimally efficient multi-authority election scheme,” in EUROCRYPT ’97, Konstanz, Germany, vol. 1233. New York City: Springer, 1997, pp. 103–118.

[8] A. Kiayias and M. Yung, “Self-tallying elections and perfect ballot secrecy,” in Public Key Cryptography, D. Naccache and P. Paillier, Eds. Berlin, Heidelberg: Springer Berlin Heidelberg, 2002, pp. 141–158.

[9] F. Hao et al., “Anonymous voting by two-round public discussion,” IET Information Security, vol. 4, no. 2, pp. 62–67, 2010.

[10] R. Küsters et al., “Ordinos: A verifiable tally-hiding e-voting system,” in EuroS&P. IEEE, 2020, pp. 216–235.

[11] N. Huber et al., “Kyvos: Publicly tally-hiding verifiable e-voting,” in ACM SIGSAC Conference on CCS, 2022, pp. 1443–1457.

[12] D. Khader et al., “A fair and robust voting system by broadcast,” in (EVOITE 2012), July 11-14, 2012, Castle Hofen, Bregenz, Austria, Bonn, Germany: GI, 2012, pp. 285–299.

[13] J. Groth, “Efficient maximal privacy in boardroom voting and anonymous broadcast,” in Financial Cryptography. Springer, 2004, pp. 90–104.

[14] J. Benaloh et al., “End-to-end verifiability,” arXiv preprint arXiv:1504.03778, 2015.

[15] H. Jonker et al., “Privacy and verifiability in voting systems: Methods, developments and trends,” Computer Science Review, vol. 10, pp. 1–30, 2013.

[16] S. Park et al., “Going from bad to worse: from internet voting to blockchain voting,” Journal of Cybersecurity, vol. 7, no. 1, 2021.

[17] W3C Working Group, “Verifiable credentials use cases,” [Online], 2019.

[18] N. Koblitz, A course in number theory and cryptography, 2nd Edition, ser. Graduate texts in mathematics. Springer, 1994, vol. 114.

[19] entathionallie, “Enterprise Ethereum Alliance,” [Online], 2017.

[20] Witnet Team, “elliptic-curve-solidity,” [Online], 2019.

[21] entethalliance.org, “Enterprise Ethereum Alliance,” [Online], 2017.

[22] Nathan Reiff, “What Is Polygon (MATIC)? Definition, Strengths, and Weaknesses,” [Online], 2022.

[23] Zerocoin Team, “Big number library for solidity,” [Online], 2019.

[24] I. Stan and I. Homoliak, “SBvote: Scalable Self-Tallying Blockchain-Based Voting,” in Trustworthy, and performant smart contracts,” in Blockchain-Based Voting,” in ICSICS. Berlin, Heidelberg: Springer-Verlag, 2016, pp. 223–240.

[25] F. Zagórski et al., “Remotegrity: Design and use of an end-to-end verifiable remote voting system,” in ACNS. Springer, 2013, pp. 441–457.

[26] D. Chaum et al., “Scantegrity: End-to-end voter-verifiable optical-scan voting,” IEEE Security & Privacy, vol. 6, no. 3, pp. 40–46, 2008.

[27] J. Benaloh, “Improving privacy in cryptographic elections (technical report),” Tech. Rep., 1986.

[28] A. Hevia and M. Kiwi, “Electronic jury voting protocols,” in LATIN ’02. Springer, 2002, pp. 415–429.

[29] A. Kiayias et al., “An internet voting system supporting user privacy,” in ACSAC. IEEE, Dec 2006, pp. 165–174.

[30] B. Adida, “Helios: Web-based open-audit voting,” in USENIX Security. USENIX Assoc., 2008, pp. 335–348.

[31] Tivi, “Tivi voting,” [Online], 2017.

[32] D. Sandler et al., “Votebox: A tamper-evident, verifiable electronic voting system,” in USENIX Security. USENIX Assoc., 2008, pp. 349–364.

[33] Bell et al., “Star-vote: A secure, transparent, auditable, and reliable voting system,” in EVT/WOTE. USENIX Assoc., 2013, pp. 18–37.

[34] S. F. Shahandashti and F. Hao, “Dre-ip: A verifiable e-voting scheme without tallying authorities,” in ESORICS. Berlin, Heidelberg: Springer-Verlag, 2016, pp. 223–240.

[35] M. Soud et al., “Trustvote: on elections we trust with distributed ledgers and smart contracts,” in IEEE BRAINS. IEEE, 2020, pp. 176–183.

[36] F. Hjalmarsdotter et al., “Blockchain-based e-voting system,” in IEEE CLOUD. IEEE, 2018, pp. 983–986.

[37] Agrora, “Bringing voting systems into the digital age,” https://www.agora.vote/ , 2018.

[38] R. Mattile et al., “CaV: Cast-as-Intended Verifiability in Blockchain-based Voting,” in ICBC’19, Seoul, Korea (South), May 14-17, 2019. Washington, DC: IEEE, 2019, pp. 24–28.

[39] K. Koitl et al., “Blockchain e-voting done right: Privacy and transparency with public blockchain,” in ICSESS. IEEE, 2019, pp. 592–595.

[40] Z. Zhao and T-H. H. Chan, “To vote privately using Bitcoin,” in ICICS. Springer, 2015, pp. 82–96.

[41] P. Tarasov and H. Tewari, “Internet voting using Zcash,” Cryptology ePrint Archive, 2017.

[42] Y. Liu and Q. Wang, “An e-voting protocol based on blockchain,” Cryptology ePrint Archive, 2017.

[43] O. Baudron et al., “Practical multi-candidate election system,” in PODC ’01. New York, NY, USA: ACM, 2001, pp. 274–283.