B-Privacy: Defining and Enforcing Privacy in Weighted Voting

Samuel Breckenridge* *Cornell Tech, IC3*
Dani Vilardell* *Cornell Tech, IC3*
Andrés Fábrega *Cornell Tech, IC3*
Amy Zhao *Ava Labs, IC3*
Patrick McCorry *Arbitrum Foundation*
Ari Juels *Cornell Tech, IC3*

***Abstract*—In traditional, one-vote-per-person voting systems, *privacy* equates with ballot secrecy: voting tallies are published, but individual voters' choices are concealed.**

Voting systems that weight votes in proportion to token holdings, though, are now prevalent in cryptocurrency and web3 systems. We show that these *weighted-voting* systems overturn existing notions of voter privacy. Our experiments demonstrate that even with secret ballots, publishing raw tallies often reveals voters' choices.

Weighted voting thus requires *a new framework for privacy*. We introduce a notion called *B-privacy* whose basis is *bribery*, a key problem in voting systems today. B-privacy captures the *economic cost* to an adversary of bribing voters based on revealed voting tallies.

We propose a mechanism to boost B-privacy by noising voting tallies. We prove bounds on its tradeoff between B-privacy and transparency, meaning reported-tally accuracy. We show experimentally across 2,503 proposals in 27 Decentralized Autonomous Organization (DAOs) that, with minimal transparency degradation, our mechanism raises B-privacy by a geometric mean factor of $3.5\times$.

Our work offers the first principled, practical, systemic guidance for weighted-voting systems, complementing existing approaches that focus on ballot secrecy.

1 Introduction

In traditional one-person-one-vote systems, privacy equates with ballot secrecy: aggregate tallies are published but individual choices remain hidden [2, 18, 38, 40].

In this work, we show that *weighted voting* fundamentally alters this picture of privacy in voting systems. Long used for share-based voting in corporate governance [28], weighted voting has also become the predominant tool for governance in blockchain protocols and DAOs [7] as well as for delegated voting in proof-of-stake consensus [25].

*These authors contributed equally to this work.

Unlike traditional voting systems that assign equal weight to each vote, weighted systems allocate influence proportionally to token (or share) ownership. We show that this proportionality creates new privacy risks: even when individual ballots are hidden, the tallies themselves often reveal how participants voted.

A simple example illustrates the problem.

Example 1 (Tally leakage)**.** *Alice possesses* 1.2 *voting weight, while all other voters possess exactly* 1 *voting weight. In a weighted tally—e.g., weight 1507 voting* yes *vs. weight 2510.2 voting* no*—if one choice has a .2 fractional part,* ***it must correspond to Alice's choice****.*

This a toy example. In this work, however, we conduct experiments on 3,844 recorded proposal votes across 31 DAOs. (DAOs today lack persistent ballot secrecy, but are advancing toward it [33, 34].) We show that, even with hypothetical ballot secrecy, weighted tallies in the DAOs under study *reveal a significant fraction of individual voters' choices*.

Addressing weighted-voting privacy. Weighted-voting systems thus require a *new privacy framework*, which we introduce in this work.

Strong privacy is achievable in the weighted setting by suppressing tally details and publishing only the winner. Such redaction, however, *would sacrifice transparency*, omitting critical statistics—such as the margin of victory and the rate of voter participation—that are non-negotiable for achieving trust in governance.

Two key questions thus arise:

Q1. **How can we effectively *measure* the privacy associated with published tallies in weighted voting?**

Q2. **How can we *enforce* privacy while preserving transparency for published tallies in weighted voting?**

Answering these questions requires a privacy framework tailored to weighted voting, since existing notions like ballot secrecy were designed for one-person-one-vote systems and fail to capture the risks posed by tallies themselves.

Key among these risks today is the rampant, growing problem of *bribery / vote-buying*. For example, vote-buying constitutes a $250+ million market in the Curve protocol [29], while the LobbyFi vote-buying protocol commands 8–14% of votes on major proposals in the popular Arbitrum L2 blockchain.

Tally privacy in the weighted-voting setting connects directly to bribery. For bribery strategy to be effective, an adversary must condition payouts on voter choices in a way that rewards compliance. To do so, the adversary must be able to deduce—or at least accurately estimate—how voters cast their votes. Adversarial exploitation of information in published tallies thus makes bribery strategies enforceable.

This observation motivates the new privacy notion we introduce in this work: **B-privacy** (short for "Bribery-privacy"). B-privacy measures the *economic cost* to an adversary of bribing voters given the information revealed in a published weighted-vote tally. B-privacy is complementary to standard coercion-resistance: while coercion-resistance prevents voters from cooperating with adversaries to prove how their ballot was cast, B-privacy addresses what the adversary can infer with tally access alone. Together, these properties provide comprehensive protection against bribery attacks. B-privacy thus addresses Q1 above with a concrete, economically grounded measure of privacy-related risk.

B-Privacy. Informally, the B-privacy of a system is the *minimum bribe* an adversary must pay to voters to achieve a desired outcome with a certain probability p (e.g., to ensure a yes outcome in a yes/no vote).

B-privacy is measured for a particular *tally algorithm*, which specifies how tallies are published. For example, a tally algorithm might publish individual cleartext ballots (resulting in minimal B-privacy, as an adversary can pay out perfectly targeted bribes). Or it might publish only the winner (maximizing B-privacy, but eroding transparency).

We define B-privacy in terms of a *bribery game*, a Bayesian game between the adversary and a group of rational voters. In this game, the adversary specifies bribe amounts and conditions of payment based on the published tally. (E.g., bribes might be paid if the yes / no winning margin exceeds 10%.) Voters then vote in a way that maximizes their expected utility, which combines their private utilities for vote outcomes and the potential bribe. B-privacy is defined as the minimum cost for the adversary to achieve its desired outcome in this game with a given probability p at *Bayesian Nash equilibrium*.

B-privacy is grounded in strategic behavior and cost, rather than idealized notions of secrecy, and so offers a practical lens on weighted-voting privacy. While no system can eliminate bribery, we introduce a tally algorithm that can boost B-privacy while retaining strong transparency.

Enforcing B-privacy via noising. We introduce a simple tally algorithm that *adds (Laplacian) noise* to a published tally. In answer to Q2 above, we show that this algorithm boosts B-privacy—i.e., yields a higher cost of bribery—while minimally perturbing tallies.

Our approach recalls techniques for enforcing differential privacy. A subtle but critical issue, however, is that adding noise can *flip a proposal's reported outcome*, implying an erroneous outcome. For example, in a 49.9% yes vs. 50.1% no vote, adding 0.2% noise in the yes direction would flip the reported outcome from no to yes.

Our tally algorithm thus also *corrects* the published outcome if necessary. This requirement, however, causes classical differential privacy bounds to break down. We therefore introduce *new proof techniques* to obtain bounds on the B-privacy of our tally algorithm—one of our key contributions.

Limitation: In this initial exploration of B-privacy, we restrict our focus to *binary* voting choices, notionally yes/no. We do not consider multi-choice ballots or the impact of abstention (which may be treated as a ballot option). Extension to multiple choices renders analysis much more complex. We conjecture that our results still hold directionally for the multi-choice case, a question that we leave for future work.

Contributions

Ours is the first framework that quantifies bribery risk economically and provides a tunable mechanism that balances privacy and transparency in practice for weighted voting. We review related work in Section 2 and give preliminary formalism in Section 3. Our contributions are then as follows:

- **Initiating study of weighted-voting privacy:** We introduce the problem of tally privacy for weighted-voting. We demonstrate the urgency of the problem with experimental, practical attacks on real-world DAO voting that reveal voter choices despite ballot secrecy (Section 4).
- **B-privacy:** We introduce and formally define B-privacy, an economic measure of bribery resistance that is the first privacy measure specific to weighted-voting systems (Section 5). We prove basic results on B-privacy and explore methods for computing it in practice (Section 6).
- **Noise-based privacy mechanism:** We present a simple, tunable noising mechanism that preserves winner correctness. We prove bounds on its balance between B-privacy and transparency (Section 7).
- **Empirical validation:** We analyze the effect of our noising mechanism on the B-privacy of 2,503 proposals across 27 DAOs. We show that with minimal tally perturbation—i.e., preserving strong transparency—our mechanism substantially improves B-Privacy (by a geometric mean factor of 3.5× over raw tallies) (Section 8).

We conclude in Section 9. We relegate to the paper appendices theorems and proofs omitted from the paper body for lack of space (Section A) and details on computational methods for B-privacy (Section B).

2 Related Work

Voting privacy fundamentals. Traditional privacy concepts for equal-weight voting include ballot secrecy, receipt-freeness, and coercion resistance, in order of increasing strength [3]. Ballot secrecy, a common protection in in-person voting, ensures that individual votes remain hidden from observers. Early cryptographic methods achieved this property [6,9]. Receipt-freeness prevents voters from proving their choices to others after the fact [5,21,39]. Coercion resistance achieves a similar property, but in a stronger model of pre-voting interaction with adversaries [1,11,20,24,30]. These notions and schemes do not immediately generalize to the weighted-voting setting, which thus requires new approaches.

Weighted voting privacy. Eliasson and Zúchete [14] and Dreier et al. [12] design secret-ballot schemes for weighted voting, while Cicada [19] does so specifically for blockchain governance, but all of these works disregard tally leakage of ballot contents. Kite [32] focuses on private delegation of voting weight. That work acknowledges that tallies may leak private information, briefly mentioning the idea of publishing limited-precision tallies.

Snapshot's "shielded voting" has broad real-world use in blockchain governance. It encrypts ballots, but only ephemerally, decrypting them once proposals conclude [43]. A proposed update offers persistent ballot secrecy [34] via cryptographic techniques and/or trusted execution environments.

Probabilistic tallying methods. Several approaches have explored probabilistic methods for vote tallying to address risks of tally leakage. DPWeVote applies differential privacy to weighted voting in semi-honest cloud environments using randomized response [46]. Random sample voting selects and tallies only subsets of ballots for efficiency [10], with risk-limiting tallies adapting this idea to counter coercion threats [23]. Such probabilistic approaches are unacceptable in many voting settings, as small electorates and tight margins may elevate the probability of an incorrect result being reported. Liu et al. analyze privacy under deterministic voting rules using distributional differential privacy, but focus primarily on the unweighted setting and without metrics for attack resistance [4].

Economic modeling of voting behavior. Game-theoretic models are an established approach for analyzing strategic voting behavior. Seidmann shows that when voters may receive external rewards, private voting leads to better organizational outcomes than public voting [40], while Bayesian models have been used to examine coordination effects [17], jury decision-making [13], shareholder voting [28], and weighted average voting [37]. Such games have also been used to examine the robustness of quadratic voting to strategic manipulation, including collusion and fraud [45]. Similarly, our B-privacy framework examines the robustness of different tally release mechanisms to bribery in weighted voting systems.

Decentralized autonomous organizations. DAOs represent an important setting for weighted voting and a rich source of real-world voting data. DAOs today provide either no or ephemeral ballot secrecy [43], but are starting to embrace ballot secrecy because of bribery concerns [8,33,34]. Empirical studies demonstrate that public visibility creates peer pressure and herding dynamics [16,41], theoretically reducing decentralization [15]. The research community has identified secure voting—specifically, privacy and coercion-resistance—as a critical open problem for DAOs [42], especially in the face of emerging threats like DAOs created for vote-buying, known as Dark DAOs [31,35]. While these challenges are well-documented, existing work lacks functional metrics to assess how they translate to bribery vulnerabilities. Our B-privacy framework addresses this gap with a quantifiable measure of bribery resistance, and our noise-based tally algorithm offers a practical approach to enhancing B-privacy while preserving tally transparency.

3 Voting Framework

To study privacy in weighted voting, we first present a general underlying framework used to represent elections and their outcomes. We model a standard voting data format that allows results to be reported in an arbitrary form, so that our definitions generalize to different tallying approaches.

Definition 1 (Voting transcript)**.** *A* voting transcript $t = (\mathbf{w}, \mathbf{c})$ *records the results of a proposal with n voters choosing from a set C of options, where:*

- $\mathbf{w} = (w_1, \ldots, w_n)$ *where* $w_i \in \mathbb{R}^+$ *is the voting weight of voter i and*
- $\mathbf{c} = (c_1, \ldots, c_n)$ *where* $c_i \in C$ *is the option chosen by voter i.*

We write $|\mathbf{s}|$ *for the* ℓ_1*-norm of sequence* $\mathbf{s}$ *and for convenience denote the total weight as* $W = |\mathbf{w}|$.

This definition captures scenarios where voters cast all their voting weight for a single option. While our model doesn't allow vote splitting, a voter who splits their votes can be treated as multiple logical voters with corresponding weights.

Definition 2 (Tally algorithm)**.** *A* tally algorithm *is a (possibly randomized) algorithm* tally *applied to a voting transcript t, where* $\mathsf{tally}\colon T \to O$ *maps transcripts from the space of possible transcripts T to outcomes in outcome space O.*

The outcome space O can take various forms depending on the tally algorithm—it might consist of aggregated voting weight for each choice, binary winners, full transcripts, or any other encoding of voting results. To illustrate this definition with a concrete example, consider the raw tally algorithm $\mathsf{tally}_{\mathsf{raw}}$, corresponding to the first case, in which transcripts are mapped to aggregated voting weight for each choice.

Algorithm Name	Specification
Winner-Only	$\mathsf{tally}_{\mathsf{winner}}(t) = \arg\max_{j \in C} \sum_{i:c_i=j} w_i$
Raw Tally	$\mathsf{tally}_{\mathsf{raw}}(t) = \left(\sum_{i:c_i=j} w_i\right)_{j \in C}$
Noised Tally	$\mathsf{tally}_{\mathsf{noised}(\nu)}(t) = \left(Y_j + \sum_{i:c_i=j} w_i\right)_{j \in C}$, $Y_j \xleftarrow{\$} \nu$
Full-Disclosure	$\mathsf{tally}_{\mathsf{public}}(t) = t$

Table 1: Tally algorithms. Each algorithm maps transcript t to an outcome: Winner returns only the winning choice; Raw tally reports the aggregate weight per choice; Noised tally adds noise $Y_j \sim \nu$ to each choice's tally; Full-Disclosure returns the complete voting transcript t.

Example 2 (Raw tally)**.** *Let* $\mathbf{w} = (3.5, 2, 1)$ *be voter weights and* $\mathbf{c} = (\mathit{yes}, \mathit{no}, \mathit{yes})$ *corresponding choices. Then the raw tally for* $t = (\mathbf{w}, \mathbf{c})$ *is:*

$$\mathsf{tally}_{\mathsf{raw}}(t) = \left(\sum_{i:c_i=\mathit{yes}} w_i, \sum_{i:c_i=\mathit{no}} w_i \right) = (4.5, 2).$$

That is, the yes *option receives total weight 4.5, and* no *receives total weight 2.*

Different tally algorithms offer different privacy-transparency tradeoffs. We summarize the key algorithms used throughout this paper in Table 1. Of particular interest is the noised tally algorithm, which we later show can significantly improve privacy while maintaining strong transparency guarantees.

The raw tally algorithm is the most natural and direct extension of one-person-one-vote privacy to weighted systems. Although Example 1 demonstrated that privacy can be broken in toy settings, one might hope that larger electorates or real-world proposals would obscure individual choices. We now show, however, that this is not the case.

4 Practical Attacks on Raw Tallies

We introduce two attack strategies for extracting individual votes from weighted-voting tallies. We call them a *whale attack* and a *subset sum attack*. We combine these two attacks into a *unified attack algorithm* that efficiently extracts individual ballots given only raw tallies and voter weights.

We explore the efficiency of our unified attack algorithm on data from DAOs. The fact that weighted voting is prevalent in DAOs and individual voting records are publicly available allows us to simulate hypothetical ballot-secrecy scenarios and validate our attacks against ground truth.

Key Finding

Even under hypothetical ballot secrecy, use of the raw tally algorithm results in significant loss of ballot privacy for DAO voters.

Thus weighted-voting systems cannot directly apply techniques from traditional voting and achieve equivalent privacy guarantees.

4.1 Attack setting and methodology

DAOs provide an ideal testbed for understanding weighted voting privacy. They use weighted voting and generally reveal individual ballots publicly on-chain, giving us ground-truth data to validate our attacks.[1]

We investigate a counterfactual scenario: *If a DAO enforced ballot secrecy but disclosed exact tallies (as is standard in traditional voting), what level of voter privacy would result?* We test our attack algorithm against this scenario by simulating attacks where only aggregate tallies are public, then verifying success against known individual voting records. To do so we use a dataset of votes from the Snapshot voting platform between September 2020 and February 2025 collected for a previous large-scale study of DAO voting [15], limiting our analysis to DAOs with more than 5 proposals, which yields 3,844 proposals across 31 DAOs.

In our attack model, we assume the set of participating voters and their weights are known (as is typical in token-based systems), but individual vote choices are hidden. Each proposal presents voters with multiple options—typically binary yes/no decisions, although some include additional alternatives. We neglect voters who implicitly abstain by not casting a vote, only targeting abstaining voters if they explicitly cast their ballot for an "abstain" choice. Our attacks work for proposals with any number of choices.

We now describe our two complementary attack strategies.

Whale attack. The term *whale* denotes large token holders in cryptocurrency systems—and high-weight voters in DAOs. Our *whale attack* exploits the simple but effective observation that if a whale's weight exceeds the total votes for some choice—the whale couldn't have backed that choice.

Let $\mathsf{tally}_{\mathsf{raw}} = (s_1, s_2, \ldots, s_{|C|})$ where s_j represents the total weight for choice $j \in C$. If $w_i > s_j$ for some voter i and choice j, then $c_i \neq j$.

Our whale attack is an *iterative* algorithm: after identifying and removing a whale's vote from one choice, we recompute the remaining tallies and apply the attack again. This process can cause tallies to flip—if enough weight is removed from the initially winning choice, a different choice may become

[1] While some systems, such as Snapshot, offer shielded voting as an option, these mechanisms conceal ballots only ephemerally and reveal (anonymous) address voting choices when a proposal vote has concluded.

the winner, enabling further whale identification. The algorithm continues until no more whales can be identified, often revealing a substantial fraction of the electorate by weight. It runs in $O(n \cdot |C|)$ time, making it efficient even for large electorates, and serves as an effective preprocessing step for our more computationally intensive subset sum attack.

Subset sum attack. Our *subset sum* attack exploits the precision of raw tallies by searching for vote assignments that produce observed outcome. In many cases, there exists a unique transcript $(\mathbf{w}, \mathbf{c})$ that yields the raw tally $\mathsf{tally}_{\mathsf{raw}}(\mathbf{w}, \mathbf{c})$. Reconstructing the vote assignment $\mathbf{c} = (c_1, \ldots, c_n)$ reduces to finding, for each choice $j \in C$, the subset of voters $\{i : c_i = j\}$ whose weights sum to the total s_j.

This generalizes the classic subset sum problem to multiple partitions: given voter weights $\mathbf{w}$ and target values $(s_1, s_2, \ldots, s_{|C|})$ from the tally, partition the voters such that each subset's weight sum matches its corresponding target. While multiple partitions could theoretically produce identical sums (and do when voters have identical weights), the high precision of token weights in DAOs (typically 18 decimal places) makes such collisions overwhelmingly unlikely, ensuring that any discovered solution is almost certainly the correct one. The subset sum variant underlying our attack algorithm is NP-hard in general, but two factors make it tractable using well-studied algorithms: (1) the precision constraint mentioned above and (2) small electorates are common in DAO governance.

Lagarias and Odlyzko [27] propose an expected polynomial-time algorithm, but it works only for low-density instances, a requirement not satisfied by most DAO voting weight distributions. Instead, we employ the meet-in-the-middle approach of Horowitz and Sahni [22], which splits voter weights into two halves, enumerates all partial sums in each half, and searches for combinations that satisfy the target constraints. This algorithm has $O^*(2^{n/2})$ time and space complexity, making it practical for $n \lesssim 45$ voters on commodity hardware—a significant improvement over the naive $O^*(2^n)$ brute-force approach.

To recover individual votes, we solve a modified problem for each voter i: we attempt to find a valid partitioning of $\{w_j : j \neq i\}$ such that adding w_i to the subset for choice c yields the target sum s_c. If exactly one choice c allows a valid solution, then voter i must have chosen c.

Preprocessing using the whale attack reduces the effective problem size from n to some $n' \leq n$ voters and the complexity of the subset sum instance from $O^*(2^{n/2})$ to $O^*(2^{n'/2})$. This often shrinks problem instances to tractable size.

Our unified attack algorithm is specified as Algorithm 1.

Algorithm 1 Unified Attack Algorithm for Extracting Ballots from a raw tally $\mathsf{tally}_{\mathsf{raw}}(t)$

Input: Voters $\{1, \ldots, n\}$, weights $\mathbf{w} = (w_1, \ldots, w_n)$, raw tally $\mathsf{tally}_{\mathsf{raw}}(t) = (s_1, \ldots, s_{|C|})$
$\mathbf{d} \leftarrow \mathbf{0}$ ▷ Determined votes vector
$U \leftarrow \{1, \ldots, n\}$ ▷ Undetermined voters
while true **do** ▷ Whale Attack
  $j^* \leftarrow$ any element of $\arg\max_{j \in C} s_j$ ▷ Winner
  $s_2 \leftarrow$ 2nd-largest value in $(s_1, \ldots, s_{|C|})$
  if $s_{j^*} = s_2$ **then break** ▷ No unique winner
  $W \leftarrow \{i \in U \mid w_i > s_2\}$ ▷ Whales
  if $W = \varnothing$ **then break**
  for all $i \in W$ **do**
    $d_i \leftarrow j^*$ ▷ Voter i must have voted for j^*
    $U \leftarrow U \setminus \{i\}$
    $s_{j^*} \leftarrow s_{j^*} - w_i$ ▷ Update remaining tally
end while
if $|U| \leq 45$ **then** ▷ Subset Sum Attack
  for all $i \in U$ **do**
    Split $U \setminus \{i\}$ into two halves L, R
    $L_s \leftarrow \{\sum_{k \in S} w_k \mid S \subseteq L\}$
    $R_s \leftarrow \{\sum_{k \in S} w_k \mid S \subseteq R\}$
    $Q \leftarrow \{j \in C \mid \exists \ell \in L_s, r \in R_s : \ell + r + w_i = s_j\}$
    if $|Q| = 1$ **then**
      let j^* be the sole element of Q
      $d_i \leftarrow j^*$ ▷ Voter i must have voted for j^*
      $U \leftarrow U \setminus \{i\}$
      $s_{j^*} \leftarrow s_{j^*} - w_i$ ▷ Update remaining tally
return d

4.2 Results

We applied our unified attack algorithm (Algorithm 1) to 3,844 proposals across 31 DAOs. The attack breaks plausible deniability—meaning that it definitively identifies at least one voter's choice—on 3,118 proposals (81.0%) and recovers *all* voter choices on 1,122 proposals (29.2%). Among these 3,118 vulnerable proposals, the attack leaks on average 41.6% of ballots and 85.1% of voting weight per proposal.

Figure 1 shows the mean effectiveness across DAOs, revealing two key patterns:

Small DAOs experience near-complete privacy failures. Among proposals with 45 or fewer voters, we achieve complete vote recovery in 791 out of 878 cases (90.1%). The remaining 87 proposals resist attack only due to voters with identical weights casting different ballots—a scenario that creates fundamental ambiguity. The cluster of smallest DAOs in the top-right of the plot demonstrates that below a certain size threshold, privacy protections collapse entirely. This reflects the subset sum component succeeding on virtually every small proposal, often aided by whale-attack preprocessing that reduces larger proposals to manageable sizes. In 346 proposals that initially exceeded the 45-voter threshold, whale-attack preprocessing successfully reduced the residual voter count to $\leq$45, enabling the subset sum attack to succeed.

Larger DAOs' vulnerability depends on whale concentration. While DAOs with more voters generally appear more resistant (clustering near the y-axis), significant outliers exist. Large DAOs where the subset sum attack is inapplicable may still be highly vulnerable to whale attacks depending on their voting weight distribution. This whale impact is visible in the region near 0% ballots leaked —few individual votes are revealed, but the most influential voters are completely exposed, with up to 80% total voting weight leaked.

Figure 2 illustrates variations in attack success across proposals within individual DAOs. The effectiveness of both the whale and subset-sum attacks is clearly visible in the Balancer results (far right panel). For all proposals with fewer than 45 voters, the attack achieves complete success, leaking 100% of voting weight. For proposals exceeding 45 voters, whale-attack preprocessing successfully reduces the problem size below the 45-voter threshold in all but two cases, enabling the subset-sum attack to proceed.

Attack success varies considerably across the other three DAOs, revealing several key patterns. Larger winning margins (shown in yellow/green) consistently lead to higher privacy compromise rates, as whales become easier to identify when one choice receives disproportionately low support. This demonstrates that privacy under raw tallies depends on both electorate size and voting patterns.

Notably, DAO size alone does not determine vulnerability. Despite having roughly 10× more voters than Aavegotchi on average, Arbitrum shows much higher attack success rates, with a greater density of proposals where >60% of voting weight is leaked. The key difference lies in voting weight distribution: while both DAOs have similar average winning margins, Arbitrum exhibits much more concentrated voting weight. In high-margin proposals (>90%), whale attacks leak 88.0% of voting weight in Arbitrum versus only 38.1% in Aavegotchi, highlighting how voting weight concentration can outweigh electorate size in determining vulnerability. These results demonstrate that raw tally vulnerability depends on a combination of electorate size, winning margins, and voting weight concentration.

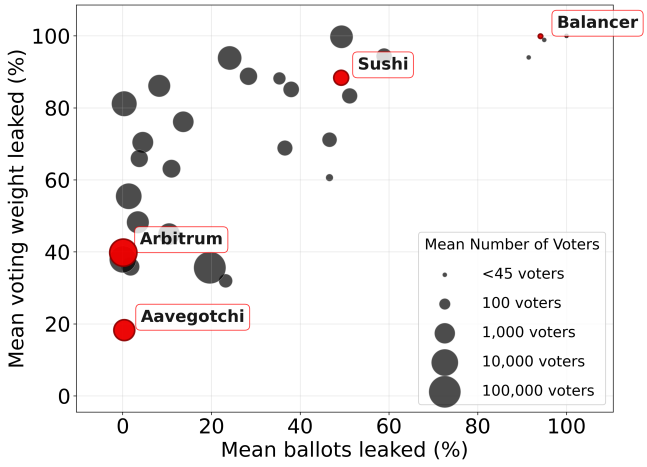

Figure 1: Unified attack algorithm results across all DAOs. Each point represents one DAO and shows mean percentage of ballots leaked (x-axis) vs. mean percentage of voting weight leaked (y-axis) across that DAO's proposals. Point size indicates average voters per proposal. The attacks succeed across diverse DAOs, demonstrating that the raw tally algorithm fails to provide adequate ballot secrecy in weighted voting systems. The attack success for highlighted DAOs is presented in more detail in Figure 2.

5 B-Privacy: Definition

In this section, we introduce *B-Privacy*, our new privacy metric for weighted voting. We first present a game-theoretic model of adversarial vote-buying that we call a *bribery game* (Section 5.1). It underpins our subsequent formal definition of B-privacy (Section 5.2).

5.1 Bribery game

Our bribery game is a Bayesian game in which voters are rational agents who maximize their expected payoff under uncertainty about other voters' preferences. Each voter must make decisions based on incomplete information about how others will vote, while an adversary strategically offers bribes to influence the outcome.

Voter utilities. Our bribery game requires an extension of our voting framework to include voter utilities. For each option $c \in C$, let $U^c = (U_1^c, \ldots, U_n^c)$ where $U_i^c \in \mathbb{R}$ represents voter i's utility from outcome c. We focus on binary proposals ($C = \{\text{yes}, \text{no}\}$) and normalize by setting $U_i^{\text{yes}} = 0$, so U_i^{no} represents voter i's relative preference for the "no" option. We assume all voters participate—abstention could be modeled as an explicit third choice, although we restrict to the binary case for simplicity as it captures the essential tension between tally transparency and B-Privacy while avoiding the additional complexity of preference aggregation across multiple alternatives. We leave extension to multi-choice settings for future work.

Our model for this game must specify what information voters have about other voters' utilities and likely choices. We follow the standard approach in Bayesian voting games (e.g. [37]) and model utilities as private types drawn from commonly known distributions. Each voter i's utility U_i^{no} is drawn independently from distribution $\mathbf{U}_i^{\text{no}}$ with infinite support, where these distributions are public but the realized values are private information. Although assuming common knowledge of utility distributions may be unrealistic in some cases, in our setting it models a powerful observer leveraging public data (on-chain voting history, forum discussions etc.) to form accurate priors on voter utilities.

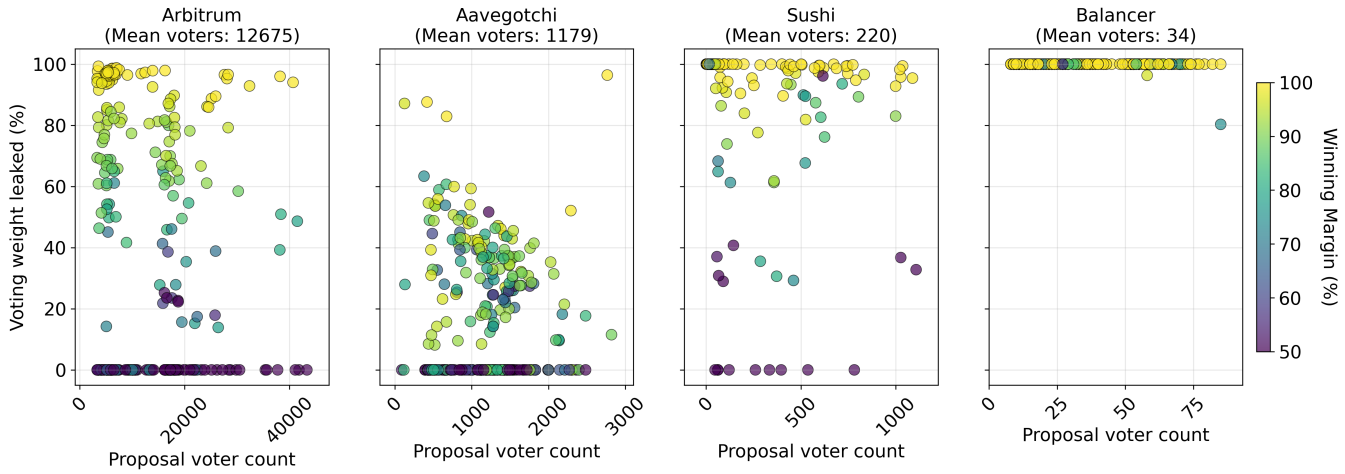

Figure 2: Variations in attack effectiveness across four DAOs with different scales. Each point represents one proposal, colored by winning margin. The view across proposals demonstrates that Attack success is driven by the interaction of electorate size, whale concentration and winning margins—large whales become easier to identify when one choice receives disproportionately low support. Balancer shows near-complete compromise due to small size, while Arbitrum's higher vulnerability compared to Aavegotchi (despite 10× more voters) demonstrates that voting weight concentration can be more important than electorate size for determining privacy risk.

Coercion-resistance. Our bribery game assumes the underlying voting system is coercion-resistant: voters cannot cooperate with an adversary to prove how a ballot was cast. Under this assumption, the published tally is the sole information available for designing bribery strategies. B-privacy therefore complements coercion-resistance: the latter prevents verification through direct interaction, while the former limits what can be inferred from the tally alone.

We present our full bribery game in Figure 3.

Adversarial strategy. In our bribery game, the adversary commits to bribe amounts $\mathbf{b}$ prior to the proposal. They do not depend on the outcome. The payment conditions $\mathbf{f}$, in contrast, can depend on the outcome. This separation cleanly distinguishes the information-theoretic aspects of tally algorithm (captured by optimal $\mathbf{f}$) from the economic optimization of bribe amounts ($\mathbf{b}$), enabling the comparison among tally algorithms that is our main focus.

5.2 B-privacy definition

Using the bribery game we can now define B-Privacy.

Definition 3 (B-Privacy)**.** *For given inputs to the bribery game* $(\mathbf{w}, \mathbf{U}^{\text{no}})$*, the* B-privacy $B_{\mathsf{tally}}(p)$ *is the minimum total bribe budget required for an adversary to achieve success probability* p *when tally algorithm* tally *is used:*

$$B_{\mathsf{tally}}(p) = \min_{\mathbf{b},\mathbf{f}} |\mathbf{b}| \; s.t. \quad p_{\mathsf{succ}} \geq p.$$

The relative *B-privacy of tally algorithm* tally *is the ratio* $B_{\mathsf{tally}}(p)/B_{\mathsf{tally}_{\mathsf{public}}}(p)$.

B-Privacy is measurable for any tally algorithm. Even $\mathsf{tally}_{\mathsf{winner}}$, which only discloses the winner, enables strategic bribery through outcome-conditional payments (e.g., "payment if yes wins"). Looking forward, one of our contributions is to identify tally algorithms that maximize transparency while maintaining high B-Privacy.

6 B-Privacy: Computation

Computing B-privacy, as specified in Definition 3, requires computation of optimal adversarial bribery strategies. In this section, we introduce foundational analytic concepts for this purpose (Section 6.1) and then present results and methods for computing B-privacy (Section 6.2).

6.1 Pivotality and bribe margin

In analyzing and computing B-privacy, we make use of two foundational concepts: *pivotality* and *bribe margins*. For a given voter i, tally algorithm, and adversarial strategy, these concepts respectively reflect the voter's likelihood of receiving a bribe and of affecting the vote outcome given the voter's individual voting choice.

Both quantities are computed from voter i's perspective and assume a Bayesian Nash equilibrium has been reached. Unless otherwise noted, probabilities are taken over the randomness in other voters' utilities $U_j^{\text{no}} \xleftarrow{\$} \mathbf{U}_j^{\text{no}}$ for $j \neq i$, which determine the equilibrium choices c_j of other voters and the resulting voting transcript $t = (\mathbf{w}, \mathbf{c})$.

Bribery Game

Game Inputs:

- Voter weights $\mathbf{w} = (w_1, \ldots, w_n)$
- Utility distributions $\mathbf{U}^{\mathsf{no}} = (\mathbf{U}_1^{\mathsf{no}}, \ldots, \mathbf{U}_n^{\mathsf{no}})$
- Tally algorithm tally

Players: n voters (indexed by $i \in \{1, \ldots, n\}$) and one adversary

Adversary's Objective: The adversary seeks to maximize the probability of a yes outcome.

Information Structure:

- Private utilities $U_i^{\mathsf{no}} \xleftarrow{\$} \mathbf{U}_i^{\mathsf{no}}$ are drawn for each voter i
- Each voter i observes only their own private utility U_i^{no}
- The distributions $\mathbf{U}_1^{\mathsf{no}}, \ldots, \mathbf{U}_n^{\mathsf{no}}$ are common knowledge among all players

Game Sequence:

1. *Adversary's strategy*: Adversary commits to bribe amounts $\mathbf{b} = (b_1, \ldots, b_n)$ and bribery condition functions $\mathbf{f} = (f_1, \ldots, f_n)$ where $f_i : O \to \{0,1\}$ with O being the set of possible outcomes, and $f_i(\mathsf{tally}(t)) = 1$ indicating voter i receives bribe b_i given voting transcript $t = (\mathbf{w}, \mathbf{c})$
2. *Voting stage*: Voters simultaneously choose votes $\mathbf{c}$ to maximize expected utility
3. *Outcome*: Tally algorithm tally reveals outcome information; bribes paid per condition functions

Equilibrium: Bayesian Nash equilibrium where each voter's strategy maximizes their expected utility given their beliefs about others' behavior, and these beliefs are consistent with equilibrium play.

Figure 3: Bribery game underpinning B-privacy definition.

Definition 4 (Pivotality)**.** *For voter i, the pivotality Δ_i measures how much their vote affects the probability of a* no *outcome:*

$$\Delta_i = \Pr[\mathsf{no}\ \textit{wins} \mid c_i = \mathsf{no}] - \Pr[\mathsf{no}\ \textit{wins} \mid c_i = \mathsf{yes}],$$

Pivotality is a standard concept in voting games [13, 17, 45]. In our framework, it captures the intuition that voters with lower pivotality are more susceptible to bribes—since their vote is less likely to affect the outcome, the potential bribe becomes relatively more attractive. The bribe margin, by contrast, is specific to our bribery framework and captures how a voter's choice affects their expected bribe payment under the adversarial strategy.

Definition 5 (Bribe margin)**.** *For voter i with bribery condition function f_i, the bribe margin α_i is the additional probability of receiving a bribe when voting* yes *versus* no*:*

$$\alpha_i = \Pr[f_i(\mathsf{tally}(t)) = 1 \mid c_i = \mathsf{yes}] - \Pr[f_i(\mathsf{tally}(t)) = 1 \mid c_i = \mathsf{no}].$$

These quantities determine voter i's decision as shown in the following theorem:

Theorem 6 (Adversary's success probability)**.** *Given bribe vector* $\mathbf{b}$ *and bribery condition functions* $\mathbf{f}$*, the adversary's probability of achieving a* yes *outcome is*

$$p_{\mathsf{succ}} = \Pr\left[\sum_{i=1}^{n} w_i X_i > W/2\right],$$

where $X_i = \mathbb{I}[U_i^{\mathsf{no}} \le \frac{\alpha_i b_i}{\Delta_i}]$ for $U_i^{\mathsf{no}} \xleftarrow{\$} \mathbf{U}_i^{\mathsf{no}}$ is an indicator random variable and voting behavior follows the Bayesian Nash equilibrium induced by $(\mathbf{b}, \mathbf{f})$*.*

Proof. Under bribe vector $\mathbf{b}$ and bribery condition functions $\mathbf{f}$, voter i's expected utility $\mathbb{E}[U_i]$ given they vote yes is:

$$\Pr[\mathsf{no\ wins} | c_i = \mathsf{yes}] \cdot U_i^{\mathsf{no}} + b_i \cdot \Pr[f_i(\mathsf{tally}(t)) = 1 | c_i = \mathsf{yes}].$$

And given they vote no is:

$$\Pr[\mathsf{no\ wins} | c_i = \mathsf{no}] \cdot U_i^{\mathsf{no}} + b_i \cdot \Pr[f_i(\mathsf{tally}(t)) = 1 | c_i = \mathsf{no}].$$

Accordingly, in equilibrium, voter i votes yes if and only if

$$\begin{aligned} \mathbb{E}[U_i | c_i = \mathsf{no}] - \mathbb{E}[U_i | c_i = \mathsf{yes}] &\le 0 \\ \Delta_i U_i^{\mathsf{no}} - \alpha_i b_i &\le 0 \\ U_i^{\mathsf{no}} &\le \frac{\alpha_i b_i}{\Delta_i}. \end{aligned}$$

The second line follows by substituting the definitions of pivotality Δ_i and bribe margin α_i.

Every voter's private utility is drawn as $U_i^{\mathsf{no}} \xleftarrow{\$} \mathbf{U}_i^{\mathsf{no}}$, so a voter's contribution to the yes total is the random variable $w_i X_i$ where $X_i = \mathbb{I}[U_i^{\mathsf{no}} \le \frac{\alpha_i b_i}{\Delta_i}]$. The adversary succeeds when the yes total exceeds $W/2$, which occurs with probability

$$p_{\mathsf{succ}} = \Pr\left[\sum_{i=1}^{n} w_i X_i > W/2\right].$$

□

The key to analyzing B-Privacy for a given bribery strategy $(\mathbf{b}, \mathbf{f})$ lies in computing the equilibrium values of bribe margin α_i and pivotality Δ_i. We show later for the tally algorithms we consider, bribe margins (or bounds on them) can be expressed in terms of pivotality, which simplifies analysis. A Bayesian Nash equilibrium occurs where each voter's belief about their pivotality is consistent with the strategy this belief induces.

6.2 Optimal adversarial strategy

To compute B-Privacy, we must solve the optimization / minimization problem in Definition 3.

A key insight is that we can restrict attention to strategies where condition functions operate independently across voters—that is, voter i's payment depends only on the outcome, not on correlations with other voters' payments. While correlated payment schemes might seem more powerful, we show in Appendix A.1 that any correlated strategy can be replaced by an equivalent independent strategy with the same B-privacy. The intuition is that each voter cares only about

their own probability of receiving a bribe under different outcomes. Correlating payments across voters doesn't change these individual probabilities, so it provides no advantage to the adversary. This allows us to focus on the simpler case where $\mathbf{f} = (f_1, \ldots, f_n)$ for independent condition functions.

Given this simplification, our goal becomes characterizing the adversary's optimal choices of bribe amounts $\mathbf{b}$ and condition functions $\mathbf{f}$ that, per the definition of B-privacy, minimize total cost while achieving success probability p. We refer to the values of $\mathbf{b}$ and $\mathbf{f}$ that solve this B-privacy optimization problem $\min_{\mathbf{b},\mathbf{f}} |\mathbf{b}|$ as "optimal."

We show in Theorem 7 that the optimal strategy has a simple structure: the adversary should choose condition functions that maximally exploit the information revealed by the tally algorithm, then set bribe amounts to achieve the target success probability at minimum cost.

As with Definitions 4 and 5, probabilities are taken over other voters' utilities $U_j^{\mathsf{no}} \xleftarrow{\$} \mathbf{U}_j^{\mathsf{no}}$ for $j \neq i$, which determine their equilibrium choices.

Theorem 7 (Optimal bribery condition functions)**.** *The optimal condition function for voter i is the one that maximizes bribe margin α_i. Define $p^o_{i,c} = \Pr[\mathsf{tally}(t) = o \mid c_i = c]$. Then the optimal condition function takes the form:*

$$f_i^*(o) = \mathbb{I}\{p^o_{i,\mathsf{yes}} \geq p^o_{i,\mathsf{no}}\}$$

and corresponds to optimal bribe margin:

$$\alpha_i^* = \sum_{o \in O} \max(p^o_{i,\mathsf{yes}} - p^o_{i,\mathsf{no}}, 0).$$

The proof is deferred to Appendix A.2. This theorem tells us that adversaries should condition bribes on outcomes that serve as strong evidence of voter compliance. Specifically, the adversary pays voter i when the observed outcome o is more likely under the scenario where voter i voted yes than under the scenario where they voted no. The resulting bribe margin α_i^* quantifies how much this optimal conditioning improves the adversary's ability to target bribes. Theorem 6 shows that higher bribe margins allow the adversary to achieve the same success probability with lower total bribe costs.

Computing $B_{\mathsf{tally}}(p)$. Given optimal condition functions $\mathbf{f}^*$, computing $B_{\mathsf{tally}}(p)$ requires solving a coupled optimization problem involving both bribe allocation and equilibrium computation. The challenge is that voter equilibrium behavior (captured by pivotalities Δ) determines optimal bribe amounts $\mathbf{b}$, while the bribes themselves affect voter probabilities, which in turn determine the pivotalities. This creates a circular dependency that complicates analytical treatment. Additionally, bribe margins α for some tally algorithms depend on the pivotalities, adding another layer of interdependency.

Rather than pursue analytical solutions to the resulting Bayesian Nash equilibria, we use a computational approach to search heuristically for and thus approximate, the bribe amounts $\mathbf{b}^*$ for a given budget B—and their associated adversarial success probability. We adopt an iterative approach detailed in Appendix B. Briefly, for a given budget B, we initialize pivotalities and iterate until convergence:

1. Compute bribe margins using current pivotalities.
2. Allocate bribes by solving the constrained optimization problem: maximize the adversary's success probability subject to the budget constraint $\sum_i b_i = B$ using calculus-based optimization techniques.
3. Update pivotalities based on the resulting voter choice probabilities.

We then use binary search over budgets to find the smallest budget B such that the adversary achieves success probability p. This is an approximation of (upper bound on) the true minimum budget B^*, which corresponds to the B-privacy value $B_{\mathsf{tally}}(p)$. The optimization landscape may contain local minima, so our approach finds locally optimal solutions that approximate the global optimum. While our results are approximations, we use the techniques we would expect an adversary to adopt and thus capture achievable bribery costs in practice.

Summary: computing B-privacy. To compute $B_{\mathsf{tally}}(p)$ for a given tally algorithm tally and probability p, we must solve the optimization problem specified in Definition 3. While optimal condition functions $\mathbf{f}^*$ have the analytical characterization of Theorem 7, we use numerical methods for the coupled equilibrium and bribe allocation problem. The optimization may converge to local minima, so analytical solutions or systematic exploration of allocation strategies would strengthen these approximations. However our computational approach, applied to real DAO data in Section 8, models how a strategic adversary would approximate the B-privacy minimization in practice. We find empirically that this approach converges reliably across all tally algorithms and proposals in our analysis.

6.3 B-Privacy and Plausible Deniability

To connect B-Privacy with classical privacy notions, we examine its relationship to *plausible deniability*, a privacy concept that captures whether individual choices can be inferred from disclosed information.

Within our bribery game framework, plausible deniability measures the adversary's uncertainty about a voter's choice given the tally outcome.

Definition 8 (Plausible Deniability)**.** *For voter i and tally algorithm* tally*, let $p_o^{i,c} = \Pr[c_i = c | \mathsf{tally}(t) = o]$. We define the plausible deniability as:*

$$PD_i^{\mathsf{tally}}(o) = \min\left(\frac{p_o^{i,yes}}{\Pr[c_i = yes]}, \frac{p_o^{i,no}}{\Pr[c_i = no]}\right).$$

The expected plausible deniability of voter i under tally algorithm tally *is:*

$$EPD_i^{\mathsf{tally}} = \mathbb{E}_{o\sim\mathsf{tally}(t)}[PD_i^{\mathsf{tally}}(o)].$$

where the expectation is over the distribution of possible outcomes induced by the randomness in voter utilities in our bribery game model.

In our definition the normalization by prior probabilities distinguishes information revealed by the tally from from prior knowledge, and we take the minimum across both choices to capture worst-case privacy loss: plausible deniability is compromised if the adversary can confidently rule out either choice. This definition captures the privacy failures from Section 4: when our attacks definitively identified a voter's choice (e.g., a whale whose weight exceeds total no votes must have voted yes), that voter has $PD_i^{\mathsf{tally}}(o) = 0$.

Plausible deniability directly relates to the bribe margin α_i, which in turn determines B-Privacy:

Theorem 9. *For voter i in the Bribery Game with tally algorithm* tally*,*

$$EPD_i^{\mathsf{tally}} = 1 - \alpha_i^*,$$

where α_i^* is the optimal bribe margin for voter i.

The proof can be found in Appendix A.3. This equation demonstrates the inverse relationship between bribe margin and plausible deniability: as the adversary's ability to condition bribes on outcomes increases (higher α_i^*), the voter's privacy decreases (lower EPD_i).

7 Noised Tally Algorithms

The attacks presented in Section 4 show that releasing raw tallies undermines user privacy—in many cases stripping users even of plausible deniability, i.e., leaking their exact voting choices. This ability by adversaries drives down B-privacy.

In this section we explore the mechanism mentioned in Section 3: *noised* tally algorithms. We analyze how their properties can improve B-privacy while maintaining a high level of transparency in resulting tallies.

As with our previous analyses, for simplicity we assume binary choice proposals. The goal then is to tally the weight that voted yes (the remaining weight must have voted no) with added noise from some distribution ν:

$$\mathsf{tally}_{\mathsf{noised}(\nu)}(t) = Y + \sum_{i:c_i=\mathsf{yes}} w_i, \quad Y \xleftarrow{\$} \nu.$$

A subtle but key technical issue here is that noising may yield a tally that *indicates the incorrect winner*. As long as the support of the noise distribution includes values greater than the margin, there is a risk of incorrectly flipping the result. Tightly bounding ν might seem to address this problem, but may unduly erode the benefit of noising and will still induce an incorrect result with non-zero probability. We instead consider a simple solution: Specify the winner in the tally in addition to the noised result. We call this a *corrected* noised tally algorithm. Denoted by $\mathsf{tally}_{\mathsf{noised}(\nu)+}$, it is as follows:

$$\mathsf{tally}_{\mathsf{noised}(\nu)+}(t) = \left(\mathsf{tally}_{\mathsf{noised}(\nu)}(t), \mathsf{tally}_{\mathsf{winner}}(t)\right).$$

Our noising approach is closely related to differential privacy. The basic noised tally algorithm instantiated with Laplacian noise is identical to the Laplacian mechanism and satisfies differential privacy. However, preserving election integrity (always reporting the correct winner) is *fundamentally incompatible* with differential privacy—any mechanism that deterministically reveals the winner cannot be differentially private. This is clearest when one voter controls majority weight: revealing the winner reveals that voter's vote regardless of noise. Our corrected noised tally therefore requires custom privacy analysis. The following theorem bounds bribe margins for corrected noised tallies.

Theorem 10 (Bound on corrected noised tally bribe margin)**.** *When the corrected noised tally* $\mathsf{tally}_{\mathsf{noised}(\nu)+}$ *is used, the optimal bribery condition function for any voter i has bribe margin at most*

$$\Delta_i + (1-\Delta_i)\int_{-\infty}^{\infty} \max\left(\Pr[Z = u - w_i] - \Pr[Z = u], 0\right) du,$$

for random variable $Z \sim \nu$.

The proof is deferred to Appendix A.4.

Intuition. The bribe margin decomposes based on whether voter i is pivotal. When pivotal (with probability Δ_i), voter i's vote determines the winner, giving the adversary perfect conditioning ability. When not pivotal (with probability $1-\Delta_i$), both choices yield the same winner, so the adversary must use the noised tally to distinguish between them. The integral term measures how much the noise distributions overlap when voter i's weight shifts the tally by w_i. Greater overlap means less adversarial advantage from the noised information. We empirically explore this intuition in Section 7.1 by adapting our raw tally attacks to the noised setting.

Theorem 10 result reveals a fundamental tradeoff in noised tally algorithms. While outputting the winner guarantees correctness of the final decision, transparency depends critically on the noise distribution chosen. Adding more noise improves B-Privacy (by reducing the integral term) but degrades the fidelity of the raw tally, limiting its usefulness for understanding margins and community consensus.

In Section 8, we explore navigation of this tradeoff in practice—that is, whether we can achieve strong relative B-Privacy with sufficiently low error in the raw tally to preserve meaningful transparency.

Practical Implementation. Verifiable private voting systems rely on Zero-Knowledge Proofs (ZKPs) to prove result

correctness without revealing individual ballots. While standard ZKPs are deterministic, our mechanism requires stochastic noise. This can be implemented verifiably by deriving randomness from hashed witness commitments [26], acting as a Fiat-Shamir challenge that prevents tallier manipulation.

7.1 Attacks on noised tallies

To provide practical intuition for how noise limits adversarial capabilities, we adapt our raw tally attacks to the noised setting. Consider an adversary who receives a noised tally and knows the noise magnitude is bounded by 10% of total voting weight ($0.1W$).

Noise fundamentally breaks the subset sum attack, since it's extremely unlikely that any voter partitioning produces the exact noised tally values. The whale attack, however, remains partially viable with modifications.

Under raw tallies, if a whale's weight w_i exceeds the total votes s_j for some choice j, then certainly $c_i \neq j$. With noise, this logic requires adjustment: the adversary can only conclude $c_i \neq j$ if $w_i > s_j + 0.1W$, accounting for the possibility that noise reduced the true tally by up to $0.1W$.

This adapted whale attack—which we implement by modifying the threshold condition in our algorithm—represents a conservative adversarial strategy that maintains high confidence in leaked votes. Figure 4 shows the dramatic reduction in attack effectiveness: the same DAOs that suffered near-complete privacy compromise under raw tallies achieve substantially better privacy protection with noised tallies.

The results illustrate why adding noise creates practical challenges for adversaries. While a more aggressive adversary might accept greater uncertainty by using smaller thresholds (or none at all), such strategies sacrifice certainty and so are not directly comparable to the raw tally attack. The practical limitations on the attack considered here aligns with our theoretical bound from Theorem 10, demonstrating how noise translates into concrete privacy benefits.

8 Empirical Analysis of B-Privacy

Our model characterizes how adversarial knowledge, weight distributions, and tally algorithms influence B-Privacy. We now seek to understand *in practice* the extent to which noised tallying can raise the cost of bribery without undermining transparency and how intrinsic election properties such as decentralization mediate this trade-off. We perform an empirical analysis on the same DAO dataset used in Section 4, excluding degenerate proposals where a single whale controls more than 50% of the total voting weight. Such proposals are omitted because they trivially yield relative B-Privacy 1: the whale's vote alone effectively determines the outcome under any tally algorithm. We also exclude proposals from Stargate DAO, as the large number of voters in these proposals made determining the equilibrium in the bribery game computationally infeasible.

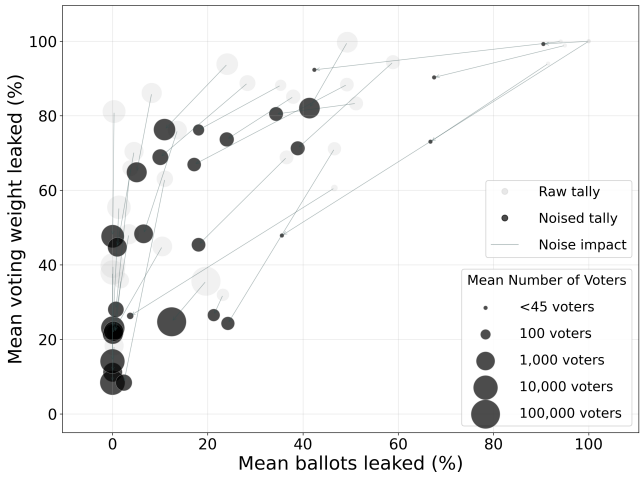

Figure 4: Comparison of raw tally versus adapted noised tally attack effectiveness across all DAOs when the noise magnitude is bounded by $0.1W$. Setup mirrors Figure 1, but shows how each DAO's position shifts when attacks are adapted for noised tallies. The leftward and downward shift demonstrates greatly reduced attack effectiveness under noise.

We compute B-Privacy over the resulting dataset of 2,503 governance proposals drawn from 27 DAOs, restricting to binary choices as before. We compare three tally algorithms: full-disclosure $\mathsf{tally}_{\mathsf{public}}$ (revealing individual votes), corrected noised tally $\mathsf{tally}_{\mathsf{noised(v)+}}$ of Theorem 10, and winner $\mathsf{tally}_{\mathsf{winner}}$ (revealing only the winning choice). Since we use an upper bound on bribe margins for the corrected noised tally (Theorem 10), our computed B-privacy values for this algorithm represent lower bounds on the true B-privacy, making our privacy improvements conservative estimates.

8.1 Experimental setup

Voter utilities. We simulate adversarial beliefs about voter utilities using normal distributions centered at the observed vote choice: $\mathbf{U}_i^{\mathsf{no}} \sim \mathcal{N}(1,1)$ if voter i voted no and $\mathbf{U}_i^{\mathsf{no}} \sim \mathcal{N}(-1,1)$ if they voted yes. This captures realistic adversarial uncertainty: a voter who voted no would vote yes in our model only if their sampled utility is negative, which occurs with probability $P(\mathcal{N}(1,1) < 0) \approx 16\%$. Thus roughly 84% of voters stick with their observed choice, representing reasonable but imperfect adversarial knowledge that could be inferred from public information such as past voting patterns or public statements. Crucially, while varying σ affects absolute B-privacy levels (higher uncertainty increases B-privacy), we find that relative B-privacy—the ratio between different tally algorithms—remains stable across distribution choices. This makes our comparative analysis robust to the specific

modeling assumptions about adversarial knowledge.

Adversarial success probability. We represent a highly motivated adversary seeking a high probability of success by setting the adversarial target success probability to $p = 0.9$. Results for other high values of p are qualitatively similar.

Noise distribution and tally perturbation. We use a Laplacian noise distribution $\nu = \text{Lap}(0, b)$ for $\text{tally}_{\text{noised}(\nu)+}$. For a given proposal, we calibrate the parameter b in a way that helps with understanding how tallies are perturbed and thus the impact on transparency, as follows.

Let the total voting weight for the proposal be W. For a noise distribution ν, we define the *tally perturbation* (TP) with frequency q as a percentage $d\%$ such that:

$$\Pr(|Z| \leq dW) = q, \text{ for } Z \xleftarrow{\$} \nu. \tag{1}$$

In other words, a q-fraction of the time, the noise Z drawn from ν has magnitude $\leq dW$. In our experiments, we set the TP frequency $q = 0.95$. We explore $d \in [0\%, 100\%]$, but in practical applications would typically choose small TP values, e.g., $d = 10\%$. For such parameters, 95% of the time, $|Z| \leq 0.1W$, so that the published tally differs from the true, raw tally, by at most 10%.

For a chosen frequency q and value d of TP, we set b for noise $\nu = \text{Lap}(0, b)$ to achieve it. For Laplace noise $Z \sim \text{Lap}(0, b)$, this results in the choice $b = \frac{dW}{\ln 20}$ since $\Pr(|Z| \leq x) = 1 - e^{-x/b}$.

An example helps illustrate.

Example 3 (Tally perturbation)**.** *Consider a proposal with:*

- *Weights:* $\mathbf{w} = (1, 1.1, 1.3)$ *(thus* $W = 3.4$*)*
- *Voter choices:* $\mathbf{c} = (\textit{yes}, \textit{no}, \textit{yes})$
- *Raw tally:* $T = (\textit{no}: 1.1, \textit{yes}: 2.3)$

Suppose $q = 0.95$ *and* d = *10%. Then computation of the corrected noised tally might look like this:*

- *Noise distribution:* $\nu = \text{Lap}(0, b)$ *for* $b = 0.1W / \ln 20$
- *Noise draw:* $Z = 0.1 \sim \nu$
- *Noised tally:* $\tilde{T} = (\textit{no}: 1.2, \textit{yes}: 2.2)$
- *Corrected noised tally:* $(\textit{Winner:yes}, \tilde{T})$

Note that the noise magnitude, $0.1 < dW = 0.1 \times 3.4 = 0.34$*, as expected* 95% *of the time* ($q = 0.95$)*. Note also that while individual voter choices can be deduced with certainty from the raw tally, they cannot be from the noised tally. (E.g., which voter voted* no*?)*

Minimum Decisive Coalition (MDC). As a measure of the decentralization represented by DAO proposals in our experiments, we introduce a concept that we call the *Minimum Decisive Coalition* (MDC).

For a given proposal, the *MDC* is the smallest number of voters who could *change* their votes to flip the outcome. For instance, in Example 3, the MDC is 1, as either yes voter can unilaterally determine the outcome.

Two main factors drive MDC. First, the presence of whales: large token holders decrease the coalition size required to shift the outcome, thus raising MDC. Second, the closeness of the proposal: tighter margins reduce MDC.

8.2 Results

With this setup we now present three broad empirical results.

Result 1: Noised and winner-only tally improve B-Privacy. As shown in Figure 5, adding Laplace noise with a 10% tally perturbation increases B-privacy across all DAOs. The gains are substantially larger for DAOs with higher MDC. In particular we see a clear delineation in relative B-privacy of both winner and corrected noised tally algorithms between 4 DAOs with average MDC >10 at the bottom of the plot (Proof of Humanity, Beanstalk DAO, Aavegotchi and Shell Protocol) and the other DAOs which all have MDC < 5. Intuitively, when voting weight is more dispersed, a modest tally perturbation more effectively obscures any single voter's signal, whereas in low-MDC DAOs where voting weight is concentrated, the same 10% tally perturbation often fails to mask the choices of large holders ("whales"), so the privacy lift is smaller. Given that whales pervade the current DAO landscape, in most proposals, adding noise, or even revealing only the winner, only increases relative B-privacy by a modest factor.

Result 2: Higher noise and MDC raise B-Privacy. Figure 6 plots relative B-Privacy versus the tally perturbation $d \in [1\%, 100\%]$, grouping proposals by their Minimum Decisive Coalition (MDC) and averaging results across the Aavegotchi DAO ($n = 276$ proposals). Aggregating within MDC cohorts reveals two clear patterns: (i) B-Privacy rises monotonically with d; and (ii) for any fixed d, proposals with larger MDC—i.e., requiring larger swing coalitions and thus more decentralized—achieve higher B-Privacy. We focus on Aavegotchi because it offers both a wide range of MDC values and a large number of proposals, making the trends especially clear. This pattern holds broadly across DAOs: adding more noise consistently raises B-Privacy, while lower MDC tends to reduce it. However, as shown in Figure 5, most DAOs exhibit very low average MDC, which explains why relative B-Privacy remains limited even under high noise or private tallying.

Result 3: Whales are the optimal targets of bribery. An adversary's most effective strategy targets voters whose compliance can be verified with highest fidelity. Under the corrected noised tally, Theorem 10 shows smaller voters are less attractive both due to their lower pivotality and because when voters are not pivotal, adversaries must rely on distinguishing between noise distributions to infer votes. For smaller voters, their lower weight w_i means these noise distributions overlap more substantially, making it harder for adversaries to deter-

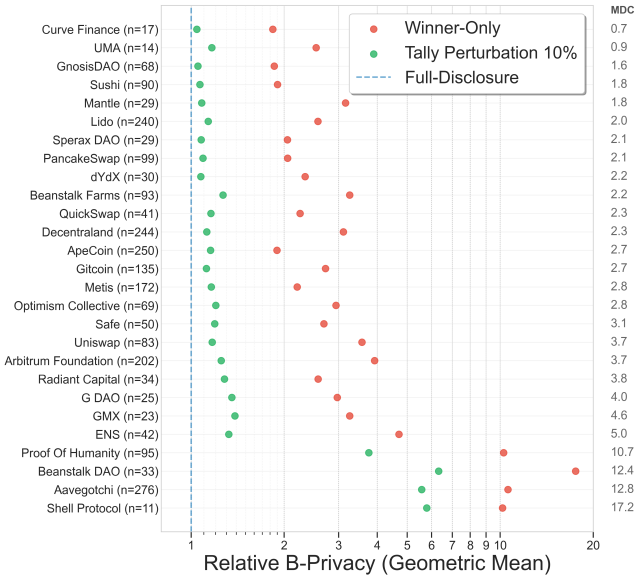

Figure 5: Relative B-Privacy by DAO under different Tally algorithms specified in Table 1. Results are averaged across proposals using geometric mean. Each row is a DAO, and rows are sorted using by the average MDC; red dots denote relative B-privacy in winner-only setting, green dots a lower bound on relative B-privacy in the corrected noised setting with tally perturbation $d = 10\%$. The dashed vertical line marks the full-disclosure baseline. The x-axis is logarithmic; right-hand labels give the average MDC per DAO, left hand labels give the amount of proposals per DAO (n).

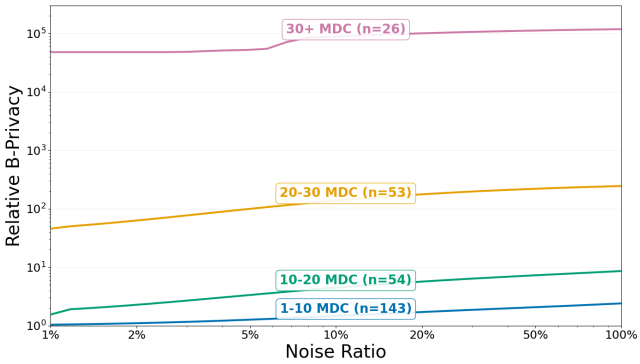

Figure 6: Relative B-Privacy as a function of tally perturbation d, grouped by minimum decisive coalition (MDC) size across 269 proposals from Aavegotchi [36]. Each line averages the relative B-Privacy across all proposals within an MDC cohort (n denotes the number of such proposals). The y-axis shows relative B-Privacy on a log scale.

mine which choice they made.We confirm this empirically: even under full-disclosure, only the largest voters are worth bribing—in our representative ApeCoin proposal[2] with 556 voters, only 16 receive bribes, even when the full disclosure tally algorithm is used. As tally algorithms disclose less information (moving from full-disclosure to corrected noised to winner-only), this number decreases further as smaller voters become effectively hidden by uncertainty. The theoretical prediction aligns with computational results, with whales remaining distinguishable while smaller voters become protected by noise. We illustrate this behavior in Figure 7; other proposals exhibit the same pattern.

Key Finding

Empirical analysis of B-privacy shows:

- Noised and winner-only tallying raise B-privacy in our experiments over real-world DAO data.
- B-privacy rises most sharply given a low Minimum Decisive Coalition (MDC)—generally resulting in a proposal from a close vote and/or presence of whale dominance.
- Whales become the primary targets of bribery under high noise. Adding tally noise (or winner-only release) effectively hides small voters, so optimal adversaries focus payments on high-weight voters.

8.3 Summary guidance for DAOs

In weighted voting, privacy is more than a voter-centric right—it is a structural defense against economic manipulation. Even when ballots remain hidden, precise results can leak enough information for adversaries to mount highly efficient bribery attacks. This undermines both decision integrity and community trust.

Drawing from our DAO dataset evaluation, several practices emerge as effective in boosting B-Privacy without unduly degrading transparency:

- **Apply corrected noised tally algorithms:** Even minimal Laplacian noise, tuned to keep results within 10% of the exact tally for most proposals, can raise B-Privacy by up to an order of magnitude. Although this varies widely depending on the weight distribution and proposal dynamics.
- **Adjust tally perturbation for proposal sensitivity:** Apply stronger tally perturbations for high-value or contentious proposals where privacy is at a premium, less perturbation for routine decisions.
- **Account for weight distribution:** Skewed token holdings increase privacy risks. Reducing whale dominance

[2]Proposal id: `0x1478808403c96b9ee0e7b2a97cc0215044ca45ac14b6cef2376bb526daedef86`

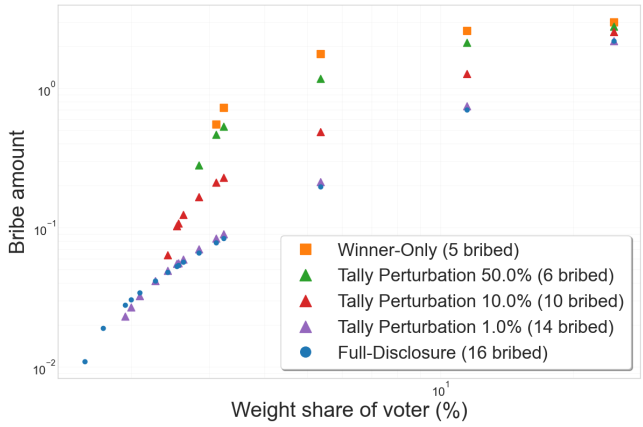

Figure 7: Optimal bribe distribution across voters for Winner, Corrected Noised, and Full-Disclosure tally algorithms. Plotted on representative proposal `0x1478808403c...` from ApeCoin with 556 voters and an MDC of 4. X-axis corresponds to weight shares, i.e., fraction of total weight, of individual voters. Y-axis shows bribe amount received by individual voters, measured in utility units. Where there is no dot on Y-axis for a given tally algorithm, the corresponding voter receives no bribe.

can increase MDC across proposals such that small tally perturbations cause sharp increases in B-privacy.

- **Focus anti-bribery measures on whales:** Since adversaries optimally target high-weight voters who offer the best bribe margins, governance systems should prioritize monitoring and protecting whale voters. Traditional privacy mechanisms primarily shield smaller voters, but whales remain the most attractive and vulnerable targets for manipulation regardless of noise levels.

In our experiments, modest noise preserved the key benefits of transparency—e.g., enabling voters to understand the magnitude of winning margins—while increasing the adversary's bribery cost significantly. The results suggest that strong B-Privacy and transparency are not inherently in conflict; with careful tuning, both can be achieved in practice.

9 Conclusion

We have introduced B-Privacy, a new metric for privacy in the weighted-voting setting, where classical voting-privacy notions such as ballot secrecy break down. B-Privacy measures the cost of bribery to an adversary induced by tally algorithms. It offers an economic lens on a key consequence of privacy in weighted voting: as adversaries gain more precise knowledge of voter behavior, their cost of bribery decreases, raising systemic security risks. Our work gives rise to a number of future research directions, among them:

- **Multi-choice proposals:** As initial work, our B-privacy framework currently supports only binary-choice proposals. Extending to abstention (as an explicit ballot choice) raises new issues around participation cost and community dynamics. More generally, multiple ballot choices require modeling far more complex strategic coordination and bribery schemes.
- **Analytical equilibrium characterization and robust optimization:** Our current approach relies on computational methods to find Bayesian Nash equilibria and calculus-based optimization for bribe allocation, which may converge to local minima. Analytical bounds on equilibrium behavior and more systematic exploration of allocation strategies could provide deeper theoretical insights and more robust B-privacy approximations.
- **Alternative governance mechanisms:** Our experimental results show how large whale presence (low MDC) in DAOs today raises the risk of bribery. Our work injects new urgency into the question of how DAOs can protect against whale dominance—a popular topic of study and community action [41, 44].
- **Parameter exploration and modeling assumptions:** Our experiments make simplifying assumptions, including normal utility distributions known to all players, and Laplacian noise for tally perturbations. Future work could explore more realistic utility models capturing voter heterogeneity and imperfect information, alternative noise distributions, and other modeling choices and parameters.

In summary, B-Privacy yields both theoretical results and practical guidance for DAO communities and other weighted-voting settings. With the growing reliance on weighted voting in blockchain governance and a a movement toward secret-ballot systems, as well as the mounting privacy-related threats to system integrity, B-Privacy promises to serve as serve a key metric for evaluating privacy weighted voting.

Ethical Considerations

Our research analyzes voting privacy in weighted systems through theoretical attacks on public DAO data and game-theoretic models of bribery. The primary stakeholders are DAO voters and governance participants, who benefit from understanding privacy limitations in current transparency practices and proposed countermeasures that better balance privacy and transparency. Parties not directly participating in governance but sharing an interest in its success, such as protocol teams, foundations, governance platforms, tooling providers, and token holders similarly benefit from a better understanding of the risks of bribery.

Our work poses minimal direct risks since we analyze only publicly available data and study counterfactual scenarios (what privacy would look like if DAOs adopted ballot secrecy). The attack methods we demonstrate are relatively

straightforward applications of well-known algorithmic techniques to public tallies. While adversaries could potentially use similar approaches to deanonymize past and future tallies, these techniques represent natural applications of subset-sum algorithms and basic weight-based inference that informed actors likely already recognize. Poor parameterization of the noising technique we propose could create false privacy assurances in highly centralized settings. However, the benefits of making this tool available clearly outweigh misapplication risks, and we explore effective parameterization in our empirical evaluation.

Our bribery analysis provides theoretical insights into optimal manipulation strategies, but operates under significant simplifying assumptions. Our assumptions around gaussian utility distributions, perfect information about weights and utilities, a constrained adversarial strategy space, and binary choices all limit direct applicability to real-world manipulation campaigns. The framework serves primarily to quantify privacy-bribery tradeoffs and inform defenses rather than as a practical attack guide. We use publicly available data in our evaluations so that our attacks on private voting are simulated and do not actually risk compromising the privacy of any users, and we mitigate harms around false privacy assurances by explicitly characterizing the settings in which noising is ineffective.

The decision to publish follows a clear beneficence analysis. DAO governance faces active manipulation threats, evidenced by existing vote-buying markets worth hundreds of millions of dollars. Our work addresses a critical gap: while these communities prioritize transparency, they lack frameworks for understanding when transparency undermines the very democratic values it aims to support.

By providing both theoretical analysis and practical countermeasures, our research enables informed decision-making about governance design. The proposed noised tally mechanisms offer concrete alternatives that preserve essential transparency while significantly increasing resistance to both privacy attacks and bribery. Our work contributes to the broader effort to mature DAO governance by highlighting fundamental tradeoffs and providing tools to navigate them responsibly.

Open Science

The code used to generate the results of the paper, as well as the dataset of votes from the Snapshot voting platform used by experiments throughout the paper can be found at https://zenodo.org/records/17965708. The repository is structured as follows:

- attack: guess_voters.cc contains the raw tally attack from Section 4 and the adapted version for noised tallies described in Section 7.1. Compiling and running the program outputs CSV-formatted result files that are processed by the scripts in the same directory to generate plots. plot_dao_aggregate_analysis.py generates Figure 1, plot_dao_analysis.py generates Figure 2 and plot_noise_comparison.py generates Figure 4.
- bribery_resistance: core.py contains the code used to compute B-privacy as described in Section 6 and Appendix B. The other scripts make use of the methods defined in this file to generate the plots from Section 8. batch_bribery_analysis.py generates Figure 5, bribery_cost_against_noise.py generates Figure 6 and bribe_distribution_proposal.py generates Figure 7.
- data_input: all_snapshot.zip is a zipped CSV file, referenced by the code in the other directories, that contains the Snapshot voting data used by experiments throughout.

References

[1] Dirk Achenbach, Carmen Kempka, Bernhard Löwe, and Jörn Müller-Quade. Improved coercion-resistant electronic elections through deniable re-voting. 2015.

[2] Toke S Aidt and Peter S Jensen. From open to secret ballot: Vote buying and modernization. *Comparative Political Studies*, 2017.

[3] Syed Taha Ali and Judy Murray. An overview of end-to-end verifiable voting systems. *Real-world electronic voting*, 2016.

[4] Liu Ao, Yun Lu, Lirong Xia, and Vassilis Zikas. How private are commonly-used voting rules? In *Conference on Uncertainty in Artificial Intelligence*. PMLR, 2020.

[5] Josh Benaloh and Dwight Tuinstra. Receipt-free secret-ballot elections. In *Proceedings of the twenty-sixth annual ACM symposium on Theory of computing*, 1994.

[6] Josh Daniel Cohen Benaloh. *Verifiable secret-ballot elections*. Yale University, 1987.

[7] Vitalik Buterin. Daos, dacs, das and more: An incomplete terminology guide. *Ethereum Blog*, 2014.

[8] Vitalik Buterin. Moving beyond coin voting governance. Technical report, 2021.

[9] David Chaum. Elections with unconditionally-secret ballots and disruption equivalent to breaking rsa. In *Workshop on the Theory and Application of of Cryptographic Techniques*. Springer, 1988.

[10] David Chaum. Random-sample voting. Technical report, 2016.

[11] Tassos Dimitriou. Efficient, coercion-free and universally verifiable blockchain-based voting. *Computer Networks*, 2020.

[12] Jannik Dreier, Pascal Lafourcade, and Yassine Lakhnech. Defining privacy for weighted votes, single and multi-voter coercion. In *Computer Security–ESORICS 2012*. Springer, 2012.

[13] John Duggan and César Martinelli. A bayesian model of voting in juries. *Games and Economic Behavior*, 2001.

[14] Charlott Eliasson and André Zúquete. An electronic voting system supporting vote weights. *Internet Research*, 2006.

[15] Andres Fabrega, Amy Zhao, Jay Yu, James Austgen, Sarah Allen, Kushal Babel, Mahimna Kelkar, and Ari Juels. Voting-Bloc Entropy: A New Metric for DAO Decentralization. In *USENIX Security*, 2025.

[16] Rainer Feichtinger, Robin Fritsch, Lioba Heimbach, Yann Vonlanthen, and Roger Wattenhofer. Sok: Attacks on daos. *arXiv preprint arXiv:2406.15071*, 2024.

[17] Mark Fey. Stability and coordination in duverger's law: A formal model of preelection polls and strategic voting. *American Political Science Review*, 1997.

[18] Alan S Gerber, Gregory A Huber, David Doherty, Conor M Dowling, and Seth J Hill. Do perceptions of ballot secrecy influence turnout? results from a field experiment. *American Journal of Political Science*, 2013.

[19] Noemi Glaeser, István András Seres, Michael Zhu, and Joseph Bonneau. Cicada: A framework for private non-interactive on-chain auctions and voting. *Cryptology ePrint Archive*, 2023.

[20] Panagiotis Grontas, Aris Pagourtzis, Alexandros Zacharakis, and Bingsheng Zhang. Towards everlasting privacy and efficient coercion resistance in remote electronic voting. In *International Conference on Financial Cryptography and Data Security*. Springer, 2018.

[21] Martin Hirt and Kazue Sako. Efficient receipt-free voting based on homomorphic encryption. In *International Conference on the Theory and Applications of Cryptographic Techniques*. Springer, 2000.

[22] Ellis Horowitz and Sartaj Sahni. Computing partitions with applications to the knapsack problem. *J. ACM*, 1974.

[23] Wojciech Jamroga, Peter B Roenne, Peter YA Ryan, and Philip B Stark. Risk-limiting tallies. In *International Joint Conference on Electronic Voting*. Springer, 2019.

[24] Ari Juels, Dario Catalano, and Markus Jakobsson. Coercion-resistant electronic elections. In *Proceedings of the 2005 ACM Workshop on Privacy in the Electronic Society*, pages 61–70, 2005.

[25] Sunny King and Scott Nadal. Ppcoin: Peer-to-peer crypto-currency with proof-of-stake. *self-published paper, August*, 2012.

[26] Ahmed Kosba, Dimitrios Papadopoulos, Charalampos Papamanthou, and Dawn Song. {MIRAGE}: Succinct arguments for randomized algorithms with applications to universal {zk-SNARKs}. In *29th USENIX Security Symposium (USENIX Security 20)*, pages 2129–2146, 2020.

[27] Jeffrey C Lagarias and Andrew M Odlyzko. Solving low-density subset sum problems. *Journal of the ACM (JACM)*, 1985.

[28] Kyounghun Lee and Frederick Dongchuhl Oh. Shareholder voting and efficient corporate decision-making. *Research in Economics*, 2024.

[29] Thomas Lloyd, Daire O'Broin, and Martin Harrigan. Emergent outcomes of the vetoken model. In *2023 IEEE international conference on omni-layer intelligent systems (COINS)*. IEEE, 2023.

[30] Wouter Lueks, Iñigo Querejeta-Azurmendi, and Carmela Troncoso. Voteagain: A scalable coercion-resistant voting system. In *USENIX Security*, 2020.

[31] William Mougayar. The dark dao threat: Vote vulnerability could undermine crypto elections. Accessed: 2025-08-23.

[32] Kamilla Nazirkhanova, Vrushank Gunjur, X Jesus, and Dan Boneh. Kite: How to delegate voting power privately. *arXiv preprint arXiv:2501.05626*, 2025.

[33] Aztec Network. Nounsdao private voting: Final update. https://aztec.network/blog/nounsdao-private-voting-final-update, 2023. Accessed: 2025-08-17.

[34] Shutter Network. Coming soon to daos: Permanent shielded voting via homomorphic encryption. https://blog.shutter.network/coming-soon-to-daos-permanent-shielded-voting-via-homomorphic-encryption/, 2025. Accessed: 2025-08-17.

[35] Ana Paula Pereira. Dark daos: Vitalik buterin, cornell researchers mitigate bribery threats. https://cointelegraph.com/news/dark-daos-vitalik-buterin-cornell-researchers-mitigate-bribery-threats. Accessed: 2025-08-23.

[36] Pixelcraft Studios Pte. Ltd. Aavegotchi — play fun games, earn real crypto. https://www.aavegotchi.com. Accessed 2025-08-21.

[37] Régis Renault and Alain Trannoy. The bayesian average voting game with a large population. *Economie publique/Public economics*, 2007.

[38] Joong Bum Rhim and Vivek K. Goyal. Keep ballots secret: On the futility of social learning in decision making by voting. https://arxiv.org/abs/1212.5855, 2012.

[39] Kazue Sako and Joe Kilian. Receipt-free mix-type voting scheme: A practical solution to the implementation of a voting booth. In *International Conference on the Theory and Applications of Cryptographic Techniques*. Springer.

[40] Daniel J. Seidmann. A theory of voting patterns and performance in private and public committees. *Social Choice and Welfare*.

[41] Tanusree Sharma, Yujin Potter, Kornrapat Pongmala, Henry Wang, Andrew Miller, Dawn Song, and Yang Wang. Unpacking how decentralized autonomous organizations (daos) work in practice. In *2024 IEEE International Conference on Blockchain and Cryptocurrency (ICBC)*, pages 416–424. IEEE, 2024.

[42] Joshua Tan, Tara Merk, Sarah Hubbard, Eliza R Oak, Helena Rong, Joni Pirovich, Ellie Rennie, Rolf Hoefer, Michael Zargham, Jason Potts, et al. Open problems in daos. *arXiv preprint arXiv:2310.19201*, 2023.

[43] Tatu. Shutter brings shielded voting to snapshot. https://blog.shutter.network/shutter-brings-shielded-voting-to-snapshot/, 2022.

[44] Zayn Wang, Frank Pu, Vinci Cheung, and Robert Hao. Balancing security and liquidity: A time-weighted snapshot framework for dao governance voting. https://arxiv.org/abs/2505.00888, 2025.

[45] E Glen Weyl. The robustness of quadratic voting. *Public choice*, 2017.

[46] Ziqi Yan, Jiqiang Liu, and Shaowu Liu. Dpwevote: differentially private weighted voting protocol for cloud-based decision-making. *Enterprise Information Systems*, 2019.

A Additional Theorems and Proofs

This appendix contains the formal proofs of theorems presented in Sections 6 and 7 which cover our analytic results for computing B-privacy and bounding B-privacy for noised tally algorithms.

A.1 Independence of bribery condition functions

Theorem 11 (Independence of bribery condition functions)**.** *For any bribery condition function $f : O \to \{0,1\}^n$ (possibly randomized and correlated across voters), there exists an equivalent independent strategy using condition functions $f_i : O \to \{0,1\}$ that achieves the same bribe margins α_i for all voters i, and hence the same B-privacy.*

Proof. Given any (possibly randomized, correlated) condition function f, construct independent functions by setting:

$$f_i(o) = f(o)_i \text{ for each voter } i \text{ and outcome } o$$

Crucially, each function f_i represents an independent invocation of the original correlated function f—not a single correlated invocation whose components are distributed across voters. This eliminates correlations while preserving the marginal payment probability for each voter.

The bribe margin for voter i under the correlated strategy is:

$$\Pr[f(\mathsf{tally}(t))_i = 1 \mid c_i = \text{yes}] - \Pr[f(\mathsf{tally}(t))_i = 1 \mid c_i = \text{no}]$$

For the independent strategy this becomes:

$$\Pr[f_i(\mathsf{tally}(t)) = 1 \mid c_i = \text{yes}] - \Pr[f_i(\mathsf{tally}(t)) = 1 \mid c_i = \text{no}]$$

By construction we have $\Pr[f(\mathsf{tally}(t))_i = 1] = \Pr[f_i(\mathsf{tally}(t)) = 1]$, so the bribe margin is the same in either case.

Since each voter's decision depends only on their individual expected payoff—which depends only on their own bribe margin and pivotality (Theorem 6)—the transformation preserves equilibrium behavior and adversarial success probability. □

A.2 Proof of optimal bribery condition functions

For clarity we prove the first part of Theorem 7 as a lemma:

Lemma 12 (Optimality of maximal bribe margin)**.** *For the adversary in the bribery game, the optimal bribery condition functions are those that maximize the bribe margin α_i for each voter i.*

Proof. Suppose for contradiction that there exists an optimal solution $(\mathbf{b}^*, \mathbf{f}^*)$ with some condition function f_k^* that does not maximize voter k's bribe margin. Let α_k denote the bribe margin of f_k^*, and let $\alpha_k^{\max} > \alpha_k$ be the maximal achievable bribe margin for voter k.

Since $\alpha_k^{\max} > \alpha_k$, we can write $\alpha_k^{\max} = a\alpha_k$ for some $a > 1$.

Consider an alternative strategy $(\mathbf{b}', \mathbf{f}')$ where:

- $f_i' = f_i^*$ for all $i \neq k$
- f_k' achieves the maximal bribe margin $\alpha_k^{\max}$

- $b'_i = b^*_i$ for all $i \neq k$
- $b'_k = \frac{b^*_k}{a}$

By Theorem 6, voter k's expected payoff from accepting the bribe is:

$$\alpha_k \cdot b^*_k = a\alpha_k \cdot \frac{b^*_k}{a} = \alpha^{\max}_k \cdot b'_k$$

Since the expected payoff is unchanged, voter k's equilibrium behavior remains the same. The behavior of all other voters is also unchanged, so the success probability remains p.

However, the total bribe budget decreases:

$$\sum_{i=1}^{n} b'_i = \sum_{i \neq k} b^*_i + b'_k < \sum_{i \neq k} b^*_i + b^*_k = \sum_{i=1}^{n} b^*_i$$

This contradicts the assumption that $(\mathbf{b}^*, \mathbf{f}^*)$ was optimal. □

We now proceed to considering the form of the optimal bribery condition functions:

Proof of Theorem 7. By Lemma 12, the optimal condition function for voter i is the one that maximizes the bribe margin α_i. We now characterize this function.

Define $p^o_{i,c} = \Pr[\mathsf{tally}(t) = o \mid c_i = c]$. For any condition function $f_i : O \to \{0,1\}$, consider the definition of the bribe margin, recalling that probabilities are taken over the random variable corresponding to voters' choices $\mathbf{c}$:

$$\begin{aligned}\alpha_i &= \Pr[f_i(\mathsf{tally}(t)) = 1 \mid c_i = \mathsf{yes}] - \Pr[f_i(\mathsf{tally}(t)) = 1 \mid c_i = \mathsf{no}] \\ &= \sum_{o \in O} f_i(o) p^o_{i,\mathsf{yes}} - \sum_{o \in O} f_i(o) p^o_{i,\mathsf{no}} \\ &= \sum_{o \in O} f_i(o) \left(p^o_{i,\mathsf{yes}} - p^o_{i,\mathsf{no}}\right)\end{aligned}$$

Since $f_i(o) \in \{0,1\}$, this sum is maximized by setting $f_i(o) = 1$ if and only if the term in parentheses is non-negative, which gives:

$$f^*_i(o) = \mathbb{I}\{p^o_{i,\mathsf{yes}} \geq p^o_{i,\mathsf{no}}\}.$$

This yields the optimal bribe margin:

$$\alpha^*_i = \sum_{o \in O} \max(p^o_{i,\mathsf{yes}} - p^o_{i,\mathsf{no}}, 0).$$

□

Corollary 7.1. *The optimal bribery condition functions for standard tally algorithms are (where $o = \mathsf{tally}(t)$ denotes the outcome):*

- $\mathsf{tally}_{\mathsf{winner}}$*:* $f^*_i(o) = \mathbb{I}\{o = \mathsf{yes}\}$ *with bribe margin* $\alpha^*_i = \Delta_i$
- $\mathsf{tally}_{\mathsf{public}}$*:* $f^*_i(o) = \mathbb{I}\{o_i = \mathsf{yes}\}$ *with bribe margin* $\alpha^*_i = 1$

Proof. We apply the optimal condition function form to each tally algorithm:

(1) Winner algorithm. Here $\mathsf{tally}(t) \in \{\mathsf{yes}, \mathsf{no}\}$ reveals only the winning choice. Since voting yes cannot make a yes outcome less likely, we have:

$$\Pr[\mathsf{tally}(t) = \mathsf{yes} \mid c_i = \mathsf{yes}] \geq \Pr[\mathsf{tally}(t) = \mathsf{yes} \mid c_i = \mathsf{no}].$$

By Theorem 7, the optimal condition function is $f_i(o) = \mathbb{I}\{o = \mathsf{yes}\}$.

The bribe margin is:

$$\begin{aligned}\alpha_i &= \Pr[\text{yes wins} \mid c_i = \mathsf{yes}] - \Pr[\text{yes wins} \mid c_i = \mathsf{no}] \\ &= \Pr[\text{no wins} \mid c_i = \mathsf{no}] - \Pr[\text{no wins} \mid c_i = \mathsf{yes}] \\ &= \Delta_i.\end{aligned}$$

(2) Full-disclosure algorithm. Here $\mathsf{tally}(t) \in \{\mathsf{yes}, \mathsf{no}\}^n$ reveals all individual votes. Since $\mathsf{tally}(t)_i = c_i$ always:

$$\begin{aligned}&\Pr[\mathsf{tally}(t)_i = \mathsf{yes} \mid c_i = \mathsf{yes}] = 1 \\ &> 0 = \Pr[\mathsf{tally}(t)_i = \mathsf{yes} \mid c_i = \mathsf{no}].\end{aligned}$$

By Theorem 7, the optimal condition function is $f_i(o) = \mathbb{I}\{o_i = \mathsf{yes}\}$ with bribe margin $\alpha_i = 1$. □

A.3 Bound on Plausible Deniability

Proof of Theorem 9. Let $p^o_{i,c} = \Pr[\mathsf{tally}(t) = o | c_i = c]$ and $p^{i,c}_o = \Pr[c_i = c | \mathsf{tally}(t) = o]$. Using Bayes' rule we have

$$p^o_{i,c} = \frac{p^{i,c}_o \Pr[\mathsf{tally}(t) = o]}{\Pr[c_i = c]}.$$

We also have that

$$\max(p^o_{i,\mathsf{yes}} - p^o_{i,\mathsf{no}}, 0) + \min(p^o_{i,\mathsf{yes}}, p^o_{i,\mathsf{no}}) = p^o_{i,\mathsf{yes}}.$$

Then we can finally claim

$$\begin{aligned}\alpha^*_i &= \sum_{o \in O} \max(p^o_{i,\mathsf{yes}} - p^o_{i,\mathsf{no}}, 0) \\ &= \sum_{o \in O} p^o_{i,\mathsf{yes}} - \sum_{o \in O} \min\left(p^o_{i,\mathsf{yes}}, p^o_{i,\mathsf{no}}\right) \\ &= \sum_{o \in O} \frac{p^{i,\mathsf{yes}}_o}{\Pr[c_i = \mathsf{yes}]} \Pr[\mathsf{tally}(t) = o] \\ &\quad - \sum_{o \in O} \min\left(\frac{p^{i,\mathsf{yes}}_o}{\Pr[c_i = \mathsf{yes}]}, \frac{p^{i,\mathsf{no}}_o}{\Pr[c_i = \mathsf{no}]}\right) \Pr[\mathsf{tally}(t) = o] \\ &= 1 - EPD^{\mathsf{tally}}_i.\end{aligned}$$

□

A.4 Bound on corrected noised tally bribe margin

Proof of Theorem 10. We consider the tally algorithm $\mathsf{tally}_{\mathsf{noised}(\nu)+}(t) = (\mathsf{tally}_{\mathsf{noised}(\nu)}(t), \mathsf{tally}_{\mathsf{winner}}(t))$.

Throughout this proof we abbreviate notation, writing $\mathsf{tally}_{\mathsf{n}+}$ for $\mathsf{tally}_{\mathsf{noised}(\nu)+}$, $\mathsf{tally}_{\mathsf{n}}$ for $\mathsf{tally}_{\mathsf{noised}(\nu)}$, $\mathsf{tally}_{\mathsf{w}}$ for $\mathsf{tally}_{\mathsf{winner}}$, and $\Pr[\mathsf{tally} = o \mid c]$ for $\Pr[\mathsf{tally}(t) = o \mid c_i = c]$.

By Theorem 7, for any voter i when using the optimal bribery condition function, the bribe margin is:

$$\alpha_i^* = \sum_{o \in O} \max\left(\Pr\left[\mathsf{tally}_{\mathsf{n}+} = o \mid \mathsf{yes}\right] - \Pr\left[\mathsf{tally}_{\mathsf{n}+} = o \mid \mathsf{no}\right], 0\right)$$

Since the combined outcome pairs (o_1, o_2) where o_1 is the noised tally and $o_2 \in \{\mathsf{yes}, \mathsf{no}\}$ is the winner, we decompose the sum over all possible outcomes:

$$\alpha_i^* = \sum_{o_2 \in \{\mathsf{yes},\mathsf{no}\}} \int_{-\infty}^{\infty} \max(\Pr\left[\mathsf{tally}_{\mathsf{n}} = o_1, \mathsf{tally}_{\mathsf{w}} = o_2 \mid \mathsf{yes}\right] - \Pr\left[\mathsf{tally}_{\mathsf{n}} = o_1, \mathsf{tally}_{\mathsf{w}} = o_2 \mid \mathsf{no}\right], 0)\, do_1.$$

Let $\mathbb{T}$ be the set of all possible sums of weights of voters that choose yes excluding voter i, and T be the random variable for this value with probability taken over all other voters' choices. Let $Z \sim \nu$ be the noise random variable. Define the winner function:

$$\mathsf{winner}(x) = \begin{cases} \mathsf{yes} & \text{if } x \geq \frac{W}{2} \\ \mathsf{no} & \text{otherwise.} \end{cases}$$

We condition on the partial tally T to decompose each probability. When voter i votes yes, the total yes tally is $s + w_i$; when voting no, it is s:

$$\Pr\left[\mathsf{tally}_{\mathsf{n}} = o_1, \mathsf{tally}_{\mathsf{w}} = o_2 \mid \mathsf{yes}\right] = \sum_{s \in \mathbb{T}} \Pr[T = s] \Pr\left[Z = o_1 - (s + w_i)\right] \cdot \mathbb{I}\left\{o_2 = \mathsf{winner}(s + w_i)\right\}.$$

$$\Pr\left[\mathsf{tally}_{\mathsf{n}} = o_1, \mathsf{tally}_{\mathsf{w}} = o_2 \mid \mathsf{no}\right] = \sum_{s \in \mathbb{T}} \Pr[T = s] \Pr\left[Z = o_1 - s\right] \cdot \mathbb{I}\left\{o_2 = \mathsf{winner}(s)\right\}.$$

For clarity, define:

$$P_y(s, o_1, o_2) = \Pr[Z = o_1 - (s + w_i)] \cdot \mathbb{I}\{o_2 = \mathsf{winner}(s + w_i)\},$$
$$P_n(s, o_1, o_2) = \Pr[Z = o_1 - s] \cdot \mathbb{I}\{o_2 = \mathsf{winner}(s)\}.$$

For brevity, we omit the arguments (s, o_1, o_2) when the context is clear.

Substituting back into the bribe margin calculation and applying the inequality $\max\left(\sum_s f(s), 0\right) \leq \sum_t \max\left(f(s), 0\right)$:

$$\begin{aligned}
\alpha_i^* &= \sum_{o_2 \in \{\mathsf{yes},\mathsf{no}\}} \int_{-\infty}^{\infty} \max\left(\sum_{s \in \mathbb{T}} (P_y - P_n) \Pr\left[T = s\right], 0\right) do_1 \\
&\leq \sum_{o_2 \in \{\mathsf{yes},\mathsf{no}\}} \int_{-\infty}^{\infty} \sum_{s \in \mathbb{T}} \max\left((P_y - P_n) \Pr\left[T = s\right], 0\right) do_1 \\
&= \sum_{o_2 \in \{\mathsf{yes},\mathsf{no}\}} \int_{-\infty}^{\infty} \sum_{s \in \mathbb{T}} \max\left(P_y - P_n, 0\right) \Pr\left[T = s\right] do_1 \\
&= \sum_{s \in \mathbb{T}} \Pr\left[T = s\right] \sum_{o_2 \in \{\mathsf{yes},\mathsf{no}\}} \int_{-\infty}^{\infty} \max\left(P_y - P_n, 0\right) do_1.
\end{aligned}$$

We now decompose the sum over partial tallies based on whether voter i is pivotal.

First consider the partial tallies in which voter i is pivotal. Let $\mathbb{T}^{\mathsf{pivotal}} = \{s \in \mathbb{T} : \frac{W}{2} - w_i < s < \frac{W}{2}\}$. For $s \in \mathbb{T}^{\mathsf{pivotal}}$, voter i's choice determines the winner: $\mathsf{winner}(s + w_i) \neq \mathsf{winner}(s)$. This means that for any fixed o_2, exactly one of the indicators $\mathbb{I}\{o_2 = \mathsf{winner}(s + w_i)\}$ or $\mathbb{I}\{o_2 = \mathsf{winner}(s)\}$ equals 1, so exactly one of P_y and P_n is nonzero. Therefore $\max(P_y - P_n, 0) = P_y$, and we have:

$$\begin{aligned}
&\sum_{s \in \mathbb{T}^{\mathsf{pivotal}}} \Pr[T = s] \sum_{o_2 \in \{\mathsf{yes},\mathsf{no}\}} \int_{-\infty}^{\infty} \max(P_y - P_n, 0)\, do_1 \\
&= \sum_{s \in \mathbb{T}^{\mathsf{pivotal}}} \Pr[T = s] \sum_{o_2 \in \{\mathsf{yes},\mathsf{no}\}} \int_{-\infty}^{\infty} P_y\, do_1 \\
&= \sum_{s \in \mathbb{T}^{\mathsf{pivotal}}} \Pr[T = s] \\
&= \Pr[\text{voter } i \text{ is pivotal}] = \Delta_i.
\end{aligned}$$

Now consider the partial tallies in which voter i is not pivotal. In these cases, $\mathsf{winner}(s + w_i) = \mathsf{winner}(s)$, so for any fixed o_2, the indicators $\mathbb{I}\{o_2 = \mathsf{winner}(s + w_i)\}$ and $\mathbb{I}\{o_2 = \mathsf{winner}(s)\}$ are either both 1 or both 0. We need only consider the case where both equal 1, allowing us to drop the sum over o_2:

Let $q(u) = \Pr[Z = u - w_i] - \Pr[Z = u]$. Then:

$$\begin{aligned}
&\sum_{s \in \mathbb{T} \setminus \mathbb{T}^{\mathsf{pivotal}}} \Pr[T = s] \sum_{o_2 \in \{\mathsf{yes},\mathsf{no}\}} \int_{-\infty}^{\infty} \max(P_y - P_n, 0)\, do_1 \\
&= \sum_{s \in \mathbb{T} \setminus \mathbb{T}^{\mathsf{pivotal}}} \Pr[T = s] \int_{-\infty}^{\infty} \max(q(o_1 - s), 0)\, do_1 \\
&= \sum_{s \in \mathbb{T} \setminus \mathbb{T}^{\mathsf{pivotal}}} \Pr[T = s] \int_{-\infty}^{\infty} \max(q(u), 0)\, du \\
&= (1 - \Delta_i) \int_{-\infty}^{\infty} \max(q(u), 0)\, du.
\end{aligned}$$

Combining both sets of partial tallies and substituting back $q(u) = \Pr[Z = u - w_i] - \Pr[Z = u]$ completes the proof:

$$\alpha_i^* \leq \Delta_i + (1 - \Delta_i) \int_{-\infty}^{\infty} \max\left(\Pr[Z = u - w_i] - \Pr[Z = u], 0\right) du$$

□

B Computational Methods for B-Privacy

This appendix details the numerical methods used to compute B-privacy values $B_{\mathsf{tally}}(p)$ for a tally algorithm tally at target success probability p. We use fixed-point iteration to approximate the Bayesian Nash equilibrium and Lagrange multipliers with numerical search for bribe allocation optimization. For each proposal, the procedure returns the minimal total bribe B^* and an associated per-voter bribe vector $\mathbf{b}^*$ such that $p^{\mathcal{A}}_{\mathsf{succ}} \geq p$

Inputs and utility model. Let $\mathbf{w} = (w_1, \cdots, w_n)$ be voter weights, $W = \sum_i w_i$ the total voting weight and $\mathbf{c} = (c_1, \cdots, c_n)$ the observed voter choices on a given proposal. For computational tractability, in our experiments, we model each voter i's utility distribution $\mathbf{U}_i^{\mathsf{no}}$ as Gaussian, although other distributions could be used. We set the prior based on their observed vote: voters who voted for the winning choice are modeled as having higher utility for "no" ($\mu_i = +1$), while voters who voted for the preferred outcome have $\mu_i = -1$. Formally:

$$\mathbf{U}_i^{\mathsf{no}} \sim \mathcal{N}(\mu_i, \sigma^2), \quad \mu_i = \begin{cases} +1 & \text{if } c_i = \mathsf{winner}, \\ -1 & \text{otherwise}, \end{cases} \qquad \sigma = 1.$$

This models the intuition that voters who voted against the adversary's preference (which we always model as the losing side) likely have higher intrinsic utility for that outcome.

Tally algorithm-dependent bribe margins. For each tally algorithm tally, we compute per-voter bribe margins α_i as follows: **Full-disclosure** ($\mathsf{tally}_{\mathsf{public}}$)**:** $\alpha_i = 1$, using the exact bribe margin from Corollary 7.1. **Winner-only** ($\mathsf{tally}_{\mathsf{winner}}$)**:** $\alpha_i = \Delta_i$, using the exact bribe margin from Corollary 7.1.**Corrected noised tally** ($\mathsf{tally}_{\mathsf{noised}(\nu)+}$)**:** $\alpha_i = \Delta_i + (1 - \Delta_i) \cdot \mathrm{TV}_i$, using the upper-bound on bribe margin from Theorem 10. $\mathrm{TV}_i = 1 - e^{-\beta w_i/2}$ is the value of the total variation distance integral, with β being the noise parameter for Laplace($1/\beta$) noise.

Note the for corrected noised tally setting the bribe margin to an upper bound means we compute a lower bound on B-privacy by Theorem 7.

Fixed-point computation of equilibrium. Given a bribe vector $\mathbf{b}$ and bribe margins $\alpha = (\alpha_i)$, Theorem 6 implies that in equilibrium, the probability that voter i votes for the adversary's preferred outcome is:

$$p_i = \Phi\left(\frac{\alpha_i b_i/\Delta_i - \mu_i}{\sigma}\right),$$

where Φ is the standard normal CDF and Δ_i is voter i's pivotality. The pivotality vector $\Delta = (\Delta_i)$ must satisfy the equilibrium condition: when all other voters play according to the probabilities $(p_j)_{j \neq i}$, voter i's pivotality is:

$$F_i(\Delta) = \Pr\left[\sum_{j \neq i} w_j X_j \in [W/2 - w_i, W/2)\right], \quad X_j \sim \text{Bernoulli}(p_j).$$

We compute Δ as the fixed point $\Delta = F(\Delta)$, which corresponds to the Bayesian Nash equilibrium condition that no voter wants to deviate given others' equilibrium behavior.

To evaluate the distribution $\sum_{j \neq i} w_j X_j$, we use Monte Carlo with common random numbers and antithetic variates to reduce variance. While costlier than Gaussian approximations, this avoids central limit theorem regularity requirements and yields stable accuracy even under extreme weight disparity.

We iterate $\Delta^{(t+1)} = \alpha F(\Delta^{(t)}) + (1 - \alpha)\Delta^{(t)}$ with under-relaxation $\alpha = 0.7$ until convergence.

Optimal bribe allocation. We maximize the expected yes-weight with $p_i(b_i) = \Phi(z_i)$, $z_i = \frac{\alpha_i b_i/\Delta_i - \mu_i}{\sigma}$. The Lagrangian is

$$\mathcal{L}(b, \lambda) = -\sum_i w_i\, p_i(b_i) + \lambda\Big(\sum_i b_i - B\Big).$$

For any active i ($b_i > 0$), KKT stationarity gives

$$\lambda \;=\; w_i \frac{\partial p_i}{\partial b_i} = w_i \frac{\alpha_i}{\Delta_i \sigma}\, \phi(z_i) \quad \Longrightarrow \quad \phi(z_i) = \frac{\lambda \Delta_i \sigma}{w_i \alpha_i}.$$

Since $\phi(z) = \frac{1}{\sqrt{2\pi}} e^{-z^2/2}$, inverting (absorbing constants into λ) yields

$$z_i = \sqrt{-2 \ln\left(\frac{\lambda}{w_i \alpha_i}\right)_+}, \qquad x_+ := \max\{x, 0\}.$$

Back-substituting $z_i = \frac{\alpha_i b_i/\Delta_i - \mu_i}{\sigma}$ gives

$$b_i(\lambda) = \frac{\Delta_i}{\alpha_i}\left(\mu_i + \sigma \sqrt{-2 \ln\left(\frac{\lambda}{w_i \alpha_i}\right)_+}\right).$$

In the centered case ($\mu_i = 0$) this is

$$\boxed{b_i(\lambda) = \frac{\Delta_i}{\alpha_i}\, \sigma \sqrt{\max\left\{0, -2 \ln\left(\frac{\lambda}{w_i \alpha_i}\right)\right\}}}$$

with λ chosen so $\sum_i b_i(\lambda) = B$. We are working within a non-convex landscape, which means that the solutions obtained using this method are locally optimal rather than guaranteed globally optimal.

Success probability estimation. Given $(\mathbf{w}, \Delta, \alpha, \mathbf{b})$, we compute per-voter success probabilities via the equilibrium condition above and estimate $p^{\mathcal{A}}_{\mathsf{succ}}$ by Monte Carlo with variance reduction techniques (common random numbers and antithetic variates). We use $R = 10{,}000$ samples by default, increasing to 10^6 if the estimate is within 0.01 of the target probability p.