Sedna: Sharding transactions in multiple concurrent proposer blockchains

Alejandro Ranchal-Pedrosa[1], Benjamin Marsh*[1,2], Lefteris Kokoris-Kogias[3], and Alberto Sonnino[3,4]

[1]Sei Labs
[2]University of Portsmouth
[3]Mysten Labs
[4]University College London

December 2025

Abstract

Modern blockchains increasingly adopt multi-proposer (MCP) consensus to remove single-leader bottlenecks and improve censorship resistance. However, MCP alone does not resolve how users should disseminate transactions to proposers. Today, users either naively replicate full transactions to many proposers, sacrificing goodput and exposing payloads to MEV, or target few proposers and accept weak censorship and latency guarantees. This yields a practical trilemma among censorship resistance, low latency, and reasonable cost (in fees or system goodput).

We present Sedna, a user-facing protocol that replaces naive transaction replication with verifiable, rateless coding. Users privately deliver addressed symbol bundles to subsets of proposers; execution follows a deterministic order once enough symbols are finalized to decode. We prove Sedna guarantees liveness and *until-decode privacy*, significantly reducing MEV exposure. Analytically, the protocol approaches the information-theoretic lower bound for bandwidth overhead, yielding a 2–3× efficiency improvement over naive replication. Sedna requires no consensus modifications, enabling incremental deployment.

1 Introduction

Multiple concurrent proposer (MCP) consensus mitigates single leader bottlenecks by allowing many validators to propose per slot. This improves bandwidth utilization and removes the acute impact of a slow or offline leader. Yet MCP

*ben@seinetwork.io

alone does not settle how a user should disseminate their transaction to proposers, and *dissemination* plays a key role in the user's experience of latency, price, and censorship exposure.

An important argument for blockchain systems to resort to MCP instead of single proposer consensus protocols is the native tolerance to censorship by a subset of the proposers. In MCP, it is straight-forward for users to send the same transaction to a number of the proposers and obtain censorship resistance through naive replication, whereas the closest approximation of this approach in single proposer systems involves the user sending the same transaction iteratively to the next proposer until it is included. It is also well-known that this replication comes at a significant impact on goodput: if all users want to deterministically tolerate censorship from c proposers without it incurring a cost on latency, they each need to send their respective transactions to at least $c + 1$ validators, incurring a worst-case goodput decrease by the replication factor of $c+1$ (e.g., a decrease of 80% in goodput to tolerate just 5 censoring proposers).

Some systems enforce deduplication by tuning a global parameter that navigates the trade-off between censorship resistance and goodput for all transactions. Beyond system-wide deduplication efforts, a final trade-off remains: how much of the replication bandwidth cost the system absorbs (i.e., how vulnerable it becomes to DoS attacks on goodput) versus how much users pay themselves (i.e., how costly censorship resistance becomes for them). In this sense, goodput impacts can be equated to pricing considerations for users. It is for this reason that we speak of a *trilemma of three user-facing properties*: censorship resistance (in that the transaction eventually gets included), low latency (in that the transaction gets included as soon as there is capacity for it), and reasonable pricing (either in goodput or economic terms, and whether incurred to the system or user).

MEV and dissemination latency. Some users or systems may tolerate losing one of the three properties for specific transactions, but many applications (e.g., trading) require all three, because latency effectively counts as censorship: a brief delay can wipe out an opportunity even if the transaction eventually lands on-chain. Even short delays before inclusion are economically exploitable as MEV. Under MCP, the lever extends from in-block ordering to *who* first learns a transaction and *when*. Replicating broadly increases leakage surface; targeting a single proposer is cheaper but trivially censorable. Our goal is to minimize pre-inclusion leakage while keeping byte cost low.

Our approach. We present Sedna, a dissemination protocol that replaces whole-transaction replication with *verifiable, ratelessly coded symbols* addressed to a subset of proposers ("lanes"). The sender commits to the payload, derives a transaction identifier txID, and generates an unbounded stream of small coded symbols, each tied to the transaction via signatures. These symbols are packaged into *addressed bundles* and privately delivered to a sampled set of lanes. Inclusion occurs as soon as enough *distinct verified symbols* for txID appear in

finalized history to reconstruct the payload; at that point the payload is decoded and the transaction is executed. Order is *deterministic*: transactions are sorted by $(\mathrm{ht}_{\mathrm{incl}}(\mathsf{txID}), \mathsf{txID})$, where $\mathrm{ht}_{\mathrm{incl}}(\mathsf{txID})$ is the first height at which decoding succeeds and the commitment opening verifies. Crucially, Sedna is entirely user-facing: each sender independently selects encoding parameters to navigate the trilemma according to their transaction's specific requirements, without any protocol-level changes to the underlying MCP consensus.

Benefits. For large payloads, the byte overhead of Sedna approaches $\frac{1+\varepsilon}{1-c_e/n}$, where $\varepsilon > 0$ is the small coding overhead of the rateless scheme (e.g., 5%), n is the number of proposers, and c_e is the *effective* number of censoring proposers the user wishes to tolerate. This matches the information-theoretic lower bound for any deterministic coding scheme up to the $(1 + \varepsilon)$ factor, and is far below the m-fold cost of naive replication. Our evaluation demonstrates that for medium-to-large payloads, Sedna reduces bandwidth overhead by a factor of 2–3× compared to naive replication under typical censorship assumptions, with the gap widening as payload size increases. A user selects how many lanes to contact and how many symbols to place in each bundle so as to meet a target per-slot failure probability δ. Because symbols are verifiable and payload-hiding until decode, Sedna provides *until-decode privacy* such the adversary's early decode probability remains near zero for properly chosen parameters, substantially reducing the pre-inclusion MEV surface compared to transparent mempool dissemination. For completeness, we also analyze two simpler submission variants, naive replication and fixed-rate MDS coding, and identify the parameter regimes where each approach is bandwidth-optimal.

Contributions.

- **Protocol.** We introduce Sedna, a user-facing submission layer for MCP that replaces whole-transaction replication with commitment-bound, addressed bundles of verifiable coded symbols; it realizes decode-to-include semantics and a deterministic execution order in a lazy-execution setting, while retaining pay-for-bytes admission.
- **Correctness.** We prove header/commitment non-malleability and deterministic resolution of symbol indices, and show that monotone decoding yields a unique payload from any K verified symbols for honestly encoded transactions (where K is the decode threshold) except with negligible decoding failure probability δ_{code}.
- **Liveness and censorship resistance.** We derive per-slot inclusion bounds and give closed-form, conservative prescriptions for how many lanes to contact (and how many symbols to place per bundle) to achieve a target failure probability δ against an assumed effective censor mass c_e.
- **Privacy (and MEV).** We formalize "until-decode" confidentiality, proving that pre-decode leakage is bounded to the bits contained in revealed

symbols and the public header, and we quantify an adversary's early-decode probability (including passive eavesdropping). We discuss how these bounds reduce the pre-inclusion MEV surface and interact with deterministic ordering by $(\mathrm{ht}_{\mathrm{incl}}(\mathsf{txID}), \mathsf{txID})$.

- **Cost and comparisons.** We compute exact byte costs including header and per-symbol metadata, derive asymptotic overhead floors approaching $\frac{1+\varepsilon}{1-c_e/n}$, and compare Sedna to naive replication and fixed-rate MDS coding, identifying parameter regimes where each approach is bandwidth-optimal, both theoretically and empirically.

Roadmap. Section 2 explores the background and related work. Section 3 introduces the system model. Section 4 specifies Sedna. Section 5 proves safety, liveness, and privacy. Section 6 analyzes byte overhead and parameter selection, and Section 7 evaluates representative design points. We finally conclude in Section 8.

2 Background and Related Work

Recent work establishes that the latency cost of censorship resistance is fundamental: any protocol achieving this guarantee requires 4 rounds in synchrony and 5 rounds in partial synchrony, exactly two rounds more than classic leader-based Byzantine Broadcast [1]. The MCP framework [20] achieves these bounds, making it the canonical architecture for blockchains that take censorship resistance seriously. Our work complements this foundation by addressing the *bandwidth* dimension of the problem: given that MCP-style dissemination is required for strong censorship guarantees, how can users minimize the byte overhead of ensuring their transactions reach sufficiently many proposers?

MCP approaches. A large class of protocols finalize *sets* or vectors of proposals per round or slot and are therefore natural instances of MCP [21, 30, 24, 15, 41, 22, 3]. MCP reduces the *single-leader veto* because multiple proposers can carry a user's data in a slot. Besides a mention to a direct trade-off between deduplication and censorship resistance [15, 42] through naive replication, these MCP protocols mainly focus on consensus properties assuming transactions are already in the proposers' mempools; they do not natively optimize the user-side byte cost of reaching many lanes, nor do they provide quantitative knobs for per-slot inclusion probability under an assumed censor mass [30, 15, 41]. For instance, transactions in the Sui blockchain [32] are naturally duplicated up to 5 times [33], which reduces the system's goodput by a significant factor. Moreover, the system offers no principled mechanism that prevents a user from submitting the same transaction to all validators, and such a submission can cause the transaction to be processed many times.

MCP in production. Nevertheless, the trilemma is unavoidable, with MCP systems in production taking different approaches to tackle it. The most prevalent approach implies relying on trusted RPC servers relaying transactions in a way that it enforces the deduplication target of the system. This approach, besides being relying on trusted, centralized entities, does not entirely solve the problem of resolving who is to absorb the cost of duplication, users or the system. Sui, Aptos, Red-Belly decide to absorb potential duplications as cost to the system. Hedera instead charges this cost to users, that need to pay for the total cost of any duplication of their transaction, which places the trilemma between latency, censorship resistance and cost in user terms and not system's goodput terms. It is also worth noting that Hedera cannot guarantee full protection from incurred costs on the systems due to the asynchronous execution model, in that a user may incur a cost greater than the funds the system can take from its accounts. Filecoin, also technically an MCP system, does not rely on RPC servers and remains trustless, with a replication factor average 3-4, and some transactions being duplicated as much as 9 times[1], whose cost is entirely absorbed by the system.

One could argue that some MCP designs pursue alternative strategies to sidestep the trilemma while preserving privacy and low latency. However, in nearly all deployed MCP variants, transparent mempools expose transactions before proposal, which enables pre-inclusion leakage and facilitates classic MEV behaviors such as front-running, sandwich attacks, and strategic reordering [13]. In effect, MCP enhances the *opportunity* for inclusion, but current implementations still lack a user-facing, bandwidth-aware submission layer that enforces privacy until decode and offers explicitly tunable liveness guarantees.

Single-leader systems. Single-leader, per-slot architectures concentrate inclusion authority in a single leader for each slot [7, 44]. To mitigate MEV, Ethereum has deployed a *Proposer-Builder Separation* through MEV-Boost and private relay markets, and is actively investigating in-protocol PBS mechanisms [19, 16]. Complementary private order-flow channels, such as Flashbots Protect, bloXroute's private transactions, and MEV-Blocker, postpone transaction disclosure until the builder phase [18, 4, 12]. In parallel, threshold and time-lock encryption schemes seek to conceal transaction contents until after ordering completes [39].

These mitigations rely on off-protocol markets and trusted relay behaviors, complicate decentralization (builder concentration), and do not resolve the fundamental *single inclusion bottleneck*: a leader can still censor or strategically delay [19, 13, 16]. Moreover, private relays reduce but do not eliminate leakage (e.g., side-channels, orderflow auctions) and provide no user-level control over bandwidth amortization across multiple inclusion opportunities.

[1] `Source:https://filfox.info/en/message/bafy2bzacebyo7g3tugujmshqnxwlqvp3qxl2omxk4bol6arp4ah76lme6nyzg?t=1`

Order-fairness and encryption. Two main lines appear beyond PBS/relays. First, *order-fairness* protocols (Aequitas [26], Themis [25], SpeedyFair [31]) attempt to align ordering with network timing to reduce extractable value from reordering. Second, *encryption-based* schemes [36, 5, 10] hide contents (threshold/timelock encryption, TEEs), often trading off latency/complexity and introducing trust or liveness assumptions [39]. Separately, batch-auction designs (e.g., frequent batch auctions and on-chain batch DEXs) reduce the value of marginal reordering [6, 11]. None of these lines simultaneously addresses (i) the user's byte-budget for multi-lane dissemination, (ii) quantitative, per-slot liveness against a target censor mass, and (iii) deterministic post-decode execution semantics.

Verifiable information dispersal. Asynchronous Verifiable Information Dispersal (AVID) [8] allows a dealer to disperse data to n parties such that any $t+1$ honest parties can reconstruct, while guaranteeing the dealer cannot equivocate. In the VSS/VID literature this is often phrased as a *binding* property: by the time the first honest party completes the dissemination protocol, the dealer is *bound* to a single value, and cannot later make reconstruction yield a different (even invalid) value depending on additional information such as randomness or subsequent blocks.

While AVID provides binding and consistency, meaning every reconstruction yields the same output, it does not ensure that the dispersed data carries semantic meaning. Furthermore, classic AVID schemes are overkill for our setting as they require the dealer to commit to the entire data bundle beforehand. Sedna relaxes this constraint while maintaining a VID-style binding guarantee through signature-based bundle verification, per-index deduplication, and a deterministic decoding rule defined over the on-chain total order of symbols.

Crucially, we use the consensus output to determine which subset of data is used for decoding, allowing the sender to continue encoding and committing to new symbols on the fly. This is particularly useful for the rateless encoding version of Sedna, where the sender can dynamically adapt the number of lanes used based on how quickly initial symbols are included on-chain. This flexibility allows Sedna to dynamically balance the required latency, fees, and censorship resistance. We intentionally avoid proving that dispersed data is well formed; instead, we replace AVID-style correctness proofs with economic deterrence, as malicious senders incur bandwidth fees for every published bundle.

Replication approaches. At the network layer, Reed–Solomon/MDS codes split a payload of size S into m shares such that any k suffice to reconstruct [35, 43]. In practice, high-throughput deployments and recent prototypes apply erasure-coded, tree-based or multipath fanout at the consensus dissemination layer to accelerate block propagation and improve robustness under partial synchrony [40, 9, 38, 27, 14]. In contrast, LT/Raptor/RaptorQ codes generate an unbounded stream of small symbols with near-linear decoding and tiny overhead $(1+\varepsilon)$ [28, 37, 29, 34].

In data availability systems, erasure coding and sampling are typically used for *post-proposal* data availability rather than user-side dissemination to multiple proposers [2, 17, 14, 23].

Overall, these mechanisms are promising but operate at the system's consensus dissemination or data availability layers; and they are thus orthogonal to the user-facing submission protocol we propose with Sedna.

3 System Model and Preliminaries

Consensus. We fix an integer $n \geq 3$ and a validator set $\mathcal{V} = \{1, \ldots, n\}$. Time proceeds in slots $t \in \mathbb{N}$. In each slot, every validator $i \in \mathcal{V}$ proposes a block B_i^t; a vector-commit consensus abstraction finalizes, at slot height t, an ordered n-tuple $(B_1^t, \ldots, B_n^t)$. We refer to the conceptual stream of proposals from validator i as a lane. We assume execution is *lazy* that is to say that the system commits to data and order first, and only executes transactions after finality has been reached.

Network. We assume a partially synchronous network model, with the adversary initially being able to delay messages arbitrarily. There is an unknown Global Stabilization Time (GST) after which all messages sent by honest parties are guaranteed to be delivered by honest parties within a fixed bound Δ. The liveness of the underlying vector-consensus abstraction (i.e., its ability to finalize new blocks) depends on this assumption. Safety properties hold at all times, while liveness properties are guaranteed to hold after GST. Our analysis assumes that bundles have already been delivered to target validators' mempools; the mechanics of user-to-validator dissemination are orthogonal to the protocol's core guarantees.

Good case. We additionally reason about particular properties of inclusion of a transaction submitted by an honest user post-GST. In this "good case", to isolate consensus-layer liveness, we assume that the user's bundles (sent at or before $t-\Delta$) have been successfully delivered to the target validators' mempools, and that validators have sufficient blockspace for all delivered, valid bundles. This removes fee-market and block-packing effects from the analysis, allowing us to evaluate the protocol's intrinsic properties of censorship resistance, low latency, and reduced duplication factor. These assumptions are not required for correctness.

Transaction format. A transaction is a pair $tx = (H_{\text{pub}}, \mathsf{payload})$, where $\mathsf{payload} \in \{0,1\}^{S_{\text{pl}}}$ is the payload and H_{pub} is a public header carrying the fee/accounting metadata together with a commitment to the payload. The sender samples commitment randomness σ and computes a commitment

$$C = \mathsf{Com}(\sigma, \mathsf{payload}).$$

Let $M := (\sigma \parallel \mathsf{payload})$ be the message dispersed by the code, of total length

$$S := |\sigma| + S_{\mathrm{pl}}.$$

The value σ is included as a prefix of the data to be encoded, i.e., the encoder operates on $(\sigma \parallel \mathsf{payload})$. We fold $|\sigma|$ into S for notational simplicity. This ensures σ becomes available upon successful decoding, enabling validators to verify the commitment opening. For the remainder of the paper we abuse notation slightly and refer to S as the "payload size", since $|\sigma|$ is fixed and negligible compared to S_{pl} in our parameter regimes. We write

$$H_{\mathrm{pub}} = (H_{\mathrm{pre}}, C, \Sigma), \qquad \mathsf{txID} = H(H_{\mathrm{pre}} \parallel C), \qquad \Sigma = \mathrm{Sig}_{\mathsf{sk}}(\mathsf{txID}).$$

We call H_{pre} the *preimage header*: a deterministically serialized tuple of fee/accounting fields required for mempool admission and bandwidth pricing.

MDS coding. Our Sedna protocol comes in three different variants depending on the technique used for replication: naive, MDS and rateless. As a baseline fixed-rate scheme, we consider (m, k) Maximum Distance Separable (MDS) codes. The sender encodes the payload S into m shares, each approximately S/k in size. Successful reconstruction of the original payload requires the recipient to gather any k distinct shares.

Rateless, verifiable coding. The main variant of our protocol employs a *rateless, verifiable encoder* R. On input the message $M := (\sigma \parallel \mathsf{payload})$, R produces an unbounded sequence of coded symbols

$$(y_1, y_2, \ldots), \qquad y_j \in \{0,1\}^{\ell_{\mathrm{sym}}}.$$

For clarity, we write the j-th symbol generator induced by R as R_j. Let $M \in \{0,1\}^S$ denote the (fixed) input message. Each R_j is a deterministic function

$$R_j : \{0,1\}^* \to \{0,1\}^{\ell_{\mathrm{sym}}},$$

such that $y_j = R_j(M)$, i.e., y_j is the j-th coded symbol produced by R (with any required per-symbol coefficient information derived deterministically from j and public parameters).

For a fixed transaction identifier txID, any verified pair (j, y_j) that appears in a bundle with $\mathsf{Ver}(\mathsf{txID}, i, J_i, \{y_j\}_{j \in J_i}, \Sigma_i) = 1$ is cryptographically bound to that transaction via the sender's signature.

We additionally assume that for any fixed $\mathsf{payload}$, the collection of symbols $\{y_j\}$ behaves as (pseudo)random linear combinations of the payload blocks, so that any set of r symbols reveals at most $r\,\ell_{\mathrm{sym}}$ bits of information about $\mathsf{payload}$. This property underpins our privacy analysis in Section 5.

From any K verified distinct symbols (for indices j of the verifier's choice), Dec reconstructs $M = (\sigma, \mathsf{payload})$ with probability at least $1 - \delta_{\mathrm{code}}$. We parameterize

$$K \stackrel{\mathrm{def}}{=} \left\lceil (1+\varepsilon)\,\frac{S}{\ell_{\mathrm{sym}}} \right\rceil,$$

where $\varepsilon > 0$ is a small overhead (e.g., 5%) and δ_{code} the residual decoding failure probability under the chosen rateless code family (e.g., LT/Raptor/RaptorQ).

Sender model. Senders can customize their desired trade-off between latency, cost, and censorship-resistance for each transaction. Sender $s \in \mathcal{S}$ has a specific censorship-resistance requirement for each transaction tx_s, expressed in the number of censoring validators $c(tx_s)$ the inclusion of transaction tx_s must tolerate. In particular, it is clear that at least $c(tx_s)+1$ validators must be reached, $0 \leq c(tx_s) \leq n-1$, to deterministically tolerate censoring from up to $c(tx_s)$ validators. We abuse notation by referring to c in the remainder of this document. Intuitively, for the naive approach of re-submitting the full transaction payload and metadata to c different validators, $c+1$ becomes the replication factor of the transaction. Thus, while f is a system parameter of Byzantine fault tolerance, c is a user parameter of censorship resistance.

Effective censorship. Given f, c we define $c_e(f, c)$ as the effective number of censored lanes. We abuse notation by referring to c_e. Typically, for $c \leq f$ then the adversary is not strong enough to selectively censor particular lanes, meaning that it can only censor the transaction in the lanes that it controls directly, hence $c_e = c$. For $c > f$, the adversary can additionally prevent progress on correct lanes, as finalization requires $\lfloor (n+f)/2 \rfloor + 1 = 2f+1$ votes (in the $n = 3f+1$ model), but since it is however possible for all protocols to enforce at least $n-f$ lanes must be decided in any iteration of MCP, then $c_e = n - (2f+1) + c$. Note that in this case $c_e = n$ for $c > 2f$ and thus the strongest adversary tolerable is $c = 2f$. Nonetheless, as the concrete power of the adversary to censor lanes outside of its direct control is dependent on the particular instance of the consensus implementation, we simply refer to c_e in the remainder of this document.

4 Sedna Protocol

We now present the Sedna protocol. At a high level, the sender (1) commits to the transaction payload and derives a unique identifier txID, (2) generates an unbounded stream of small, verifiable coded symbols, (3) packages disjoint subsets of symbols into bundles addressed to a sampled set of validator lanes, and (4) privately delivers these bundles. Validators verify incoming bundles by checking hash consistency, the header signature, the local accounting predicate, and the bundle signature. They do *not* verify that symbols are correctly encoded with respect to the commitment; malformed or inconsistent symbols are handled economically (fees are charged) and filtered by post-decode transaction validity checks.

Commitment and addressed shares. Using the commitment randomness σ and commitment $C = \mathsf{Com}(\sigma, \mathsf{payload})$ from Section 3, the sender obtains the

identifier $\mathsf{txID} = H(H_{\text{pre}} \parallel C)$ and header $H_{\text{pub}} = (H_{\text{pre}}, C, \Sigma)$. For a chosen set of symbol indices j, the sender forms addressed shares

$$\mathsf{Share}_{i,j} = (\mathsf{txID}, i, j, y_j, H_{\text{pub}}).$$

Validators maintain coded mempools of addressed bundles that passed local checks and bytes-fee payment. A block B_i^t is a sequence of such addressed bundles assembled by validator i.

Sedna may prescribe more than one share to be sent to the same validator. As a result, we aggregate a set of shares with the same validator v as recipient $\{\mathsf{Share}_{i,j}\}_{i=v}$ into a bundle:

$$Bundle_i = \big(\mathsf{txID},\ i,\ J_i,\ \{y_j\}_{j\in J_i},\ \Sigma_i,\ H_{\text{pub}}\big),$$

where J_i is the set of symbol indices carried in the bundle and

$$\Sigma_i = \text{Sig}_{\mathsf{sk}}\big(\mathsf{txID} \parallel i \parallel \langle j, y_j\rangle_{j\in J_i}\big)$$

is the sender's signature binding all symbols y_j to their indices j and lane i for transaction txID, where $\langle j, y_j\rangle_{j\in J_i}$ denotes the list of index–symbol pairs sorted by increasing j.

For a bundle addressed to lane i with index set J_i and symbols $\{y_j\}_{j\in J_i}$, we write

$$\mathsf{Ver}\big(\mathsf{txID}, i, J_i, \{y_j\}_{j\in J_i}, \Sigma_i\big) = 1$$

to mean that the bundle signature verifies, i.e.,

$$\text{Verify}_{\mathsf{pk}}\big(\mathsf{txID} \parallel i \parallel \langle j, y_j\rangle_{j\in J_i}, \Sigma_i\big) = 1.$$

For an honest sender and any external adversary that does not know sk, EUF-CMA security implies that any bundle with $\mathsf{Ver} = 1$ must have been explicitly produced by that sender; an external adversary cannot alter indices or symbol values without invalidating the signature.

Verifying shares/bundles. Upon receiving a bundle addressed to lane i with index set J_i and symbols $\{y_j\}_{j\in J_i}$, a validator performs the following checks:

1. *Hash consistency:* recompute $\mathsf{txID}' := H(H_{\text{pre}} \parallel C)$ from the header and check that $\mathsf{txID}' = \mathsf{txID}$.
2. *Header signature:* $\text{Verify}_{\mathsf{pk}}(\mathsf{txID}, \Sigma) = 1$.
3. *Accounting/fees:* the admission and bandwidth-pricing predicate on H_{pre} is satisfied.
4. *Bundle signature:* $\text{Verify}_{\mathsf{pk}}(\mathsf{txID} \parallel i \parallel \langle j, y_j\rangle_{j\in J_i}, \Sigma_i) = 1$, i.e., $\mathsf{Ver}(\mathsf{txID}, i, J_i, \{y_j\}_{j\in J_i}, \Sigma_i) = 1$ as in Definition 1.

A bundle is admitted to the mempool only if all four checks pass.

Cross-bundle equivocation. Bundle-level signatures bind symbols within a single bundle but do not cryptographically prevent a malicious sender from signing different values for the same index j in bundles addressed to different lanes. However, this equivocation is harmless: per-index deduplication (Section 5) ensures that only the first verified symbol for each (txID, j) pair in finalized history is considered, so all honest validators agree on a unique value for each index. Moreover, if a sender equivocates, the resulting symbol set is very unlikely to correspond to a meaningful payload: decoding will typically either fail or yield an invalid transaction, which is discarded while bandwidth fees remain charged. In either case, equivocation is economically irrational.

Deduplication order. The finalized ledger at height h consists of an ordered sequence of block vectors $(B_1^t, \ldots, B_n^t)$ for $t \leq h$. We scan this sequence in increasing height t, then by lane index $i \in \{1, \ldots, n\}$, then by intra-block position. The first occurrence of any (txID, j) in this order determines the unique symbol value y_j used for decoding; subsequent occurrences are ignored.

Lane sampling. For each transaction, the sender samples a subset $U \subseteq \mathcal{V}$ of size $0 < m \leq n$ and privately delivers one addressed bundle to each lane in U. For an honest sender, the index sets J_i across lanes are chosen to be disjoint, so that each published bundle contributes s previously unseen indices.

Inclusion, decoding, and execution order. For a fixed identifier txID, let $X_{\mathsf{txID}}(h)$ denote the set of deduplicated verified index–symbol pairs (j, y_j) for txID that appear in finalized history up to height h. Decoding is attempted whenever $|X_{\mathsf{txID}}(h)| \geq K$. Upon successful decoding, validators recover $(\sigma, \mathsf{payload})$ and verify the commitment opening:

$$C \stackrel{?}{=} \mathsf{Com}(\sigma, \mathsf{payload}).$$

If verification succeeds, the transaction is included. We define the *inclusion height*, denoted $\mathrm{ht}_{\mathrm{incl}}(\mathsf{txID})$, as the minimal block height h at which both decoding succeeds and the commitment opening verifies. The final, deterministic execution order is then established by sorting all included transactions lexicographically based on the tuple $(\mathrm{ht}_{\mathrm{incl}}(\mathsf{txID}), \mathsf{txID})$.

If a decoding attempt fails at some h (an event of probability at most δ_{code} for an honestly encoded transaction once $|X_{\mathsf{txID}}(h)| \geq K$), the protocol waits for additional verified symbols and retries at the next height where new symbols are available. A transaction that fails post-decode verification (e.g., commitment opening fails, or application-level validity checks fail) is discarded; bandwidth fees remain charged.

5 Sedna Guarantees

We now establish the main correctness, liveness, and privacy properties of Sedna. Unless stated otherwise, we work post-GST under the network and consensus

assumptions of Section 3, and we consider the rateless variant defined in Section 4. Throughout this section we assume that the commitment scheme Com is computationally binding and hiding, that each symbol generator R_j is deterministic, that the hash function H is collision resistant, and that the signature scheme is EUF-CMA secure.

We first formalize the verification relation for bundles and the decoder requirements.

Definition 1 (Verification relation)**.** *For a bundle addressed to lane i with index set J_i and symbols $\{y_j\}_{j\in J_i}$, we write $\mathsf{Ver}(\mathsf{txID}, i, J_i, \{y_j\}_{j\in J_i}, \Sigma_i) = 1$ iff the bundle signature verifies:*

$$\mathrm{Verify}_{\mathsf{pk}}\big(\mathsf{txID} \parallel i \parallel \langle j, y_j\rangle_{j\in J_i}, \Sigma_i\big) = 1,$$

where $\langle j, y_j\rangle_{j\in J_i}$ denotes index–symbol pairs sorted by increasing j. By EUF-CMA security, any bundle with $\mathsf{Ver} = 1$ was produced by the holder of sk.

Definition 2 (Inclusion Height)**.** *For a fixed transaction identifier txID and ledger height h, let*

$$X_{\mathsf{txID}}(h) = \Big\{(j, y_j) \;\Big|\; \begin{array}{l}\exists \text{ a finalized bundle } B \text{ with } \mathrm{ht}(B) \le h \\ \text{whose deduplicated contribution for } (\mathsf{txID}, j) \text{ is } y_j\end{array}\Big\}.$$

Decoding is attempted whenever $|X_{\mathsf{txID}}(h)| \ge K$. The inclusion height of txID, denoted $\mathrm{ht}_{\mathrm{incl}}(\mathsf{txID})$, is

$$\mathrm{ht}_{\mathrm{incl}}(\mathsf{txID}) \;=\; \min\Big\{\, h \;\Big|\; \begin{array}{l}|X_{\mathsf{txID}}(h)| \ge K,\ \mathsf{Dec}(X_{\mathsf{txID}}(h)) = (\sigma, \mathsf{payload}), \\ \text{and } C = \mathsf{Com}(\sigma, \mathsf{payload})\end{array}\Big\}.$$

If decoding fails or commitment verification fails at height h, the system waits for additional verified symbols and retries at the next height where new symbols are available.

Definition 3 (Monotone decoding for honest encodings)**.** *The decoder Dec is* monotone for honestly encoded transactions *if, for any two index sets J_1, J_2 with $|J_1|, |J_2| \ge K$, when the corresponding symbols are generated by the same honest sender as $y_j = R_j(M)$, either Dec outputs the same M (containing payload) on both sets, or Dec fails on at least one. Any residual disagreement probability is absorbed into δ_{code}.*

We work under the validity and deduplication rules of Section 4. Recall that deduplication affects symbol selection, not fee accounting: duplicate bundles that reach finalized history still incur bandwidth fees. Note that verification does not enforce correct encoding; a malicious sender may sign arbitrary symbols. Garbage payloads are deterred economically, as fees are charged regardless of whether decoding succeeds.

5.1 Safety and Correctness

We establish the core safety properties of Sedna: that for any finalized ledger the set of symbols used for each index is deterministically resolved, that headers and commitments cannot be malleated, and that decoding yields a unique payload. Together these ensure that the execution order $(\mathrm{ht}_{\mathrm{incl}}(\mathsf{txID}), \mathsf{txID})$ is a deterministic function of the finalized ledger.

Lemma 1 (Validity and Deterministic Resolution)**.** *If a finalized block contains a bundle with* $\mathsf{Ver} = 0$*, the block is invalid and must be rejected. Moreover, while a malicious sender may produce conflicting verified bundles for the same* txID*, the protocol's deduplication rule ensures that for any fixed finalized ledger, the set of accepted symbols is uniquely determined.*

Proof. Consensus validity requires $\mathsf{Ver} = 1$ for all included bundles. Conflicting symbols for the same (txID, j) are resolved by per-index deduplication: only the first occurrence in finalized order is used. □

Lemma 2 (Header and Commitment Non-Malleability)**.** *Let* $\mathsf{txID} = H(H_{\mathrm{pre}} \parallel C)$ *and* $\Sigma = \mathsf{Sig}_{\mathsf{sk}}(\mathsf{txID})$*. Assume* H *is collision resistant and the signature scheme is EUF-CMA secure. Then, given a valid pair* (H_{pre}, C) *and* (txID, Σ)*, no PPT adversary can produce* $(H'_{\mathrm{pre}}, C') \neq (H_{\mathrm{pre}}, C)$ *with the same identifier* txID *and a valid signature under the sender's key, except with negligible probability.*

Proof. If $(H'_{\mathrm{pre}}, C') \neq (H_{\mathrm{pre}}, C)$ and $H(H'_{\mathrm{pre}} \parallel C') = H(H_{\mathrm{pre}} \parallel C)$, then H has a collision. If the adversary instead changes H'_{pre} but reuses Σ on the same txID, it must either break the binding between (H_{pre}, C) and txID (again causing a hash collision) or forge a signature under the sender's key, violating EUF-CMA security. □

Lemma 3 (Unique decode for honest senders)**.** *Assume the sender is honest and encodes as* $y_j = R_j(M)$ *for all* j*, where* $M = (\sigma \parallel \mathsf{payload})$*. Assume* Dec *outputs* M *from any set of at least* K *correctly encoded index-symbol pairs with probability at least* $1 - \delta_{\mathrm{code}}$*. Then any two sets* X_1, X_2 *of correctly encoded index–symbol pairs with* $|X_1|, |X_2| \geq K$ *decode to the same payload, except with probability at most* δ_{code}*.*

Proof. Each $y_j = R_j(M)$ is unique by determinism. By Lemma 1, deduplication fixes a unique symbol per index. Thus any two sets of K verified symbols are subsets of the same codeword, decoding to the same payload except with probability δ_{code}. □

For a malicious sender who signs inconsistent or garbage symbols, decoding may fail or produce an invalid transaction. In this case, the transaction is discarded but bandwidth fees (prepaid in H_{pub}) remain charged. This fee mechanism deters garbage submissions without requiring proofs of correct encoding.

For a fixed txID and height h, we write $X_{\mathsf{txID}}(h)$ for the set of distinct verified index–symbol pairs as in Definition 2. Per-index deduplication ensures that duplicates do not change $X_{\mathsf{txID}}(h)$.

Theorem 1 (Order Determinism)**.** *Let* $\mathrm{ht}_{\mathrm{incl}}(\mathsf{txID})$ *be the inclusion height of* txID *as in Definition 2. Then the execution order obtained by sorting all included transactions lexicographically by* $(\mathrm{ht}_{\mathrm{incl}}(\mathsf{txID}), \mathsf{txID})$ *is a deterministic function of the finalized ledger.*

Proof. Finality fixes the ledger prefix up to height h, hence fixes $X_{\mathsf{txID}}(h)$ for every txID and h (Lemma 1). Since Dec is deterministic, its success/failure and output on input $X_{\mathsf{txID}}(h)$ are fixed (Lemma 3). Therefore $\mathrm{ht}_{\mathrm{incl}}(\mathsf{txID})$ is fixed by the finalized ledger, and so is the lexicographic order on $(\mathrm{ht}_{\mathrm{incl}}(\mathsf{txID}), \mathsf{txID})$. □

Theorem 2 (Non-Malleability of Sedna)**.** *Fix a finalized ledger. Under the assumptions above, for each identifier* txID *corresponding to an honestly encoded transaction, the decoded payload and its execution position* $(\mathrm{ht}_{\mathrm{incl}}(\mathsf{txID}), \mathsf{txID})$ *are uniquely determined by that ledger, except with probability at most* δ_{code} *due to decoder failure.*

Proof. By definition, $X_{\mathsf{txID}}(h)$ depends only on the finalized prefix up to h. By Lemma 1, symbols are deterministically resolved; by Lemma 3, decoding is unique; by Theorem 1, execution position is fixed. The only failure mode is decoding failure (probability at most δ_{code}). □

Theorem 2 constrains semantics for a fixed finalized history; it does not constrain an adversary's influence on $\mathrm{ht}_{\mathrm{incl}}(\mathsf{txID})$ via censorship or scheduling. We prove censorship resistance and liveness in Section 5.2.

5.2 Liveness and Censorship Resistance

We now quantify the per-slot success probability of inclusion under the good case regime. Let m denote the number of addressed lanes for a transaction, let s denote the number of symbols per bundle, and let c_e be the effective number of censored lanes as defined in Section 3. Let H be the random variable counting how many of the m addressed lanes are honest. Since we are sampling without replacement from the n lanes, we have

$$H \sim \mathrm{Hypergeom}(n, n - c_e, m).$$

Lemma 4 (Single-slot Inclusion (good case))**.** *Assume in a slot all addressed honest lanes publish their bundles. Then inclusion succeeds in that slot with probability at least*

$$(1 - \delta_{\mathrm{code}}) \cdot \Pr\big[\, H \geq \lceil K/s \rceil \,\big].$$

Proof. Condition on the event $\{H \geq \lceil K/s \rceil\}$. Each honest lane contributes s verified symbols, so at least K distinct verified indices for txID appear in finalized history. By the decoder guarantee, decoding succeeds with probability at least $1 - \delta_{\mathrm{code}}$ given any K verified symbols. Taking expectations over the distribution of H yields the stated lower bound. No independence assumption beyond the hypergeometric sampling is required. □

Lemma 5 (Multi-slot inclusion under resampling)**.** *If the sender resamples U each slot from the same distribution (parameters m, s, K) and, post-GST, the per-slot success probability satisfies*

$$p \;\geq\; (1-\delta_{\text{code}}) \cdot \Pr\big[H \geq \lceil K/s \rceil\big],$$

then the number of slots to inclusion is stochastically dominated by a geometric r.v. with parameter p.

Proof. Each slot is a Bernoulli trial that succeeds whenever enough honest addressed lanes are hit and Dec succeeds. The same conditional argument as in Lemma 4 yields the per-slot success probability lower bound, geometric domination follows. □

We show in Section 6.2, Theorem 5 the optimal submission strategy for a total per-slot failure probability δ, comparing it with variants of naive replication (Theorem 3) and MDS coding (Theorem 4). We next turn to privacy guarantees.

5.3 Privacy

Sedna aims for until decode privacy before K verified symbols for txID are published, the adversary should learn essentially only the bits contained in the revealed symbols and the public header H_{pub}. Let the adversary's pre-decode view of a transaction be captured by the leakage function

$$\mathcal{L}(tx) := \Big(H_{\text{pub}},\ \big\{(i,j,y_j) \text{ for each verified symbol observed}\big\}\Big),$$

where the set includes lane indices i, symbol indices j, and symbol values y_j. Bundle signatures are also visible to the adversary but depend only on txID and the observed (j, y_j) values.

Lemma 6 (Privacy until K symbols)**.** *For any $r < K$ observed verified symbols,*

$$H_\infty(\mathsf{payload} \mid \mathcal{L}(tx) \text{ with } r \text{ symbols}) \geq H_\infty(\mathsf{payload}) - r\,\ell_{\text{sym}}.$$

Indices and lane labels may leak structure (e.g., sampler bias) but no additional payload bits beyond $r\,\ell_{\text{sym}}$.

Proof. The adversary's view consists of the public header H_{pub} (including the commitment C), and the observed symbols $\{(j, y_j)\}$ for r indices, along with the corresponding bundle signatures. By the information-theoretic properties of rateless codes, each symbol y_j is a (pseudo)random linear combination of input blocks, contributing at most ℓ_{sym} bits of information about the payload. By the hiding property of Com, the commitment C leaks at most negligible information about payload. The bundle signatures Σ_i depend only on the transaction ID, the indices, the symbol values, and the sender's key; they do not reveal unobserved portions of the payload. Thus the adversary's information about the payload is bounded by the $r \cdot \ell_{\text{sym}}$ bits contained in observed symbols. □

We next bound the adversary's probability of early decode reconstructing the payload before any honest user could reasonably expect inclusion. Let A be the number of addressed lanes that are controlled by the adversary (so $A \sim \text{Hypergeom}(n, c_e, m)$). If each adversarial lane receives a bundle with s symbols, then the adversary obtains $A \cdot s$ symbols in that slot.

Lemma 7 (Adversarial Early Decode Probability)**.** *Let $A \sim \text{Hypergeom}(n, c_e, m)$ be the number of adversarially controlled addressed lanes. If each bundle carries s symbols, the probability that the adversary can attempt reconstruction in a slot is*

$$\Pr\big[A \geq \lceil K/s \rceil\big].$$

If $ms < K$, this probability is zero.

Proof. The adversary's symbol budget is $A \cdot s$. Early decode requires $A \cdot s \geq K$, i.e. $A \geq \lceil K/s \rceil$, which yields the stated probability. If $m\,s < K$ then even if $A = m$ (full control of all addressed lanes) the adversary holds fewer than K symbols, so early decode is impossible. □

Lemmas 6 and 7 quantify the trade off between the number of addressed lanes m, symbols per bundle s, and the decoder threshold K, increasing m and s improves liveness but also increases the adversary's early decode probability. Section 6 compares these trade offs for variants of Sedna.

6 Performance Comparison

In this section, we compare the performance of the Sedna protocol (using verifiable rateless coding) against two variants: Π_{str} that uses naive replication, and Π_{mds} that uses fixed-rate MDS coding. The primary goal is to understand the trade-offs between these strategies, particularly concerning bandwidth consumption, under varying parameters such as payload size, metadata overhead, and the sender's desired censorship tolerance (c_e) and reliability (δ). We refer to the variant of Sedna shown in the previous section as Π_{rtl}.

We first define the cost metrics for each approach in terms of total published bytes and the minimum bytes required for successful inclusion. We then analyze the asymptotic overhead for large payloads to understand fundamental scalability limits. While naive replication is simple, it often incurs significant overhead and sacrifices pre-finality privacy. MDS coding offers better bandwidth efficiency for large payloads but requires careful parameter selection (m, k) upfront and typically has zero decoding failure probability ($\delta_{code} = 0$). Rateless encoding aims to provide flexibility and potentially lower overhead, especially when accounting for metadata and probabilistic guarantees. We analyse these differences, paving the way for evaluating concrete end-to-end latency and other empirical metrics in Section 7.

6.1 Bandwidth Cost

Let M_h denote the per-bundle header overhead in bytes (the components of $Bundle_i$ excluding symbol data) and M_s the per-symbol metadata (the index j). Each of the m lanes receives a bundle containing s symbols in the rateless case, one share per lane in the MDS case (no need then to use M_s), or the full transaction in the naive case.

We analyze the bandwidth cost for each submission strategy, defining both the total published bytes (if all m addressed lanes publish their data) and the minimum required bytes for a successful decode in a single slot in Proposition 1.

Proposition 1 (Bandwidth Cost)**.** *Let m be the number of addressed lanes. Write L_{pub} for total published bytes and $L_{\min}$ for the minimum bytes required for inclusion.*

- ***(a) Naive:*** $L_{\text{pub}} = m(\mathsf{M}_h + S), \quad L_{\min} = \mathsf{M}_h + S.$
- ***(b) MDS*** (m, k)***:*** $L_{\text{pub}} = m(\mathsf{M}_h + S/k), \quad L_{\min} = k(\mathsf{M}_h + S/k).$
- ***(c) Rateless:*** $L_{\text{pub}} = m(\mathsf{M}_h + s(\mathsf{M}_s + \ell_{\text{sym}})), \quad L_{\min} = \lceil K/s \rceil (\mathsf{M}_h + s(\mathsf{M}_s + \ell_{\text{sym}}))$.

Proof. Each lane publishes one bundle. For naive replication, inclusion requires 1 honest lane; for MDS, k lanes; for rateless, $\lceil K/s \rceil$ lanes (each contributing s symbols toward the K needed). Multiplying per-bundle size by the relevant lane count gives both expressions. □

Proposition 1 provides exact costs including metadata, understanding the fundamental scalability requires examining the asymptotic behavior for large payloads, where metadata costs become negligible relative to the payload data itself. Corollary 1 derives the asymptotic overhead floor, defined as the ratio of total published bytes to the original payload size (L_{pub}/S), for each strategy.

Corollary 1 (Comparison of Asymptotic Overhead Floors)**.** *Ignoring metadata costs (assuming S is large) and analyzing the probabilistic case, the asymptotic byte overhead factor (L_{pub}/S) for each strategy reveals its fundamental scalability:*

- ***(a) Naive Replication:*** *The overhead floor is m_{opt}, the smallest m satisfying the probabilistic constraint* $\Pr[H < 1] \leq \delta$.
- ***(b) MDS Coding:*** *The overhead floor is determined by the maximum reliable code rate, which is the uncensored lane ratio. The floor is therefore* $\frac{\mathbf{1}}{\mathbf{1}-\mathbf{c_e}/\mathbf{n}}$. *This overhead scales with the* ***ratio*** *of censors.*
- ***(c) Rateless Coding:*** *The overhead floor is* $\frac{\mathbf{1}+\varepsilon}{\mathbf{1}-\mathbf{c_e}/\mathbf{n}}$. *This matches the scalability of MDS coding, with an explicit overhead factor of* $(1+\varepsilon)$ *paid for the flexibility of the rateless scheme.*

Proof. Immediate from Proposition 1 by taking $S \to \infty$. Metadata terms vanish, leaving overhead determined by the replication/coding rate. For MDS and rateless, the effective rate is $(1 - c_e/n)$; rateless pays an additional $(1+\varepsilon)$ factor for decode threshold K. □

While Corollary 1 shows coding's asymptotic superiority for large payloads ($S \to \infty$), metadata overhead significantly impacts performance for practical transaction sizes and can make naive replication competitive:

- **For Naive Replication and MDS Coding,** the per-symbol metadata within a lane M_s does not apply, i.e. $M = \mathsf{M}_h$. Its total impact on the overhead ratio is proportional to $m \cdot \mathsf{M}_h/S$, which diminishes directly as S grows. The additional indexing in the MDS case is negligible extra metadata compared to the rest of metadata M_h required in both the naive and MDS case. However, naive replication, because of requiring only one successful lane, could lead to less lanes containing any copy/share, which could pay off for transactions with a payload of similar size to M_h (i.e. $S \approx 100s$ of bytes). For medium-to-large payloads, MDS coding provides a better duplication factor.

- **For Rateless Coding,** the overhead is approximately $\left(1 + \frac{\mathsf{M}_h + s\mathsf{M}_s}{s\ell_{\text{sym}}}\right)$ times the asymptotic floor from Corollary 1. Using smaller symbols (decreasing ℓ_{sym}) makes the encoding more granular but increases the relative cost of per-symbol metadata, M_s. For finite $\mathsf{M} = \mathsf{M}_h + s\mathsf{M}_s$, one has $L_{\text{pub}_{\text{rtl}}}(\mathsf{txID})/S \approx \frac{1+\varepsilon}{1-c_e/n}\big(1 + \mathsf{M}/\ell_{\text{sym}}\big)$. Subject to block/mempool limits, taking $\ell_{\text{sym}} \gg \mathsf{M}$ drives the metadata term down; the intrinsic floor $\frac{1+\varepsilon}{1-c_e/n}$ remains. Thus, in practice, rateless coding approaches the bandwidth efficiency of MDS for payloads in the order of several KBs, where the extra metadata is amortized, while providing better end-to-end latency (Figure 7).

Therefore, the optimal strategy for transactions can be heavily influenced by metadata costs. The next sections calculate the best strategy for reduced bandwidth depending on the approach, $\mathsf{M}_h, \mathsf{M}_s,$ and S, along with the failure budget and censorship-tolerance. We evaluate concrete implementations of each variant in Section 7.

6.2 Optimal Submission Strategies

A sender's goal is to select the optimal submission strategy that minimizes total bandwidth cost while meeting their desired probabilistic liveness guarantee (i.e., a failure probability of at most δ). The optimal configuration for each strategy is defined by the solution to the following optimization problems. Theorem 3 shows that, for naive replication, the only parameter to optimize is the number of lanes m, aiming for at least one successful transmission.

Theorem 3 (Optimal Configuration for Naive Replication)**.** *To minimize bandwidth using naive replication, a sender must find the optimal number of lanes m_{opt}. This is the smallest integer m that satisfies the probabilistic constraint:*

$$\Pr[H < 1] \leq \delta, \quad \textit{where } H \sim \text{Hypergeom}(n, n - c_e, m).$$

A conservative closed-form approximation for m_{opt} is given by:

$$\boxed{m_{opt} \geq \left(\frac{b + \sqrt{b^2 + 4c_r}}{2c_r} \right)^2}$$

where $c_r = 1 - c_e/n$ is the assumed honest validator ratio and $b = \sqrt{2c_r \ln(1/\delta)}$.

Proof. Cost $m(M_h + S)$ is strictly increasing in m, so the optimum is the smallest m satisfying the constraint. The closed-form follows from a Chernoff bound on the hypergeometric tail. □

MDS coding introduces the parameter k (number of shares needed for reconstruction). Optimizing involves finding the best pair (m, k) that minimizes cost while ensuring at least k honest lanes succeed. This typically requires a numerical search over possible values of k, determining the minimum required m for each k using the constraint, as we show in Theorem 4

Theorem 4 (Optimal Configuration for MDS Coding)**.** *To minimize total bandwidth using an (m, k) MDS code, a sender must find the optimal integer pair (m_{opt}, k_{opt}) that solves the optimization problem:*

$$\min_{m,k} \left[m \cdot \left(\mathsf{M}_h + \frac{S}{k} \right) \right]$$

For any choice of k, the number of lanes m must satisfy the probabilistic constraint:

$$\Pr[H < k] \leq \delta, \quad \textit{where } H \sim \text{Hypergeom}(n, n - c_e, m).$$

A conservative closed-form approximation for the required m given a choice of k is:

$$\boxed{m \geq \left(\frac{b + \sqrt{b^2 + 4c_r k}}{2c_r} \right)^2}$$

where $c_r = 1 - c_e/n$ and $b = \sqrt{2c_r \ln(1/\delta)}$.

Proof. Analogous to Theorem 3: for each k, find the smallest m satisfying the constraint, then minimize over k. □

Rateless coding offers more flexibility by optimizing the number of lanes m, symbols per bundle s, and total symbols needed K. The goal is to find the tuple (m, s, K) minimizing cost while ensuring at least $\lceil K/s \rceil$ honest lanes succeed. This involves a multi-variable search over s and K, determining the necessary m for each pair via the constraint. Note the constraint adjusts the target probability to account for the intrinsic decoding failure probability δ_{code} of the rateless scheme itself. We show this in Theorem 5.

Theorem 5 (Optimal Configuration for Rateless Coding)**.** *To minimize bandwidth using the rateless scheme, a sender must find the optimal tuple* $(m_{opt}, s_{opt}, \ell_{\mathrm{sym}\,opt})$ *that solves the optimization problem:*

$$\min_{m,s,\ell_{\mathrm{sym}}} \left[m \cdot (\mathsf{M}_h + s \cdot (\mathsf{M}_s + \ell_{\mathrm{sym}}))\right]$$

where the decode threshold is determined by the symbol size:

$$K(\ell_{\mathrm{sym}}) \;:=\; \left\lceil (1+\varepsilon)\,\frac{S}{\ell_{\mathrm{sym}}} \right\rceil.$$

For any choice of (s, ℓ_{sym})*, the number of lanes* m *must satisfy the probabilistic constraint:*

$$\Pr\big[H < \lceil K(\ell_{\mathrm{sym}})/s \rceil\big] \le \delta - \delta_{\mathrm{code}}, \quad \textit{where } H \sim \mathrm{Hypergeom}(n, n-c_e, m).$$

A conservative closed-form approximation for the required m *is given by:*

$$\boxed{m \ge \left(\frac{b + \sqrt{b^2 + 4c_r\,K'(\ell_{\mathrm{sym}})}}{2c_r}\right)^2}$$

where $K'(\ell_{\mathrm{sym}}) = \lceil K(\ell_{\mathrm{sym}})/s \rceil$ *is the required number of honest lanes,* $c_r = 1 - c_e/n$*, and* $b = \sqrt{2c_r \ln(1/(\delta - \delta_{\mathrm{code}}))}$*.*

Proof. The optimization is a multi-variable search over the symbols per bundle s and the symbol size ℓ_{sym}. The decode threshold $K(\ell_{\mathrm{sym}})$ is not an independent variable but is determined by the choice of ℓ_{sym} via the rateless code parameterization. For each candidate pair (s, ℓ_{sym}), the minimum required m is the smallest integer satisfying the probabilistic constraint, for which the provided closed-form bound serves as a conservative estimate. The optimal tuple is the one yielding the minimum total cost across this search space. □

These theorems provide the mathematical basis for determining the most bandwidth-efficient configuration for each submission strategy under given specific system's conditions (n) and user requirements (c_e, δ, S). Section 7 provides concrete comparisons in practice, depending on each of the relevant user parameters.

6.3 Comparison to the Deterministic Lower Bound

To contextualize the performance of these probabilistic strategies, we compare them to the fundamental limit of any deterministic coding scheme.

Theorem 6 (Information-theoretic lower bound, deterministic threshold)**.** *Assume decoding must succeed from any set of* $n - c_e$ *lanes and fail from any*

smaller set. Let lane i contribute a total length ℓ_i (possibly via multiple shares), and set $L := \sum_{i=1}^{n} \ell_i$. Then

$$L \;\geq\; \frac{n}{n - c_e}\, S,$$

so the overhead factor L/S is at least $1/(1 - c_e/n)$.

Proof. Let $k := n - c_e$. For every subset J of lanes of size k, the total contributed length satisfies $\sum_{j \in J} \ell_j \geq S$. Summing over all $\binom{n}{k}$ such subsets and noting that each ℓ_i appears exactly $\binom{n-1}{k-1}$ times yields $\binom{n-1}{k-1} L \geq \binom{n}{k} S$. Using $\binom{n}{k} = \frac{n}{k}\binom{n-1}{k-1}$ gives $L \geq \frac{n}{k} S = \frac{n}{n-c_e} S$. □

The information-theoretic lower bound derived in Theorem 6 for any deterministic scheme ($1/(1-c_e/n)$) matches the asymptotic overhead floor previously calculated for probabilistic MDS and (up to $1 + \epsilon$) Rateless coding strategies (Corollary 1). This demonstrates that these probabilistic approaches are asymptotically optimal in terms of bandwidth overhead.

7 Evaluation

We evaluate Sedna through a combination of analytical computation and microbenchmarks. Our evaluation addresses three primary questions: (1) How do the bandwidth overheads of naive replication, MDS coding, and rateless coding compare across different payload sizes and censorship assumptions? (2) What is the computational cost of encoding and decoding, and does it introduce prohibitive latency? (3) How do the three approaches differ in their privacy properties, specifically the adversary's probability of early payload reconstruction?

7.1 Experimental Setup

Unless otherwise specified, we use the following default parameters throughout our evaluation. We consider a validator set of $n = 256$ validators with an effective censorship ratio of $c_e/n = 0.125$. The target per-slot failure probability is $\delta = 10^{-9}$.

For the rateless scheme, we set the coding overhead parameter $\varepsilon = 0.05$, per-bundle header metadata $M_h = 200$ bytes (e.g., 32 bytes for the transaction identifier, 32 bytes for the commitment, 64 bytes for the header signature, 64 bytes for the bundle signature, and a small amount for lane index and other header fields), per-symbol metadata $M_s = 8$ bytes (symbol index), and symbol size $\ell_{\text{sym}} = 256$ bytes.

Individual figures may use different parameter values to illustrate specific phenomena; deviations from the defaults are noted in figure captions.

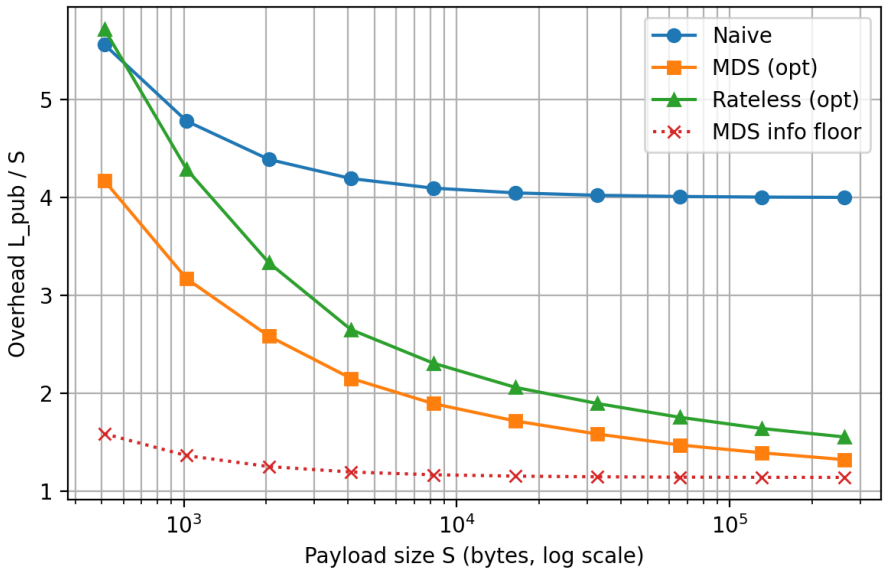

Figure 1: Bandwidth overhead factor as a function of payload size. The dotted line shows the information-theoretic lower bound $1/(1 - c_e/n)$ from Theorem 6, for $n = 256$, $c_e/n = 0.125$, $\delta = 10^{-9}$.

7.2 Bandwidth Overhead

We begin by examining the core bandwidth efficiency claims of Sedna. Figure 1 presents the overhead factor L_{pub}/S as a function of payload size for all three submission strategies, where L_{pub} denotes total published bytes and S is the original payload size.

Several observations emerge from this figure. First, naive replication maintains a nearly constant overhead across all payload sizes, reflecting the fact that the replication factor m is determined solely by the censorship tolerance requirement and is independent of S. Second, both MDS and rateless coding exhibit decreasing overhead as payload size increases, asymptotically approaching the information-theoretic floor of $1/(1 - c_e/n)$ established in Theorem 6. Third, for small payloads, metadata overhead causes all three approaches to converge, and naive replication can be competitive due to its lower per-bundle metadata requirements.

The rateless scheme incurs slightly higher overhead than MDS due to the $(1+\varepsilon)$ factor and additional per-symbol metadata M_s. However, this gap narrows for larger payloads as the metadata cost is amortized.

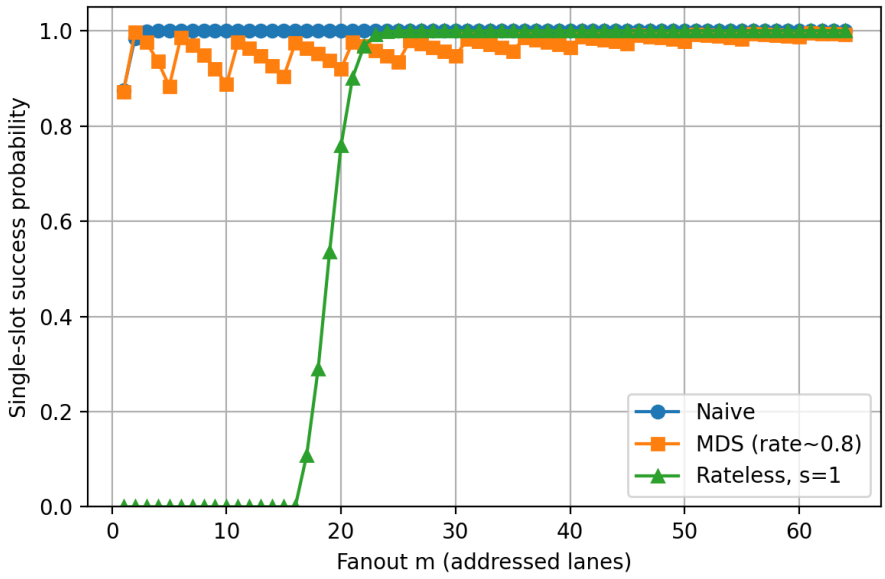

Figure 2: Single-slot success probability versus fanout m for naive replication, MDS coding, and rateless coding. Parameters: $n = 256$, $c_e/n = 0.125$, $\delta = 10^{-9}$, $S = 4096$ bytes.

7.3 Single-Slot Success Probability

The liveness guarantees of each approach depend critically on the relationship between the number of addressed lanes m and the probability of successful inclusion in a single slot. Figure 2 compares the single-slot success probability as a function of fanout m for all three strategies.

Naive replication achieves high success probability at low fanout because inclusion requires only a single honest lane to publish the transaction. In contrast, both coded approaches exhibit threshold behavior: success probability remains near zero until the fanout reaches a critical value, then rises sharply to near certainty. This threshold corresponds to the point at which the expected number of honest lanes exceeds the decoding requirement (k for MDS, $\lceil K/s \rceil$ for rateless).

Users can shift the rateless threshold by adjusting the symbols-per-lane parameter s, as illustrated in Figure 3.

Higher values of s shift the success curve leftward, enabling users to achieve target success probabilities with fewer contacted lanes. This flexibility allows users to navigate the tradeoff between the number of network connections (which affects dissemination latency) and bundle size (which affects per-lane bandwidth). The optimal choice depends on application-specific constraints such as network topology and validator bandwidth limits.

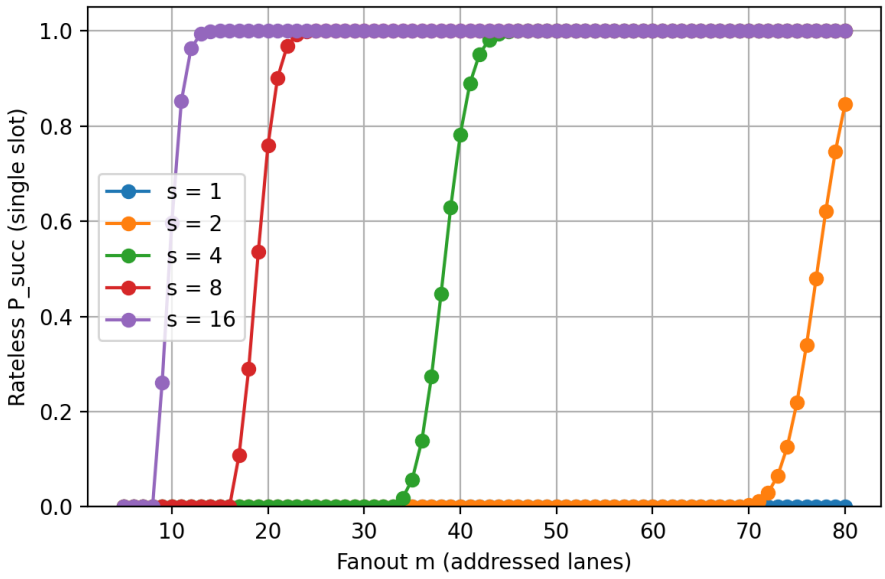

Figure 3: Effect of symbols per lane s on rateless success probability. Increasing s reduces the required number of honest lanes at the cost of larger bundle sizes. Parameters: $n = 256$, $c_e/n = 0.125$, $\delta = 10^{-9}$, $S = 32768$ bytes.

7.4 Impact of Censorship Assumptions

The bandwidth overhead of coded approaches depends fundamentally on the assumed censorship ratio c_e/n. Figure 4 illustrates how overhead scales with censorship tolerance for rateless coding.

As expected from Corollary 1, the asymptotic overhead floor increases with censorship tolerance according to $(1+\varepsilon)/(1-c_e/n)$. Importantly, even at high censorship ratios, coded approaches substantially outperform naive replication for large payloads, as shown in Figure 1.

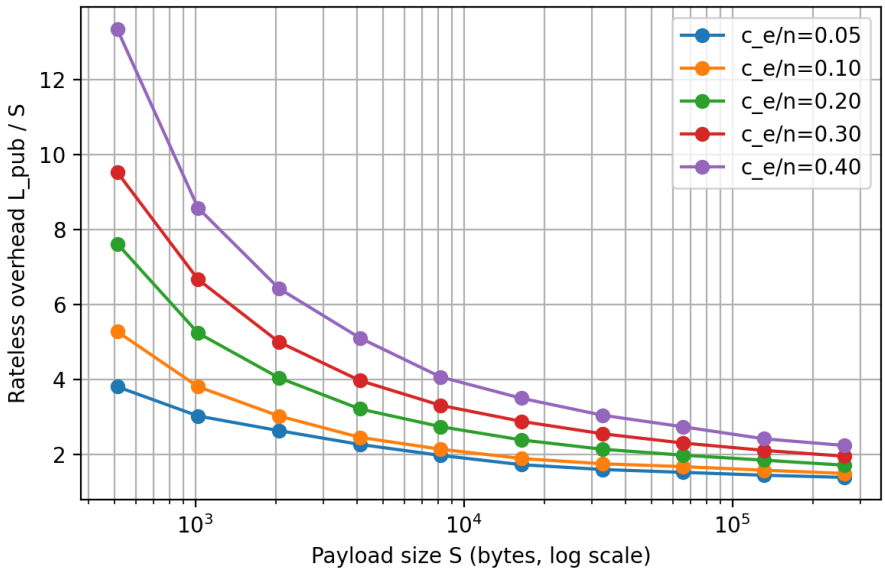

Figure 4: Rateless coding overhead as a function of payload size for varying censorship ratios $c_e/n \in \{0.05, 0.10, 0.20, 0.30, 0.40\}$. Higher censorship tolerance requires proportionally more redundancy, but overhead decreases with payload size in all cases. Parameters: $n = 256$, $\delta = 10^{-9}$.

7.5 Reliability-Cost Tradeoff

Users can select their desired failure probability δ based on application requirements. Figure 5 shows how bandwidth overhead varies with the failure budget for each submission strategy.

All approaches require increased resources as the failure budget decreases, but naive replication exhibits the steepest growth. This difference arises because naive replication must add entire copies of the payload to improve reliability, whereas coded approaches can add smaller increments of redundancy. For applications requiring high reliability, the bandwidth advantage of coding becomes more pronounced.

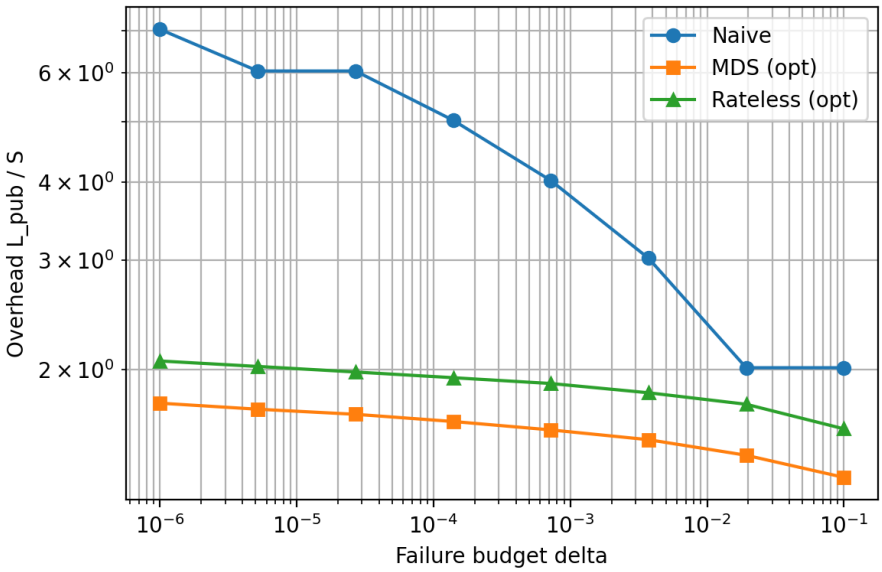

Figure 5: Bandwidth overhead as a function of failure budget δ. Lower failure probability (higher reliability) requires additional redundancy. Coded approaches scale more gracefully than naive replication as reliability requirements tighten. Parameters: $n = 256$, $c_e/n = 0.125$, $S = 4096$ bytes.

7.6 Privacy and Early Decode Probability

A key advantage of Sedna over naive replication is the reduction in pre-inclusion information leakage. With naive replication, any addressed lane that is adversarially controlled immediately learns the full transaction payload. With coded approaches, the adversary must collect at least k (MDS) or K (rateless) symbols before reconstruction is possible. Figure 6 quantifies the adversary's probability of early decode.

Under naive replication, the adversary's early decode probability is trivial, reflecting the probability that at least one of the m addressed lanes is adversarially controlled. In contrast, both MDS and rateless coding maintain early decode probability near zero across tested configurations, as shown in Figure 6. This occurs because the adversary would need to control at least k or $\lceil K/s \rceil$ of the addressed lanes, which is unlikely when parameters are chosen appropriately.

This privacy improvement directly translates to reduced MEV exposure. Under naive replication, a user's transaction is visible to the adversary with high probability before any honest proposer includes it, enabling front-running, sandwiching, and other extraction strategies. Coded approaches ensure that the payload remains hidden until honest validators collectively publish sufficient symbols for decoding, at which point inclusion is imminent.

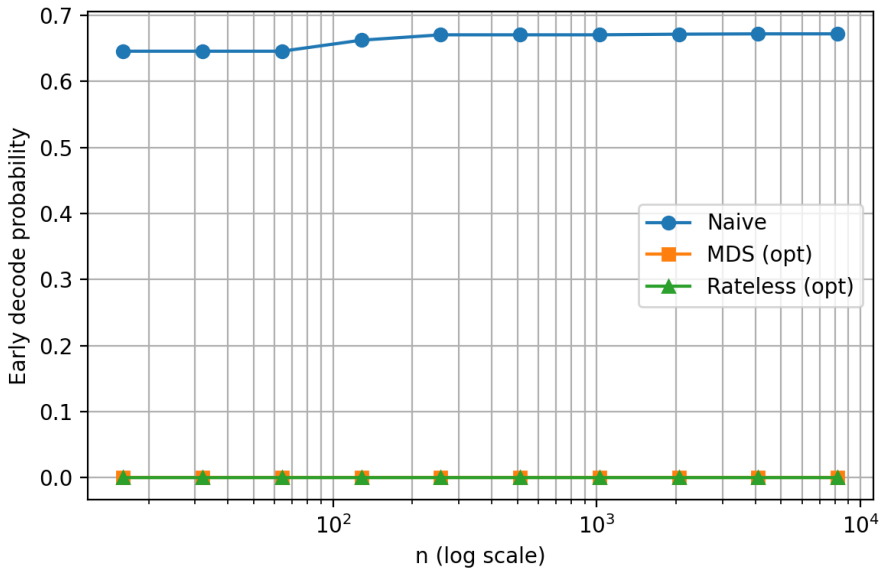

Figure 6: Early decode probability as a function of validator set size $n \in [16, 8192]$. Parameters: $c_e/n = 0.2$, $\delta = 10^{-9}$.

7.7 Computational Overhead

While the primary focus of Sedna is bandwidth efficiency, computational overhead must also be practical for real-world deployment. Figure 7 presents latency measurements for encoding and decoding operations.

Naive replication serves as a baseline with effectively zero coding overhead, requiring only signature generation and serialization. MDS and rateless coding introduce additional computational work for symbol generation, signing, and decoding. In our measurements, rateless remains consistently faster than MDS. The MDS implementation is SIMD-accelerated in our prototype, so constant-factor differences depend on engineering choices; nonetheless, we observe that MDS latency degrades more steeply (roughly doubling over the tested range), supporting our choice to recommend the rateless variant as the scalable version of Sedna.

Naive replication serves as a baseline with effectively zero encoding overhead, requiring only signature generation and serialization. Rateless coding introduces additional computational work for symbol and signature generation and verification. Interestingly, the rateless scheme shows lower end-to-end latency than naive replication in our measurements. This counterintuitive result arises because the bandwidth savings from coding dominate the additional computational overhead of symbol generation, bundle construction, and encoding. The reduced data volume leads to faster network transmission, more than com-

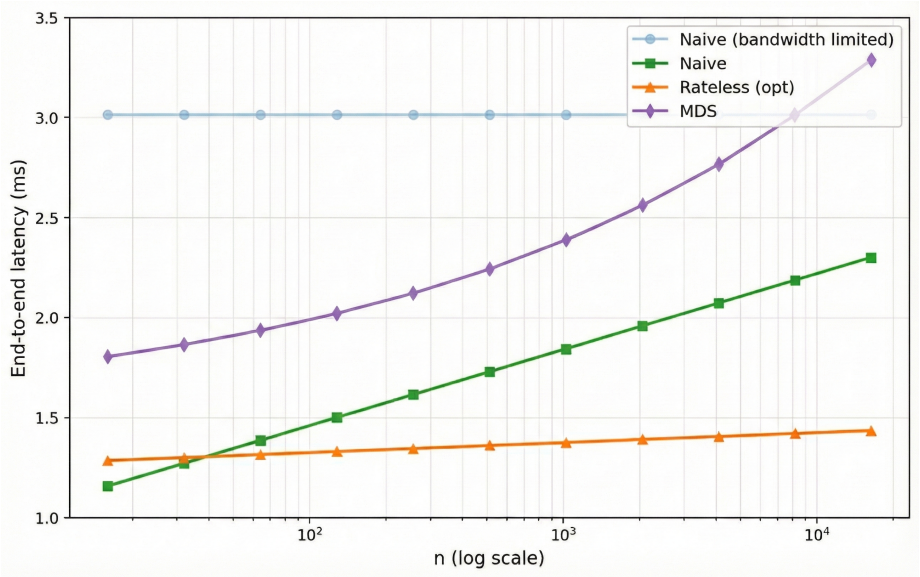

Figure 7: End-to-end latency as a function of validator set size $n \in [16, 8192]$ for naive replication (with and without bandwidth limits), MDS coding, and rateless coding. Rateless coding achieves lower latency than naive replication despite additional encoding overhead, as bandwidth savings from reduced data transmission dominate computational costs. Parameters: $c_e/n = 0.2$, $\delta = 10^{-9}$.

pensating for encoding costs. MDS coding, while providing better bandwidth utilization, degrades in latency with the number of proposers, despite MDS being SIMD-accelerated in our prototype, justifying our choice for rateless encoding as the recommended, scalable protocol version for Sedna.

7.8 Case Study: Real World Workloads

To ground our analysis in real-world systems, we consider transaction dissemination patterns observed in Filecoin, an MCP blockchain where messages are broadcast via gossip and proposers independently select from their mempools. Empirical observations indicate average replication factors of 3–4×, with some transactions replicated up to 9× due to uncoordinated submission and deduplication failures.

For systems that charge users for all duplicates, the savings from coded approaches accrue directly to users. Under models where the system absorbs duplication costs, Sedna would improve network goodput for equivalent censorship resistance guarantees.

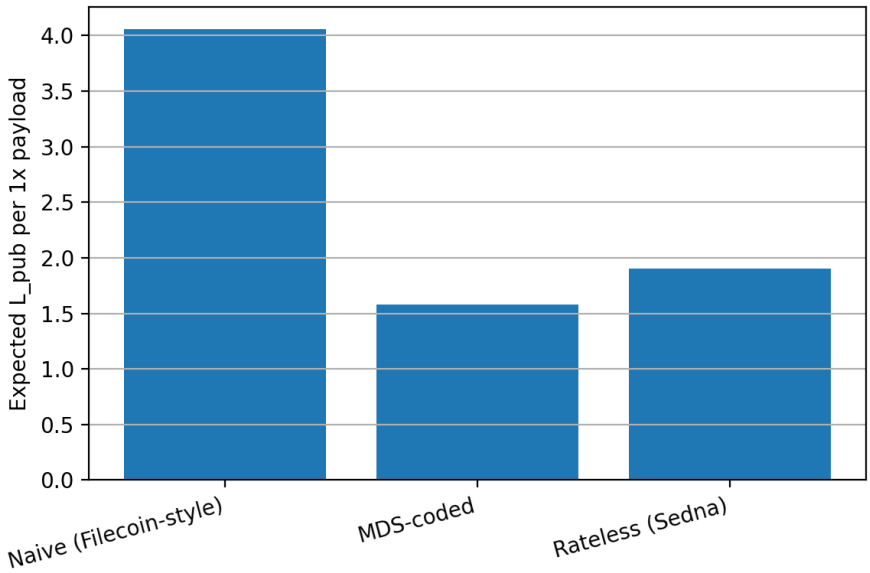

Figure 8: Comparison of replication factors with naive replication and Sedna's MDS and rateless variants, for the same level of censorship resistance. Parameters: $n = 256$, $c_e/n = 0.125$, $\delta = 10^{-9}$.

7.9 Summary

Our evaluation demonstrates that Sedna achieves its design goals across a range of parameters. For medium-to-large payloads, rateless coding reduces bandwidth overhead compared to naive replication while providing strong privacy guarantees against early payload reconstruction. MDS coding achieves slightly lower overhead but requires upfront commitment to code parameters. Computational costs remain practical for tested payload sizes. The flexibility of the rateless scheme allows users to navigate the trilemma of censorship resistance, latency, and cost according to their application-specific requirements.

8 Conclusion

In this work we presented Sedna, a user facing transaction dissemination protocol for MCP blockchains that addresses the fundamental trilemma between censorship resistance, low latency, and reasonable pricing. By replacing whole transaction replication with commitment bound, addressed bundles of verifiable coded symbols, Sedna enables users to navigate this trilemma according to their individual requirements for each transaction. Our analysis establishes several key results. First, we proved that Sedna achieves strong safety guarantees, including header/commitment non-malleability, deterministic resolution of sym-

bol indices, and deterministic execution ordering by inclusion height, all under standard cryptographic assumptions. Second, we derived closed form bounds for per-slot inclusion probability under an assumed effective censor mass, giving users concrete prescriptions for lane selection and symbol allocation to meet target reliability levels. Third, we formalized until-decode privacy, bounding pre-inclusion information leakage to the bits contained in revealed symbols and demonstrating how this reduces the MEV surface compared to transparent mempool dissemination. The bandwidth analysis reveals that Sedna's rateless variant achieves an asymptotic overhead floor of $\frac{1+\varepsilon}{1-c_e/n}$, matching the information-theoretic lower bound for deterministic schemes up to the small rateless coding factor ε. We identified the parameter regimes where each variant proves most bandwidth-efficient, with naive replication remaining competitive for small payloads where metadata dominates and coded approaches becoming advantageous as payload size grows. Sedna is entirely user-facing and requires no protocol-level changes to the underlying MCP consensus, allowing heterogeneous submission strategies to coexist. This positions it as a practical layer that MCP deployments can adopt incrementally. Future work includes empirical evaluation across production MCP systems, integration with existing fee markets, and exploration of adaptive strategies that adjust encoding parameters based on observed network conditions and censor behavior.

Acknowledgments

We would like to thank Ittai Abraham and Mahimna Kelkar for their useful input and conversations regarding this work.

References

[1] I. Abraham, Y. Efron, and L. Ren. The latency cost of censorship resistance. Manuscript, 2025. November 2025.

[2] M. Al-Bassam. Lazyledger: A distributed data availability ledger with client-side smart contracts, 2019. `https://arxiv.org/abs/1905.09274`.

[3] L. Baird. The swirlds hashgraph consensus algorithm: Fair, fast, byzantine fault tolerance. Whitepaper, 2016.

[4] bloXroute Labs. bloxroute private transactions. `https://docs.bloxroute.com/introduction/private-transactions/`, 2023. Accessed 2025-11.

[5] J. Bormet, S. Faust, H. Othman, and Z. Qu. {BEAT-MEV}: Epochless approach to batched threshold encryption for {MEV} prevention. In *34th USENIX Security Symposium (USENIX Security 25)*, pages 3457–3476, 2025.

[6] E. Budish, P. Cramton, and J. Shim. The high-frequency trading arms race: Frequent batch auctions as a market design response. *QJE*, 130(4), 2015.

[7] V. Buterin. Ethereum 2.0 phase 0/1/2 roadmap (gasper overview). `https://ethereum.org/`, 2020. Accessed 2025-11.

[8] C. Cachin and S. Tessaro. Asynchronous verifiable information dispersal. In *Proceedings of the 24th IEEE Symposium on Reliable Distributed Systems (SRDS)*, pages 191–201. IEEE, 2005.

[9] B. Y. Chan and R. Pass. Simplex consensus: A simple and fast consensus protocol. In *TCC*, pages 452–479, 2023.

[10] A. R. Choudhuri, S. Garg, J. Piet, and G.-V. Policharla. Mempool privacy via batched threshold encryption: Attacks and defenses. In *33rd USENIX Security Symposium (USENIX Security 24)*, pages 3513–3529, 2024.

[11] CoW Protocol. Cow protocol: Batch auctions for dex trading. `https://docs.cow.fi/`, 2021. Accessed 2025-11.

[12] CoW Protocol et al. Mev blocker: Private order flow protection. `https://docs.mevblocker.io/`, 2023. Accessed 2025-11.

[13] P. Daian, S. Goldfeder, T. Kell, Y. Li, X. Zhao, I. Bentov, L. Breidenbach, and A. Juels. Flash boys 2.0: Frontrunning in decentralized exchanges, miner extractable value, and consensus instability. In *IEEE S&P*, pages 585–602, 2020.

[14] G. Danezis, G. Giuliari, E. K. Kogias, M. Legner, J.-P. Smith, A. Sonnino, and K. Wüst. Walrus: An efficient decentralized storage network. *arXiv preprint arXiv:2505.05370*, 2025.

[15] G. Danezis, L. Kokoris-Kogias, A. Sonnino, and A. Spiegelman. Narwhal and tusk: a dag-based mempool and efficient bft consensus. In *EuroSys*, page 34–50, 2022.

[16] Ethereum Foundation. Proposer-builder separation (pbs) — ethereum research & specs. `https://ethresear.ch/t/proposer-builder-separation/`, 2023. Accessed 2025-11.

[17] D. Feist et al. Eip-4844: Shard blob transactions (proto-danksharding). `https://eips.ethereum.org/EIPS/eip-4844`, 2023. Accessed 2025-11.

[18] Flashbots. Flashbots protect rpc. `https://docs.flashbots.net/flashbots-protect/`, 2022. Accessed 2025-11.

[19] Flashbots. Mev-boost: Open source relay middleware for proposer-builder separation. `https://docs.flashbots.net/`, 2022. Accessed 2025-11.

[20] P. Garimidi, J. Neu, and M. Resnick. Multiple concurrent proposers: Why and how. *CoRR*, abs/2509.23984, 2025.

[21] N. Giridharan, F. Suri-Payer, I. Abraham, L. Alvisi, and N. Crooks. Autobahn: Seamless high speed BFT. In *SOSP*, pages 1–23, 2024.

[22] A. Gągol, D. Leśniak, D. Straszak, and M. Swietek. Aleph: Efficient atomic broadcast in asynchronous networks with byzantine nodes. In *AFT*, page 214–228, 2019.

[23] G. Goren, A. Hariri, T. D. Hartley, R. Kappiyoor, A. Spiegelman, and D. Zmick. Shelby: Decentralized storage designed to serve. *arXiv preprint arXiv:2506.19233*, 2025.

[24] I. Keidar, E. Kokoris-Kogias, O. Naor, and A. Spiegelman. All you need is dag. In *PODC*, page 165–175, 2021.

[25] M. Kelkar, S. Deb, S. Long, A. Juels, and S. Kannan. Themis: Fast, strong order-fairness in byzantine consensus. In *CCS*, pages 475–489, 2023.

[26] M. Kelkar, F. Zhang, S. Goldfeder, and A. Juels. Order-fairness for Byzantine consensus. In *CRYPTO*, pages 451–480, 2020.

[27] Q. Kniep, J. Sliwinski, and R. Wattenhofer. Solana alpenglow consensus, 2025.

[28] M. Luby. LT codes. In *FOCS*, page 271, 2002.

[29] M. Luby, A. Shokrollahi, M. Watson, et al. Rfc 6330: Raptorq forward error correction scheme for object delivery. IETF RFC, 2011.

[30] A. Miller, Y. Xia, K. Croman, E. Shi, and D. Song. The honey badger of BFT protocols. In *ACM CCS*, 2016.

[31] K. Mu, B. Yin, A. Asheralieva, and X. Wei. Separation is good: A faster order-fairness byzantine consensus. In *NDSS*, 2024.

[32] Mysten Labs. Sui: A platform for high-performance smart contracts. `https://github.com/MystenLabs/sui/blob/main/doc/paper/sui.pdf`, 2022. Accessed 2025-11.

[33] Mysten Labs. "sui". `https://github.com/mystenlabs/sui`, 2025.

[34] J. Peterson et al. Rfc 5053: Raptor forward error correction scheme. IETF RFC, 2007.

[35] I. S. Reed and G. Solomon. Polynomial codes over certain finite fields. In *J. Soc. Ind. Appl. Math.*, 1960.

[36] B. Riva, A. Sonnino, and L. Kokoris-Kogias. Seahorse: Efficiently mixing encrypted and normal transactions.

[37] A. Shokrollahi. Raptor codes. *IEEE Trans. Info. Theory*, 52(6), 2006.

[38] V. Shoup. Sing a song of simplex. In *DISC*, page 37:1–37:22, 2024.

[39] Shutter Network. Shutter: Threshold encryption for preventing mev. `https://docs.shutter.network/`, 2021. Accessed 2025-11.

[40] Solana Labs. Solana turbine: Block propagation protocol. `https://docs.solana.com/cluster/turbine`, 2023. Accessed 2025-11.

[41] A. Spiegelman, N. Giridharan, A. Sonnino, and L. Kokoris-Kogias. Bullshark: Dag bft protocols made practical. In *CCS*, page 2705–2718, 2022.

[42] C. Stathakopoulou, T. David, and M. Vukolic. Mir-BFT: High-throughput BFT for blockchains. *arXiv preprint arXiv:1906.05552*, 92, 2019.

[43] S. B. Wicker and V. K. Bhargava. *Reed–Solomon Codes and Their Applications*. IEEE Press, 1995.

[44] M. Yin, D. Malkhi, M. K. Reiter, G. Golan-Gueta, and I. Abraham. Hotstuff: BFT consensus in the lens of blockchain. In *PODC*, pages 347–356, 2019.