SoK: Encrypted Mempools Through the MEV Lens

Jonathan Passerat-Palmbach
Flashbots, Imperial College London

Abstract

MEV and censorship, fuelled by public mempool visibility, remain existential threats to Ethereum and have recently started to spread to its layer-two ecosystem. Encrypted mempools promise to conceal transaction content until ordering is final. While this sounds appealing, their viability rests on cryptographic, economic, and deployment trade-offs.

This paper systematises the evolution of threshold-encrypted mempools, from early schemes such as Shutter and Ferveo to the most recent research, and analyses how successive iterations have resolved bottlenecks like committee communication overhead, lack of pending transaction privacy, and position-dependent encryption. We highlight a convergence along four design axes, namely batched decryption to mitigate latency, silent setup to eliminate the complexity of distributed key generation, epochless encryption to remove position dependency, and collision-free encoding to prevent slot-collision censorship. We further survey the active Ethereum deployment debate, including EIP-8105 and the LUCID headliner submission, and map the requirements raised there onto the cryptographic corpus.

We conclude by exposing a critical limitation common to all current proposals: blind ordering and binary decryption together suppress not only the toxic part of MEV that motivated encrypted mempools, but also the same-block auction mechanisms that return value to users and sustain geographic decentralisation of the network. We argue for a programmable-privacy direction in which encrypted state can be acted on by trusted code before plaintext revelation.

2012 ACM Subject Classification Security and privacy → Cryptography; Security and privacy → Distributed systems security; Applied computing → Digital cash

Keywords and phrases Encrypted mempools, threshold encryption, MEV, Ethereum, proposer-builder separation, order-flow auctions, programmable privacy

Related Version *Published Version*: `https://doi.org/10.4230/LIPIcs.AFT.2026.30`

Acknowledgements The author thanks Quintus, Thomas, Mark, Sarah, and Luis, together with the authors who kindly presented their work to our reading group, for the discussions that shaped this systematisation.

1 Introduction

Maximal extractable value (MEV) is the value that a block proposer, or any party with influence over transaction ordering, can extract by reordering, censoring, or interleaving pending transactions, often at the expense of the users who submitted them [20]. The *mempool* is the network's broadcast queue of unconfirmed transactions awaiting inclusion in a block. On Ethereum, the proposer (the validator selected for each slot) can outsource block construction to a third-party *builder* under the proposer-builder separation (PBS) architecture [15], and the builder selects which transactions to include and in what order. We organise the design space for handling the incentives MEV creates in a permissionless network into three responses. Two passive responses let MEV manifest as spam, with *searchers*, the specialised actors who scan pending flow for extractable opportunities, competing for the same opportunity by submitting redundant transactions whose cost the chain absorbs in failed transactions and inflated gas usage, or let it concentrate into a latency race in which rewards accrue to the small number of operators who reach the proposer fastest, degrading the geographic distribution of the validator set [67]. The active response is to run auctions

that mediate searcher competition and route the resulting value back to the user, an approach that has gained adoption through order-flow auctions [36] delivered by the Flashbots Auction [27], MEV-Blocker [19], and other providers.

These three responses map onto distinct mempool architectures, each striking a different balance between speed, decentralisation, and privacy. The public mempool is fast and decentralised but exposes pending transactions to any observer. As a result, the bulk of MEV-bearing transaction flow on Ethereum is now routed through out-of-protocol private channels concentrated in a small set of high-performance builders [51, 52]. Private mempool lanes are fast and shield content from public view, but require the user to trust the operator not to act on what it sees. Advocates of encrypted mempools argue that this trust model is brittle at scale, since every party with visibility of pre-confirmation transaction content has both the ability and the incentive to extract value once stakes are large enough [61]. They further state that the harms it enables, sandwich attacks on retail trades, real-time censorship, and opaque order-flow agreements, are central rather than residual to user experience [60, 46]. The combination of privacy and decentralisation, the third combination of the design space introduced above, is the one encrypted mempools aim to deliver by hiding transaction content cryptographically rather than by trust.

One currently deployed approach combining privacy and decentralisation is BuilderNet [28], a multi-operator block-building network in which each operator runs the same builder software inside a trusted execution environment (TEE) and users encrypt their transactions to the enclave's public key. The enclave can order, simulate, and place searcher backruns against fresh state while the plaintext remains hidden from the operator. The resulting MEV is refunded to the user. The TEE substrate moves the trust assumption from any individual builder to the hardware supply chain, which is a non-trivial shift, not one that every user accepts. Encrypted mempools, the subject of this paper, are the cryptographic alternative that removes the hardware-trust dependency.

The intuition behind an encrypted mempool is to hide the content of pending transactions until their ordering is committed, so that the information an adversary would need to mount a reordering or front-running attack is unavailable during the ordering window. The construction admits several cryptographic instantiations, and we focus on the threshold-encrypted mempool, in which a committee of validators jointly holds the decryption key and releases the plaintext once ordering is fixed. This variant has attracted the most attention because the validator set already has the right structure to serve as the committee, and the trust assumption maps cleanly onto the existing proof-of-stake trust model. We argue in this paper that, even setting aside the cryptographic gaps the literature has yet to close, threshold-encrypted mempools as currently proposed operate at the wrong design point: they suppress not only the toxic MEV they were designed to neutralise but also the auctions that today route value back to users. Instead, we believe a more desirable property is to preserve auction capabilities through the use of *programmable privacy*, in which secure and trusted code, an enclave or a cryptographic primitive, can operate on encrypted state before plaintext revelation.

The threshold-encrypted-mempool literature has grown rapidly in the last three years, organised around two parallel lines of work that resolve the principal bottlenecks of the original Shutter attempt [61]. Batched threshold encryption, introduced by Choudhuri et al. [17], makes the communication cost of decryption independent of batch size and removes the pending-transaction-privacy weakness present in per-epoch designs. Silent threshold encryption [33] eliminates the interactive distributed key generation step and lets each committee member publish a hint from which a joint public key can be derived

non-interactively. Subsequent work has combined these two lines and explored hybrid trust models in which the user retains a self-decryption option as a hedge against committee unavailability [64]. The rapid succession has left fragmented terminology, overlapping threat models, and unresolved trade-offs that the protocol-developer community is now actively weighing as it considers whether any of these constructions is ready for adoption at the consensus layer.

In this paper, we first develop a four-axis taxonomy of threshold-encrypted mempool constructions that organises the design space by setup interactivity, batching, epochless encryption, and post-quantum security, and we identify a convergent set of properties that practitioners have settled on but that no current construction jointly satisfies (Section 2). Second, we synthesise the protocol-developer perspective by surveying the Ethereum community proposals on the table and the requirements they articulate, showing how the cryptographic constructions match or fail to match those requirements (Section 3). Third, we develop an execution-quality argument that the design choice the community proposals share, namely blind ordering, imposes costs that the present proposals do not internalise even under a cryptographically perfect construction (Section 4). We conclude with four open problems spanning post-quantum security, the cryptographic-and-economic gap, network-layer metadata leakage, and decryption-aggregation cost (Section 5).

2 The cryptographic landscape

Threshold-encrypted mempools instantiate the encrypted mempool idea by entrusting a committee of validators with shared decryption capability, which fits into the existing trust assumptions of proof-of-stake consensus, since the same stake-weighted validator set can play both roles. While threshold decryption protocols have long been studied in cryptography, this section organises the literature on the specific cryptographic constructions required to meet the requirements of encrypted mempools.

Three design choices distinguish the constructions that follow, namely where the decryption committee sits, how much it must coordinate during setup, and when in the block lifecycle a ciphertext becomes openable. Each construction has been pushed against three failure modes that the deployment context generates: leakage of transactions that never make it into a block, ciphertexts that must commit to a target block height before the user knows which one, and ciphertexts that must commit to a slot index within a batch and so become vulnerable to slot-collision censorship. The argument of this section is that the literature has converged on four desirable properties, namely *silent setup*, *batched decryption*, *epochless encryption*, and *collision-free encoding*, each of which is individually achievable but that no single scheme attains together with post-quantum security; Table 1 makes the convergence explicit. Section 2.2 introduces the batched-decryption line, Section 2.3 introduces the silent-setup line, Section 2.4 covers their combination, Section 2.5 traces the path from epoch dependency to collision-free encoding, and Section 2.6 examines the two post-quantum candidates.

2.1 Problem statement and the first deployments

Shutter Network has been running on Gnosis Chain since October 2022 and is the first production threshold-encrypted mempool. Its protocol, documented in [23], builds on threshold Boneh-Franklin *Identity-Based Encryption* (IBE), with a permissioned committee of *Keypers* holding shares of the IBE master secret. Users encrypt to an epoch identifier, and at the end of each epoch the committee releases a single epoch decryption key that opens

every ciphertext bound to that epoch. The design trades a once-per-epoch coordination cost for a single $O(1)$ broadcast per epoch, regardless of how many transactions the epoch carries. Key release is on a deterministic schedule, synchronised against block finality.

The same release schedule is the source of Shutter's most-discussed limitation. Choudhuri, Garg, Piet, and Policharla [17] analyse Shutter and observe that the epoch key opens *every* ciphertext bound to its epoch, whether the transaction was included on chain or not. A user whose transaction failed to land in a block, for example because higher-fee transactions filled the block before theirs, has their cleartext revealed to every party watching the chain anyway, exposing trading intent that an attacker can replay against later submissions. They formalise the missing property as *pending transaction privacy*: a ciphertext that does not enter a block must not be decryptable by the committee. Shutter's deployment substantially predates a formal whitepaper, and the third-party analysis of [17] reconstructs the construction from production code, with a written protocol description by the original co-authors following only in [23].

A parallel early line, due to Bebel and Ojha [5], designs Ferveo for Tendermint-based proof-of-stake chains, with deployment in the Anoma stack. Ferveo runs a per-epoch distributed key generation against the active validator set and threshold-decrypts individual ciphertexts. The construction achieves pending transaction privacy by virtue of decrypting transactions one at a time after inclusion, but at the cost of $O(nB)$ communication for a batch of B transactions and a fresh DKG whenever the validator set rotates. Ferveo and Shutter therefore sit at opposite corners of the same trade-off: Shutter is communication-cheap on decryption but leaks unincluded transactions, while Ferveo preserves privacy slot by slot but pays for it linearly in batch size and in coordination per epoch.

The two failure modes of pending transaction privacy and per-epoch coordination motivate the remainder of the literature.

2.2 Batched decryption

Choudhuri et al. [17] introduce *batched threshold encryption* (BTE) as the formal alternative to both IBE-style epoch decryption and naive per-ciphertext threshold decryption. In a BTE scheme, a committee of n members chooses a subset of B ciphertexts from a larger mempool and jointly produces a partial decryption of total size $O(n)$, independent of B, that lets anyone open exactly that subset, while ciphertexts outside the chosen subset remain semantically secure. The construction in [17] casts the decryption rule as a weak form of witness encryption for a relation of shape "you can decrypt iff you know a BLS signature [11] on a KZG commitment [40] to the batch tags". The partial decryption is a single group element per committee member, and security is proved in the programmable random oracle model. The setup is a per-epoch multiparty computation that runs secret multiplications on shares to produce a batch-bound common reference string. The setup is more complex than a KZG powers-of-tau ceremony: no party ever holds the trapdoor in clear, and the shared powers are computed via Beaver-style triples. The scheme is practical only on chains willing to amortise such an MPC over a relatively long epoch.

The same group of authors removes the per-epoch MPC in their follow up paper [18]. The change consists in pushing the KZG commitment from the public statement into the witness and securing the resulting check with a shifted BLS signature [18]. This reduces the BTE setup to a single distributed key generation that holds across all future batches. The reported partial decryption is 48 bytes per committee member, with partial decryption taking roughly 3.2 seconds per committee member and reconstruction of the full plaintext batch taking a further 3.0 seconds for a 500-transaction batch in their single-threaded Rust microbenchmark,

against a malicious adversary in the algebraic group model. The construction retains the BTE interface from [17], including the requirement that each ciphertext commit to a specific batch identifier; ciphertexts that fail to land in their committed batch are undecryptable thereafter. This is a milder failure mode than the Shutter epoch leak: a stuck transaction is no longer revealed, but it is also not recoverable, and the user must resubmit. This user-experience cost may block adoption in real-world deployments, as we revisit in Section 3. A weighted variant of the BTE interface, in which the threshold counts stake rather than committee heads, appears in [2] and is treated in Section 2.5.

What both BTE papers share is a design choice that became dominant in everything that followed. The decryption rule is expressed as a small set of pairing-product equations with linear verification, instantiated as threshold BLS signatures over a KZG commitment to the polynomial whose roots are the batch tags, with the message payload concealed via a witness encryption keyed on this relation. The compiler is general, and almost every subsequent pairing-based construction in this section reuses it, varying only what is in the witness and what is in the public statement. Garg et al. [32] establish the unifying perspective on this design choice by recognising these constructions as witness encryptions for "linearly verifiable" relations, composed from a small set of cryptographic gadgets such as signatures, inner products, and degree checks.

2.3 Silent setup

Even a single one-shot DKG is operationally heavy. A DKG requires synchronous interaction across the entire committee, and any committee rotation, whether intentional or membership-driven, forces a fresh proactive secret-sharing protocol [38] to migrate the shares. On Ethereum, where the validator set rotates every epoch and tens of thousands of validators come online and offline asynchronously, threshold cryptography that requires DKG-style setup does not map cleanly to the underlying network model.

Garg, Kolonelos, Policharla, and Wang [33] answer this with *silent threshold encryption* (STE). Each committee member independently generates a keypair under standard public-key infrastructure and publishes a constant-sized *hint* alongside their public key. Encryption is to the deterministic aggregate of the hints, and decryption is non-interactive in the sense that any party with the hints can verify and combine partial decryptions. The construction is pairing-based and rests on a novel *signature-based witness encryption* primitive that admits linear verification, with security proven in the generic group model. The reported encryption cost is under 7 ms in a Rust implementation. The headline limitation is that the scheme is not batched: communication is $O(n)$ per ciphertext, recovering the standard threshold encryption asymptotics. We return to the deployment cost of distributing the per-validator hints in Section 3. Independent of mempool privacy, the same silent-setup primitive has been proposed as a replacement for the relay layer in proposer-builder separation, where a small committee with no trusted dealer can enforce fair-exchange properties between builders and proposers [51].

Hall-Andersen, Simkin, and Wagner [37] revisit STE with a different set of design targets. The construction avoids both indistinguishability obfuscation, on which the earlier STE of Reyzin, Smith, and Yakoubov [56] rests, and the generic group model, on which the proof of [33] rests, and which the authors argue is too strong an idealisation to support a deployment. They introduce a new security notion called *one-shot adaptive security*, strictly stronger than static corruption but weaker than fully adaptive corruption. The construction is, by the authors' admission, only CCA1-secure, and decryption communication remains $O(n)$ per ciphertext; its post-quantum instantiation and parameter sizes are treated in Section 2.6.

Concurrently, Waters and Wu [65] present a third silent threshold construction, also pairing-based but proved selectively secure under q-type assumptions in the plain model, that is, with no idealised heuristic such as the generic group or random oracle model to lean on. Their distributed threshold encryption scheme has ciphertexts of four source-group elements plus a short tag, and supports access structures beyond plain thresholds, namely arbitrary monotone Boolean formulae over the committee. Such a policy is any rule built from ANDs and ORs of committee members, with no negations, so that "members 1 and 2, or any of the members 5 through 9" is expressible where a plain t-of-n threshold is not [6]. It identifies weighted thresholds, important for stake-weighted committees, as an open problem specific to this plain-model line: earlier silent constructions do support weighted policies, but only through SNARK-based aggregation. Waters and Wu instead reach their plain-model proof with a SNARK-free technique, which they can unfortunately make work only for unweighted thresholds.

The three silent constructions disagree on the right security baseline. [33] accepts the generic group model in exchange for a concretely efficient pairing-based scheme. [37] pays larger ciphertexts and weaker chosen-ciphertext security for post-quantum security under standard lattice and isogeny assumptions. [65] stays in the pairing world but trades generic-group hardness for q-type assumptions in the plain model. A practical deployment must choose between these three positions explicitly, since they are not interchangeable.

2.4 Combining batched and silent

Batched decryption and silent setup are orthogonal properties, and a scheme that achieves both is the natural next step. Bormet, Choudhuri, Faust, Garg, Othman, Policharla, Qu, and Wang [13] present BEAST-MEV, the first scheme that bridges the two. The construction couples a homomorphic variant of [33] with a binary-decomposition encoding inspired by the BTE line, and the resulting Silent Batched Threshold Encryption primitive admits a one-time silent setup, batched decryption, and security in the generic group model against a fully malicious adversary. Reported figures are encryption in 16.4 ms and a 129.5-second reconstruction time for a batch of 512 transactions in Rust, the latter substantially slower than the corresponding figures in [18], which is the cost paid for the silent setup. BEAST-MEV remains pairing-based and is therefore not post-quantum.

The substance of [13], beyond the headline merger of the batched decryption and silent setup properties, is a sequence of optimisations that bring the cost of silent batched decryption into the same order of magnitude as the non-silent BTE line. The construction modifies the witness-encryption relation of [33] to admit additive homomorphism on a shared randomness scalar across ciphertexts, so that a single batched partial decryption serves all ciphertexts in a sub-batch of size ℓ, with $\ell = 8$ in the prototype. The batch is organised into $O(B/\ell)$ such sub-batches and the partial decryption is reconstructed across them. Randomness reuse across sub-batches and a multiplicative-sharing trick that avoids $\mathbb{G}_T$ elements bring the per-ciphertext partial decryption down to a single $\mathbb{G}_1$ element of 48 bytes, with 0.55 ms decryption time per ciphertext at $B = 512$. The underlying tag encoding is inherited from [14], however, and the resulting scheme inherits the slot-collision censorship attack surface analysed in the next subsection. Later constructions [3, 12], treated in Section 2.5, retain the two-mode structure of a one-time committee setup and per-batch work within a single pairing-product check.

2.5 Position dependency and slot-collision censorship

Bormet, Faust, Othman, and Qu [14] present BEAT-MEV, the first epochless BTE scheme, using key-homomorphic puncturable *PseudoRandom Functions* (PRF) composed with threshold ElGamal. The puncturable PRF acts as a per-ciphertext one-time pad whose key the committee can collectively reconstruct, and security against rogue ciphertexts follows from the puncturability of the key-homomorphic PRF. Encryption is reported in under 2 ms on standard hardware, and the scheme requires only a one-time setup that supports all future batches.

The price of removing the epoch identifier is that the user must commit to something else. In [14] each ciphertext carries an index i from a polynomially bounded slot namespace, and no two ciphertexts in the same batch can share a slot. Removing the epoch dependency uncovers a subtler failure mode. The authors classify the resulting failure modes carefully: an *honest collision* occurs when two users independently pick the same i and only one transaction survives, and an *adversarial collision* occurs when a watcher observes the mempool and submits a dummy ciphertext at the same slot as a victim, undercutting them on fee so that a rational proposer drops the victim. The adversarial case is a clean cryptographic-layer censorship vector: the attacker need not be a block builder, need not understand the transaction contents, and need not be in any particular network position.

The TrX system of Fernando, Policharla, Tonkikh, and Xiang [26] accepts a position commitment but invests in system-level integration with a high-throughput BFT consensus protocol. Each batch of ciphertexts is decrypted under a per-batch *rerandomised* KZG setup, indexed by a fresh scalar κ_{dc} per decryption context, which sidesteps the requirement that the user encrypt to a specific block height. Tags are derived from $H(\mathrm{pk})$ together with an ephemeral signature on the ciphertext, not from the block height or a polynomially bounded slot namespace. The resulting ciphertext is two $\mathbb{G}_2$ elements plus a symmetric payload, the smallest among the BTE constructions we are aware of, and is instantiated on the BN254 curve. The system-level observation is that the digest and the evaluation proofs of the BTE relation depend only on the published ciphertexts and not on the threshold shares, so both can be pipelined alongside the consensus voting rounds. The decryption shares are then released optimistically together with the final consensus vote, recovering a fast-path in which the residual pairing work after consensus completes amounts to a few operations per ciphertext. The reported end-to-end overhead is 27 ms, roughly 14% over the plaintext baseline, on a 50-node wide-area-network deployment against AptosBFT. The authors note that this construction resolves an open question, raised in earlier work on context-dependent threshold decryption, of whether such a primitive can be instantiated efficiently in a low-threshold regime. The trade-off is a common reference string that grows with the number of distinct decryption contexts per public key; on chains with two-hour epochs and ten blocks per second, this remains below one gigabyte and can be streamed into memory per epoch.

Agarwal et al. [2] extend the BEAT-MEV line to weighted committees (Weighted BTE - WBTE), an axis the rest of this section leaves implicit, keeping its one-time setup, epochless encryption, and single-group-element partial decryptions while weighing members by stake rather than equally. The scheme remains index-dependent: honest collisions surface as a small failure rate that a generalised cuckoo-hashing assignment drives below 2^{-40}, and the adversarial slot-collision vector of [14] carries over. The most recent line removes position commitment entirely. Boneh, Nema, Roy, and Tas [12] achieve epochless and collision-free BTE via partial-fraction decomposition over rational functions, extending an earlier construction of Jutla, Nema, and Roy [39]. A single group element acts as the pre-decryption

key for the entire batch; ciphertexts are two group elements plus a tag and are claimed to be CCA-secure under the construction's own definition. The construction requires an interactive setup, presented with MPC-based inversions on shares but also supporting a variant that needs only a single MPC multiplication, on par with the multiplication-only setups elsewhere in this section. The original presentation has $O(B^2)$ decryption, which the authors show can be improved to $O(B \log B)$ via FFT techniques, with a shorter-ciphertext variant added in a later revision. Policharla [55] presents an independent construction with the same epochless and collision-free properties, building directly on punctured powers of τ in pairing groups without partial fractions. The setup is interactive but multiplication-only, ciphertexts are one element each in $\mathbb{G}_1$ and $\mathbb{G}_T$, and decryption is $O(B \log B)$ via FFT. Security rests on the Decisional Bilinear q-Power Diffie-Hellman assumption [9], which is the same q-type family that underpins KZG, PLONK, and Ethereum's EIP-4844 blob commitments, rather than on a new q-type assumption specific to the construction. Concurrently, Agarwal, Das, Gilkalaye, Rindal, and Shoup [3] present BTX, also based on punctured powers of τ, achieving the same shortest ciphertext size within the collision-free line of $|\mathbb{G}_1| + |\mathbb{G}_T|$,[1] with $O(B \log B)$ decryption and concrete performance within a small factor of [55].

The three constructions [12, 55, 3] demonstrate that epochless and collision-free are jointly achievable in the pairing-based setting at $O(B \log B)$ decryption complexity. The concrete differences at $B = 512$ are small: end-to-end batch decryption of 2.80 s for [55], 3.61 s for the shorter-ciphertext variant of [12], and roughly half the time of [55] for [3] on the same hardware as reported by Policharla. A further observation in [55] separates the threshold protocol from the decryption work: the FFT-style cross-term computation depends only on the public ciphertexts and reference string, not the secret shares, so it runs in parallel with the threshold protocol, cutting post-protocol latency by roughly $12\times$ at $B = 2048$ and leaving roughly B pairings on the critical path.

When it comes to ingenious tricks that make live deployments viable, it is worth considering the shared ideas and lineage between some of these BTE constructions.

The remaining design axis is post-quantum security.

2.6 Post-quantum paths

The constructions described above are all pairing-based, with security resting on discrete-logarithm-flavoured assumptions in bilinear groups. Two independent attempts have been made to step outside this regime.

Boneh, Laufer, and Tas [10] take the first of two post-quantum routes by tackling the batched-decryption side directly. The paper presents three epochless batch-decryption constructions: a key-policy ABE warm-up, a headline lattice scheme proved secure under the Learning with Errors assumption and therefore plausibly post-quantum, and a trilinear-map construction whose only known instantiation requires indistinguishability obfuscation [26].

Two observations from [10] matter for deployment. First, the lattice construction inherits a *succinctness vulnerability* once combined with the sub-batch techniques required to lift the encryptor-coordination requirement: an adversary that submits a full batch of ciphertexts under the same identity tag forces the pre-decryption key to grow linearly in the batch size, and the natural mitigation of capping the pre-decryption at the first $O(\log \ell)$ ciphertexts per tag opens a censorship vector in which $O(\log \ell)$ adversarial ciphertexts on a victim's tag

[1] The BTE papers report ciphertext size differently: [12] counts the $\mathbb{G}_T$ mask that TrX hashes away, and TrX excludes its ephemeral verification key and signature. Under a uniform convention the TrX ciphertext remains the smallest overall, so "shortest" here is within the collision-free line.

can exclude the victim from the decryptable set. Second, thresholdising the construction by secret-sharing a random matrix's trapdoor is possible but needs two rounds of interaction during pre-decryption, and the authors flag non-interactive lattice thresholding as an explicit open problem. The headline trade-off against the pairing line is parameter size: the pairing-based BTE of [55] has a ciphertext of $|\mathbb{G}_1| + |\mathbb{G}_T|$ plus a $2|\mathbb{F}|$ NIZK proof, in the hundreds of bytes on BLS12-381, against the kilobyte-scale parameters of [10].

Hall-Andersen et al. [37] study the orthogonal axis: where [10] targets batched decryption and accepts lattice-scale parameters, [37] targets silent setup and engineers a modular blueprint with two instantiations: a classical one on elliptic curves, and a post-quantum one combining the Kyber lattice KEM with an isogeny-based key exchange. The reported ciphertext sizes are 1.072 MB for the classical elliptic-curve instantiation and 1.431 MB for the post-quantum one, both at $n/3$ corruption and a 1 MB plaintext. The paper section on encrypted mempool deployment makes the architecture explicit: the user picks an ephemeral symmetric key K, encrypts the transaction body with AES under K, and uses the silent threshold scheme to encrypt only K, not the body. The small key-encapsulation ciphertext goes on chain via the mempool, and the larger symmetric ciphertext can either go on chain or sit on a separate availability layer. The technical contribution that makes this practical is the construction of small recipient public keys, replacing the linear-sized hints that [33] required.

The two post-quantum constructions are not interchangeable. [10] provides post-quantum batched decryption at kilobyte-scale parameters, and [37] provides post-quantum silent setup without batching. A construction combining silent setup, batched decryption, collision-free encoding, and post-quantum security under standard assumptions does not exist in the current literature to the best of our knowledge.

2.7 A four-axis taxonomy

The evolution of the field is naturally summarised along four binary axes introduced in this section, namely *setup* (per-epoch MPC, one-time DKG, or silent PKI-based), *decryption* (batched in the BTE sense versus standard per-ciphertext), *epochless* (whether the encryptor must commit to a target block), and *collision-free* (whether the encryptor must commit to a slot index within a batch),[2] together with a fifth column for post-quantum security under standard cryptographic assumptions. Table 1 places every scheme along these axes, alongside the security notion and idealised model in which the construction is proved secure, the committee-communication complexity of decrypting a batch, and the computational cost of that decryption. GGM, AGM, ROM, and "standard" in the security column denote the generic group, algebraic group, random oracle, and plain models respectively.

Three observations follow directly from Table 1. First, within the pairing-based corpus, silent setup, batched decryption, and the collision-free property each appear in multiple schemes, but no construction achieves all three together: [13] is silent and batched but not collision-free, [3, 55, 12] are batched and collision-free under a one-time DKG or MPC setup, and the silent constructions [33, 37, 65] do not batch. Combining the silent-batched line with the collision-free encoding of [3, 55, 12] appears to be the natural near-term direction. Second, post-quantum security is essentially orthogonal to the other axes as currently constructed: [10] achieves it in the BTE line under a one-time DKG, and [37] achieves it in the silent line

[2] Following [3]: no user-chosen index in the ciphertext, ruling out honest user collisions and the adversarial slot-collision censorship of [14].

Table 1 Taxonomy of threshold-encrypted mempool constructions, grouped by the lineages traced in this section. The four binary axes (setup, decryption, epochless, coll.-free) characterise the design choices analysed in this section, with post-quantum security as a fifth column. **Bold** marks a cell meeting the corresponding convergent property (silent setup, batched decryption, epochless encryption, collision-free encoding) or post-quantum security: no row is bold across all five columns, which is the convergence gap discussed in Section 2.7. The two closest deployment candidates under the Wagner-Ma criteria (Section 3) are set in **bold type** and marked †. The "Security" column reports the security notion together with the idealised model in which the construction is proved secure, and B-IND-CCA denotes the construction-specific batched chosen-ciphertext notion used by each BTE paper. The "Committee comm." column reports the total committee bandwidth to decrypt a batch of B ciphertexts, and the "Decryption cost" column the total computational cost of that decryption over the same unit, counting group operations and pairings, with FFT-style optimisations applied where the authors report them and $\tilde{O}$ suppressing logarithmic factors. "n/a" under "Coll.-free" marks schemes without a BTE-style batch interface, where index collision is not a meaningful failure mode. "Yes (lattice)" under "Post-quantum" indicates a concrete post-quantum instantiation, and "Yes (mod.)" indicates a modular blueprint instantiable post-quantum under standard assumptions. BLT's headline lattice construction proves selective IND-CPA in the random oracle model under LWE, weaker than the B-IND-CCA notion of the pairing line, with CCA2 reachable via the standard transforms the authors cite. The paper reports no benchmark or tight batch-size asymptotic, so the "Decryption cost" entry is the polynomial-in-B bound its lattice-arithmetic algorithm admits. WBTE's headline property, stake-weighted thresholds, is orthogonal to the four axes and not shown as a column.

Scheme	Setup	Decryption	Epochless	Coll.-free	PQ	Security	Committee comm.	Decryption cost
Epoch-bound pioneers (Section 2.1)								
Shutter [23]	1-time DKG	IBE-batched	No	n/a	No	IND-CPA, ROM	$O(1)$ per epoch	$O(n+B)$
Ferveo [5]	per-epoch DKG	Standard	No	n/a	No	IND-CCA, ROM	$O(nB)$ per batch	$O(nB)$
Batched decryption (Sections 2.2 and 2.5)								
CGPP24 [17]	per-epoch MPC	**Batched**	No	No	No	B-IND-CCA, ROM	$O(n)$ per batch	$O(B \log B)$
CGPW24 [18]	1-time DKG	**Batched**	No	No	No	B-IND-CCA, AGM+ROM	$O(n)$ per batch	$O(B \log B)$
BEAT-MEV [14]	1-time DKG	**Batched**	**Yes**	No	No	B-IND-CCA, AGM+ROM	$O(n)$ per batch	$O(B^2)$
TrX [26]	1-time DKG	**Batched**	**Yes**	**Yes**	No	B-IND-CCA, GGM	$O(n)$ per batch	$\tilde{O}(n+B)$
BLT [10]	1-time DKG	**Batched**	**Yes**	**Yes**	**Yes (lattice)**	sel. IND-CPA, ROM	$O(n)$ per batch	poly(B) lattice
WBTE [2]	1-time MPC	**Batched**	**Yes**	No	No	B-IND-CCA, AGM+ROM	$O(n)$ per batch	$O(B \log B)$
Silent setup (Section 2.3)								
GKPW STE [33]	**silent**	Standard	**Yes**	n/a	No	IND-CPA, GGM	$O(n)$ per ciphertext	$O(n)$
HSW STE† [37]	**silent**	Standard	**Yes**	n/a	**Yes (mod.)**	IND-CCA1, standard	$O(n)$ per ciphertext	$O(n)$
Waters-Wu [65]	**silent**	Standard	**Yes**	n/a	No	IND-CPA, plain	$O(n)$ per ciphertext	$O(n)$
Silent and batched (Section 2.4)								
BEAST-MEV† [13]	**silent**	**Batched**	**Yes**	No	No	B-IND-CCA, GGM	$O(n)$ per batch	$O(B^2)$
Epochless and collision-free (Section 2.5)								
BNRT [12]	1-time MPC	**Batched**	**Yes**	**Yes**	No	B-IND-CCA, AGM+ROM	$O(n)$ per batch	$O(B \log B)$
Policharla 26 [55]	1-time MPC	**Batched**	**Yes**	**Yes**	No	B-IND-CCA, AGM+ROM	$O(n)$ per batch	$O(B \log B)$
BTX [3]	1-time DKG	**Batched**	**Yes**	**Yes**	No	B-IND-CCA, AGM+ROM	$O(n)$ per batch	$O(B \log B)$

for standard threshold encryption only, so the four-property convergence with post-quantum security under standard assumptions remains open. Third, every construction in Table 1 that achieves both byte-scale ciphertexts and batch-independent decryption communication is pairing-based: the lattice scheme of [10] reaches succinct pre-decryption keys at kilobyte-scale parameters, the post-quantum silent line of [37] sits at megabyte-scale, and the smallest pairing-based ciphertexts of [55, 3] at $|\mathbb{G}_1| + |\mathbb{G}_T|$ are roughly 700 bytes on BLS12-381. Pairings therefore remain a critical primitive for the combination of byte-scale ciphertexts and batch-independent decryption communication that real-world deployments require, and a post-quantum replacement would have to give up one of these properties or develop a new technique.

3 The blockchain protocol developer's perspective

The cryptographic literature surveyed above is currently being read against an active deployment push on Ethereum. In December 2025, Luhn [45] proposed *EIP-8105: Universal Enshrined Encrypted Mempool*, a scheme-agnostic encrypted mempool architecture that admits arbitrary decryption providers including threshold encryption, MPC committees, trusted execution environments, delay encryption, and FHE schemes. In February 2026, Elowsson, Ma, and Florentine submitted [25] a minimum viable EIP of LUCID as a candidate headliner for the next Ethereum protocol fork. We focus on Ethereum throughout this paper because the deployment debate there is the most concrete: the Shutterized Gnosis Chain anchors a production baseline, while EIP-8105 and the LUCID headliner are the most advanced in-protocol proposals, and the discussion on the Ethereum Magicians forum makes the requirements that the cryptographic corpus has not yet jointly met explicit. We organise the remainder of this section around the community proposals themselves, the requirements they place on the cryptographic schemes surveyed earlier, and the questions that the cryptographic literature does not yet address.

3.1 Community proposals

Wagner and Ma [64] carry out a parallel survey from the protocol-developer side and arrive at an explicit list of requirements for a threshold-encrypted mempool at the consensus layer, namely silent setup with small public keys, non-interactive decryption, ciphertexts small enough to fit in blob payloads, and CCA2 security. Their requirements track closely with the design axes our literature review organises in Table 1, with silent setup matching the setup-ceremony axis, blob-compatible ciphertext size implied by the decryption type, and CCA-style security a finer reading of the security axis. The sharper consequence Wagner and Ma draw is that no current scheme meets all four requirements together, with [37] the closest match but only CCA1-secure. No threshold-encryption scheme in the corpus can therefore be enshrined into the consensus layer as it stands, and any deployable design must instead accommodate flexibility over which scheme is used. The authors also propose a concrete design, a *hybrid encrypted mempool*, in which the user encrypts a transaction towards both a threshold committee and a self-held key, with either party able to decrypt. Liveness drives this choice: a threshold committee can go offline, and a design that depends on the committee alone risks stalling the mempool. The security of the construction reduces to the user being unable to predict committee availability, which the design enforces through frequent committee rotation and upfront gas charging that disincentivises selective non-revelation.

Elowsson [24] takes the implication of cryptographic non-enshrinement to its logical conclusion in LUCID, an encrypted mempool design that is explicitly *decryption-agnostic*.

The decryptor for any given transaction may be a threshold committee, a single trusted party, or the user themselves, with the choice surfaced as a protocol parameter rather than a built-in. Two design properties of LUCID matter for the discussion below. First, execution of decrypted transactions is delayed to the next block, so the protocol effectively acts on one-slot-stale information. Second, transaction ordering inside a block is committed by the builder while the encrypted bodies are still sealed, a property we will call *blind ordering*. The stated goal of the design is to suppress the toxic part of MEV, namely the front-running and sandwich attacks that exploit visible mempool state, while leaving non-toxic activity such as backrunning arbitrage open. Thiery [62] subsequently proposes *encrypted frame transactions*, a same-slot variant of LUCID that uses order-execute separation to bring decrypted execution into the slot in which the encrypted transaction was committed. LUCID's authors did however show concern about the same slot design on the grounds that it moves key synchronisation onto the critical path, which compresses the propagation window for the rest of the block and introduces builder timing games [59].

3.2 Applicability to Ethereum

Reading Table 1 through the Wagner-Ma filter, only two constructions sit one requirement short of meeting all four together. HSW STE [37] has silent setup, blob-fittable ciphertexts via hybrid encryption, and a modular post-quantum instantiation, but only attains CCA1 rather than CCA2. BEAST-MEV [13] is silent and batched with small ciphertexts and B-IND-CCA security, but is pairing-based and therefore not post-quantum, with security proven in the generic group model. All other constructions in the corpus fail at least two of the four requirements, predominantly on the silent-setup axis. By the Wagner-Ma criteria, these two schemes are the closest credible deployment candidates today.

The same requirements map to concrete numerical bounds when applied to Ethereum's deployment context. Ciphertext size matters because the data must travel through either the execution payload or a data-availability blob, and the twelve-second Ethereum slot is further sub-divided under enshrined proposer-builder separation [21] into windows for proposer publication, attestation, payload reveal, and timeliness-committee voting, leaving only a fraction of the slot for cryptographic work. The HSW STE figures of 1.072 MB (classical elliptic-curve) and 1.431 MB (post-quantum) at one-megabyte plaintexts [37], per ciphertext in a non-batched scheme, sit an order of magnitude beyond the 128 KiB of a single EIP-4844 blob, so a hybrid design carrying them would need multi-blob payloads and proportionally longer propagation windows. Batched constructions such as BEAST-MEV [13] amortise differently: a per-ciphertext partial decryption of 48 bytes sits well inside a blob. The 3.2 seconds per partial decryption that the one-time-setup BTE of [18] reports at a 500-transaction batch parallelises across committee members and so carries a wall-clock cost bounded by single-member compute, but reconstruction is sequential after partial decryptions arrive and itself runs before the plaintext is available: 3.0 seconds for [18], and 129.5 seconds for the silent-setup BEAST-MEV as reported in Section 2.4. For an empirical reference scale, Mazorra, Öz, Schlegel, and Wu [47] report 6.9 million CEX-DEX searcher transactions across 1.9 million blocks containing relevant DEX trades between January 2024 and March 2025, an average of close to four CEX-DEX searcher trades in every block that carries a relevant DEX trade. The megabyte-scale ciphertexts and seconds-long decryption budgets reported in the literature must be evaluated against this scale: a handful of latency-sensitive, MEV-bearing trades in most blocks.

Storage of public-key material and the structured reference string is a second deployment constraint that receives limited attention in the cryptographic literature. The silent-setup

construction of [33] has public keys that grow linearly in the committee size, an artefact that Wagner and Ma [64] explicitly flag as disqualifying. For silent-setup schemes more broadly, the hints each validator must publish for public-key aggregation pose a coupled data-availability and integrity problem: Noyes and Policharla [51] estimate roughly three kilobytes of hints per validator at committee size 128, on the order of three gigabytes across Ethereum's roughly one million validators, all of which participating parties must retrieve and verify. The pairing-based BTE line [18, 13, 3, 55, 12] instead relies on a structured reference string that scales with the maximum batch size, produced by a one-time ceremony and persisted by every encrypting or decrypting node. The hybrid design of [64] addresses the free-option problem by routing the threshold ciphertext through a blob and conditioning the rest of the slot on attester sampling, binding the user to a ciphertext at submission time and keeping the cryptographic state off chain. This is the free option Mazorra et al. [47] identify for enshrined proposer-builder separation, where a party that can reveal or withhold a commitment after observing the market exercises an option whose value grows with volatility and the reveal window. Here the user holding the self-decryption key is the option holder.

Committee rotation is the security primitive on which the hybrid mempool design rests, and only the silent-setup line of constructions supports it without an interactive ceremony. The validator set on Ethereum rotates at every epoch boundary, and any scheme whose setup must be refreshed on rotation accumulates a ceremony cost that scales with the rotation frequency. Under proposer-builder separation [21], where a small set of well-resourced builders builds blocks and validators only attest, the cryptographic burden of partial-decryption verification, blob routing, and reconstruction can be concentrated on builders, who bear it within the payload-reveal window, without harming validator decentralisation.

Wallet and client integration is a third constraint, central to the deployment debate but absent from the cryptographic literature. Every proposal in this section introduces new transaction types, new commitment lifecycles, new key-release timing rules, and new failure modes that wallets and clients must surface to users as concrete UX consequences: longer time to inclusion, less predictable execution quality, complications for MEV refund flows, and a pay-for-envelope fee model in which the user is charged for the declared gas limit regardless of realised consumption, an arrangement that overcharges any transaction with a gas-limit buffer. These end-user costs, rather than the cryptographic ones, are what the Ethereum Magicians discussion of [25] has surfaced as the principal blockers to wallet adoption.

3.3 Shared blind spots

The community proposals agree on two requirements that follow from the realisation that no single cryptographic scheme suffices. The first is *liveness without coordination*, which the hybrid design of [64] achieves through user fallback and which the decryption-agnostic structures of [24, 45] achieve through trustless self-decryption. The second is *encryption agnosticism*, the explicit refusal to hard-code a single cryptographic scheme into the consensus layer so that the deployed scheme can evolve as the literature advances. EIP-8105 [45] formalises this by introducing a key-provider registry as a system contract, allowing any party to register a decryption provider that the protocol then treats uniformly.

A more contested question than what to enshrine is whether to enshrine the primitive at the protocol layer at all. The argument for in-protocol deployment, articulated by the Shutter team [60, 61] and by Ma [46], rests on three premises: (1) that out-of-protocol private mempools and trusted builder relationships have produced a centralisation drift that competition alone cannot unwind, (2) that visibility of pre-confirmation transaction

content is the root cause of the malicious MEV that drives users towards those pipelines, and (3) that only by-default protection at the consensus layer can break the dependency. The counter-position raised by wallet and protocol developers in the Magicians discussion of [25] is that the cryptographic constructions remain experimental, that the implementation burden on client teams is substantial, and that user demand at adoption time is uncertain, all of which favour continued off-protocol experimentation. The cryptographic literature surveyed in this paper is agnostic on this question, since every construction we examined is deployable either in-protocol or as an out-of-protocol service operated by a single committee, and the choice is thus driven by economics and governance rather than by cryptography.

The literature review assembled in this paper addresses the enshrinement question directly. By the Wagner-Ma criteria, no construction in the corpus jointly meets the four requirements: HSW STE sits one CCA gap short, BEAST-MEV one post-quantum gap short, and every other construction we examined fails on at least the silent-setup axis. The criteria were derived from a core protocol developer perspective and intended to capture what a deployable scheme should look like. By the community's own deployment criteria, enshrinement remains premature on cryptographic grounds today. Section 4 reinforces this preliminary conclusion by investigating a third *requirement*: the execution-quality cost that blind ordering imposes on MEV-bearing transactions.

The cryptographic literature treats blind ordering as a privacy feature rather than a fairness concern, since the builder learns nothing about the encrypted bodies while committing the order. In practice the builder retains broad ordering discretion that can be used to position encrypted transactions advantageously once the keys are released, and the community proposals develop different but uniformly partial mitigations, including top-of-block fee auctions in [24] and view-merge constraints on reveal timing in [62]. The corpus surveyed in Section 2 is largely silent on the question, except where the collision-free property removes the user-chosen slot index from the ciphertext. A related blind spot is *binary decryption*, namely that all schemes in the corpus treat the encrypted transaction as an all-or-nothing object that is either entirely sealed or entirely revealed. The frame-transaction structure underlying [62] hints at a richer design space in which some execution parameters can be hidden while others remain public, but no current cryptographic construction supports this directly.

The cost of these design choices is unequally distributed across transaction types. For simple transfers and similar low-contention payloads, blind ordering imposes no execution-quality cost, since their outcome does not depend on fresh and accurate state information about asset prices. The only consequence is an inclusion-latency overhead under the next-slot model, which is a UX cost rather than an economic one. For DeFi transactions and other interactions that touch contentious state, by contrast, outcomes depend on other transactions in the same block and on price movements on centralised exchanges that occur on a millisecond timescale. Here, latency translates directly into worse execution rather than into UX friction alone. Meanwhile, the leading community proposals have embraced blind ordering as a feature to combat toxic MEV. They differ only in how long they delay execution after commitment, ranging from the one-block delay of [24] to the same-slot reveal of [62]. We will now explore the implication this design choice has in forcing execution to take place over stale information.

4 Encrypted Mempools through the execution-quality lens

The cryptographic landscape in Section 2 and the protocol-developer perspective in Section 3 together establish that no current threshold-encrypted mempool construction jointly meets the four convergent properties the community has articulated. This section reinforces that finding from a third direction, by examining what the design choices the community proposals share, namely blind ordering and binary decryption, cost users at the execution layer. *Even a cryptographically perfect threshold-encrypted mempool would impose costs the present proposals do not internalise.*

We organise the remainder of this section around two claims. First, blind ordering and binary decryption together do more than neutralise the front-running and sandwich strategies that motivated MEV mitigation in the first place: they eliminate the same-block positioning auctions that today route MEV value back to users, and degrade the backrun auctions that survive into next-slot execution over stale state. Order-flow auctions, surveyed by Gosselin and Chiplunkar [36], sit on top of the Flashbots Auction architecture [27] and are also delivered by MEV-Blocker and CoW Swap [19] for swap-specific flow. Yang et al. [66] show in an empirical study who in the supply chain captures their value, and Section 4.1 treats the concrete consequences: spam, validator centralisation, and DeFi execution quality. Second, the right design target is not full encryption from broadcast to ordering, but *programmable privacy* in the sense of Benarroch et al. [7], in which an enclave or a secure program can act on encrypted state before plaintext is revealed (Section 4.2).

4.1 The cost of blind ordering

The MEV literature distinguishes between toxic MEV, which extracts value from users without their consent through front-running and sandwich attacks [20], and non-toxic MEV, in which searchers compete to deliver execution improvements such as backrun arbitrage refunds. The two share an informational substrate: transaction content is visible during the ordering window. Blind ordering removes that visibility uniformly. Toxic and non-toxic strategies are suppressed together, since the schemes surveyed in Section 2 treat the encrypted payload as an atomic object that is either entirely sealed or entirely revealed, a property we will call *binary decryption*. The property attaches to the interface the current threshold-mempool schemes expose, not to threshold encryption as a primitive. However, the programmable directions surveyed in Section 4.2 show the primitive admits richer modes. There is no intermediate state in which an auction can run over hidden but operationally usable payload: the same-block family of mechanisms that today route MEV into user rebates is collapsed, and what survives must execute against state at least one slot stale.

4.1.1 Spam, validator centralisation and execution quality

The first concrete cost is *spam* incurred by blind ordering, and it appears in two related dynamics that are usually conflated. On one side, spam appears on-chain as reverting transactions that occupy block space, since builders cannot simulate encrypted payloads at ingress and so cannot filter reverts. On the other side, spam appears at the interface between builders and order-flow auction frontends as a flood of speculative bundles submitted by searchers who, in the absence of plaintext, can only guess at the user's intent.

On chains without an auction to channel searcher competition, that competition manifests as spam, a contemporary instance of the priority gas auction dynamic originally described by Daian et al. [20]: Miller [49] documents OP-Stack rollups where spam bots consume more

than 50% of gas while paying less than 10% of fees, and Solana where MEV bots account for 40% of blockspace. Gogol et al. [35] report that more than 80% of reverted transactions on major Ethereum rollups are swaps, half targeting USDC/WETH pools on Uniswap v3 and v4. The equilibrium analysis of Mazorra et al. [48] brings a matching theoretical statement: under first-come-first-served sequencing with delayed observability of the target, searchers compete by sending redundant transactions, and the number of transactions sent is the natural measure of cost. We revisit this connection between auctions and spam in Section 4.2. Removing expressiveness from searchers does not reduce their activity but channels it into spam.

Selective-disclosure systems sit between full plaintext and full encryption: Passerat-Palmbach and Wadhwa [54] formalise the trade-off on MEV-Share, analysing both a backrunning contract that reconstructs the trade on-chain at a gas premium and a brute-force search over the missing fields that floods the MEV-share node. Under blind ordering, equivalent to the hash-only tier that Flashbots Protect calls *full privacy mode*, in which only the transaction hash is shared [30], neither searcher channel is suppressed: the MEV-Share node has no plaintext to reveal selectively, simulation pre-inclusion is unavailable, and both spam channels are amplified. Garimidi et al. [34] reach the same conclusion from the cryptographic angle, observing that if fee data is itself encrypted, malformed ciphertexts cannot be filtered without an auxiliary zero-knowledge proof of well-formedness whose overhead is substantial.

Figure 1, drawn from a public Dune query on Flashbots Protect, is consistent with this trade-off. Within each of the two Flashbots Protect RPC endpoints (*default*, which routes to BuilderNet, a TEE-attested distributed builder network with approximately 25% block share, and *fast*, which routes to all registered builders), moving from a less informative hint tier to a more informative one cuts the p95 inclusion latency by approximately half: on *default* the hash-only to default-logs step tightens p95 from 55.4 to 27.2 seconds, and on *fast* the default config (hash + default_logs) to full hints (calldata + logs + contract_address + function_selector) step tightens p95 from 39.0 to 18.6 seconds. The median remains at one Ethereum slot (12 seconds) across all tiers, so the gain is entirely in the tail. The evidence is suggestive rather than causal: hint tiers are user-selected, so the latency differences may partly reflect differences in transaction type, fee, urgency, or MEV attractiveness across the users who choose each tier, the percentiles are conditional on inclusion, and whether richer hints also change the probability of landing at all cannot be answered from the public data. Across MEV-Share's lifetime, the same dataset shows roughly 85% of landed flow in the two tiers in which searchers operate effectively blind, either with the transaction hash alone or with the hash plus the fact that a swap occurred in a particular pool. An encrypted-mempool design committed to blind ordering forgoes that headroom entirely, and effectively behaves as the hash-only tier.

The second cost is *validator centralisation*. Today, MEV revenue reaches the validator set through MEV-Boost, which Yang et al. [66] document at the limit of a competitive market: two builders produced more than 85% of blocks at the time of their study, three builders today,[3] and the proposer losses they characterise accrue precisely because builders with privileged order flow need not bid the full extractable value to win. MEV mitigations slated for adoption rely on this builder market as an "MEV oracle" to price the lost revenue, and centralisation distorts that price. Under blind ordering, the entire mechanism that translates the builder market's pricing into validator rewards is suppressed. The competitive surface that remains for validators is ingress speed, which favours operators in a few low-latency

[3] Per the public MEV-Boost dashboard at `https://mevboost.pics`, May 2026.

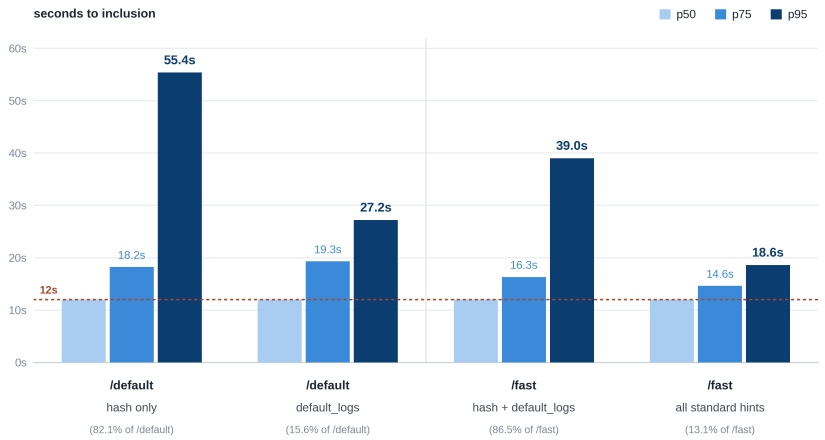

Figure 1 Within-endpoint inclusion latency by hint tier on Flashbots Protect, April 2023 to May 2026. Bars give p50, p75, and p95 of the time from submission to inclusion (seconds) for each (endpoint, hint tier) cell with at least 100k landed transactions, with bracketed values giving the tier's share of landed flow within that endpoint and the dashed line marking one Ethereum slot. The /default endpoint routes to BuilderNet (TEE-attested, ≈25% block share), and /fast routes to all registered builders (≈99% block share). Within each endpoint, the richer hint tier roughly halves p95 while p50 stays on the one-slot reference. Percentiles are conditional on inclusion. Tier shares within an endpoint do not sum to 100% because cells below the 100k-transaction threshold are omitted. Source: Dune query 3397259 (volume-weighted across monthly buckets).

locations and removes the principal subsidy that has kept solo and small-operator staking viable. The intervention designed against builder centralisation thus accelerates centralisation of the validator set instead, the very outcome proposer-builder separation was meant to avoid [15]. Rondelet and Kilbourn [57] raise the same structural point at the level of mempool privacy in general: interventions designed against a single MEV symptom reorganise the surrounding market in ways the cryptographic analysis does not anticipate.

The third cost is *execution quality* for DeFi transactions, and it has been quantified in previous studies. Kim and Akdeniz [44] measure reordering slippage on Uniswap v2 and v3 over September and October 2023 and report $9 million in user losses from intra-block position alone, with 72% of swaps and 92% of swaps over $250,000 negatively affected. Their measurement places most of the slippage beyond the first 10% of block positions, with the sharpest degradation between the 10% and 25% position marks, and they document up to a six-fold difference in execution price between top and bottom of block for the same trade. Execution outcomes for transactions that touch contentious state depend on fresh information about prices on centralised exchanges, which move on millisecond timescales, and on the position of the transaction within the block. Under blind ordering, the builder cannot place a transaction at the position that minimises slippage because the content is not yet visible. Under next-slot execution, an entire block has been finalised by the time the plaintext can be acted upon, and the execution decision is made against price information already at least one slot stale. Centralised execution venues keep compressing, not extending, the gap between price observation and trade settlement, Ethereum itself proposes to halve its slot time [1], and the leading community proposals for encrypted mempools move in the opposite direction.

4.1.2 Concrete impact on execution quality

Let's consider the example of a swap large enough for its execution price to depend on where it lands in the block, the regime Kim and Akdeniz [44] quantify above. Under the encrypted-mempool design of Elowsson in LUCID [24], the swap is committed in slot N and executed in slot $N + 1$, so any backrun that would refund arbitrage value to the user runs at slot $N + 1$ at the earliest, against state already twelve seconds stale to off-chain markets. The same-slot variant of Thiery [62] compresses the gap but still commits the routing path, the pools the swap traverses to minimise price impact, before the encrypted body is revealed. Under a TEE-mediated private mempool such as BuilderNet [28], the same swap is solved against fresh state inside the enclave, the backrun is placed in the same block, and the arbitrage value returns to the user as a rebate. The externality of the delayed price update extends beyond the user: during the twelve-second window in which the Uniswap pool price diverges from off-chain markets, other protocols that read the Uniswap pool as a price oracle execute against the stale price, and lending protocols that depend on those oracle reads may miss liquidations that the fresh price would have triggered. Recently, proprietary-AMM designs have gained traction on other chains, and to be viable on Ethereum L1, they would depend on the same builder access to fresh state to land repricing updates, which blind ordering rules out [50].

The analysis above leads to three conclusions the surveyed encrypted-mempool proposals do not resolve. (1) Blind ordering does not in fact eliminate toxic MEV: sophisticated attackers route around the protection by submitting attack legs across blocks, and wide sandwich bots such as the one identified in [22] succeed precisely because the attack does not need to bundle the frontrun and backrun in the same block, so an encrypted mempool that hides the payload but leaves the underlying execution model untouched would not prevent this class. (2) The proposals impose substantial engineering cost on the protocol and surrounding stack, since builders must be incentivised to include encrypted transactions while out-of-protocol private-mempool revenue remains available, and low end-user adoption would leave wallet and client ecosystems carrying technical debt while functionally similar out-of-protocol solutions deliver the same pre-trade privacy properties with better execution. (3) Decryption itself does not create new state transitions, but the execution of decrypted content does, and the timing gap between key release and execution opens MEV vectors that the proposals acknowledge but do not fully resolve. Taken together, these point to a design space in which auctions, not encryption alone, do the heavy lifting on MEV mitigation, and we turn to that design space in the next subsection.

4.2 The case for programmable privacy

The structural observation that follows from the execution-quality discussion is that running an auction *after* decryption is too late. By the time the plaintext is revealed, the ordering is fixed and the only remaining degree of freedom is the position of subsequent unencrypted transactions in the next block, against stale state. The auctions that today route MEV into user rebates require an intermediate computational mode in which an algorithm can route, simulate, and price against the encrypted payload while keeping the plaintext hidden from its operator. The cryptographic constructions in Section 2 do not provide that mode. They are designed to protect the plaintext until ordering is fixed and then release it, and the design is binary by intention.

The intent-based framing now common in the OFA literature sharpens this observation. The user expresses an intent, namely the outcome they want, rather than the transaction

that realises it, and a solver competes for the right to translate that intent into execution at the best price. The chain-abstraction taxonomy of Chiplunkar and Gosselin [16] decomposes the value at stake into the part that depends only on ordering and the part that depends on observing the intent before settlement. An encrypted mempool can capture only the former, since whoever first sees the revealed transaction can act on it at the next slot; a private mempool captures both, precisely because its operator observes the intent in the clear, which is the trust cost that motivated encryption in the first place. Programmable privacy is the design target that responds to both components together without reintroducing that trust. We use the term *programmable privacy*, in the sense of Benarroch et al. [7], for the alternative design target in which the encrypted state can be acted on by trusted code before plaintext revelation. Two technological substrates support programmable privacy today, with very different trust models.

The first is hardware-based via Trusted Execution Environments (TEEs). The Rollup-Boost system described by the Flashbots team [29] and the BuilderNet deployment in [28] both operate the block builder inside a TEE, with users encrypting their transactions to the enclave's public key. The enclave can simulate transactions against current state, filter reverts, place backruns in the same block, and refund the resulting MEV to the user, all without exposing plaintext to the operator of the hardware. The bottom-of-block searcher architecture of Bakhtin et al. [4] sits in the same family and is best understood as a hardware-backed analogue of selective-disclosure systems such as MEV-Share, in which the searcher receives the equivalent of full-reveal hints over the state changes induced by a partially built block. Here, the hardware boundary restricts the strategy the searcher can run on this state to the bottom-of-block backrun, ruling out frontrunning by construction. The trust assumption across these systems rests on the hardware supply chain rather than on a threshold of honest committee members, a substantively different security model that must be evaluated on its own terms. This shifts trust rather than enshrining new trusted parties: Ethereum already relies on a small set of builders, a concession accepted to keep the validator set decentralised, and the TEE model moves that reliance onto a comparatively disinterested ecosystem, the CPU manufacturers and their supply chain, who have no direct stake in transaction ordering. It carries a real residual cost worth clearly identifying for a fair comparison, since for Intel SGX and TDX a single vendor designs, fabricates, and attests the silicon, the security-critical microcode cannot be publicly audited, and the design tooling and advanced fabrication nodes rest with a handful of firms [43, 42]. Even a breach stays comparatively contained, however: it returns trust to the operator running the enclave, the same reliance on a known builder that private mempools already require.

The second substrate is software-based, namely programmable cryptography in the sense of fully homomorphic encryption and related primitives. Boneh, in a recent talk on the cryptographic frontier for encrypted mempools [8], sketches a long-horizon programme in which selective computation on encrypted transactions replaces the binary release of plaintext. The framework of Garg et al. [32] provides one unifying construction in the threshold setting, in the form of witness encryption for linearly verifiable relations. An otherwise intractable application such as an FHE-based backrun, demonstrated as a feasibility prototype by Passerat-Palmbach [53], would become viable as FHE hardware acceleration matures, with visible efforts in that direction [63, 68]. The performance penalty rules out deployment at Ethereum block-time scale today, but the constructions have improved substantially and the open problems are engineering blockers rather than fundamental.

The implication for the present deployment debate is that enshrining a binary-decryption threshold scheme today fixes the wrong design point. Even setting aside the cryptographic

gap identified by Wagner and Ma [64], the design choice closes off the programmable-privacy direction without making the trade-off explicit. A more cautious position is to keep experiments running off-protocol, both on the TEE substrate where programmable privacy is already deployable, and on the cryptographic substrate where the building blocks are maturing, before fixing the interface at the consensus layer. The overall direction towards encrypted mempools is commendable, but it requires new designs that fully embrace the economic incentives that have proved pivotal in driving adoption in the space.

5 Open challenges

The literature review and the discussion of execution quality both point to a set of open problems that the cryptographic and protocol-development communities will need to address jointly. We close the paper by identifying four: post-quantum convergence, the crypto-economic gap, network-layer metadata leakage, and decryption-aggregation cost.

The first is *post-quantum security under the convergent design properties.* Table 1 contains one construction with a concrete post-quantum instantiation, the lattice-based scheme of Boneh et al. [10] in the batched-decryption line under a one-time DKG, and one modular blueprint instantiable post-quantum, the silent construction of Hall-Andersen et al. [37], which decrypts one ciphertext at a time rather than batching a block and reports megabyte-scale ciphertexts at one-megabyte plaintexts. The combination of silent setup, batched decryption, collision-free encoding, and post-quantum security under standard assumptions does not exist in the corpus to the best of our knowledge. The challenge is structural rather than incremental. The pairing-based BTE line owes its byte-scale ciphertexts and batch-independent decryption communication to algebraic properties of bilinear groups, and a post-quantum replacement would have to either reproduce those properties under lattice or isogeny assumptions or abandon one of the convergent properties. Designing a non-interactive thresholding protocol for post-quantum batched decryption remains an explicit open problem flagged by the authors of [10].

The second open problem is the gap between cryptographic and economic analysis this paper argues for. Rondelet and Kilbourn [57] argue that the impact of private mempools on markets and incentives renders an analysis omitting the economic lens incomplete. The discussion above instantiates that critique against the proposals on the table, but the broader question of how to design encrypted-mempool primitives that internalise execution quality as a first-class objective is open. None of the constructions in Section 2 models the economic value of the plaintext as part of the security definition, and the security games are written purely in terms of what stays encrypted. A design blueprint that takes the economic lens as a constraint rather than as a downstream evaluation is yet to be articulated.

The third open problem is *metadata leakage at the network layer.* Cryptographic mempools hide the transaction payload but not the network-layer metadata that wraps it, including the IP address of the broadcaster, the timing of the broadcast, the size of the ciphertext, and the position of the ciphertext within a batch. Kavousi et al. [41] formalise the problem and propose BlindPerm, a protocol-layer mechanism that piggybacks on an encrypted mempool to shuffle transactions before execution without an external randomness beacon, at a small constant-factor overhead. BlindPerm addresses the in-protocol permutation question, which is orthogonal to the broader network-layer problem of hiding the broadcaster's identity. The anonymous-broadcast direction targets the latter: the ZipNet construction of Rosenberg et al. [58] uses trusted execution environments for liveness and classical cryptography for anonymity, and the Panetière design of [31] pursues the same TEE-assisted anonymous-

broadcast direction with guaranteed output delivery, remaining live with up to half of the servers arbitrarily corrupt. Integrating these network-layer constructions with the threshold-encrypted mempool corpus is an open architectural problem.

The fourth open problem is the *decryption-aggregation cost in epochless schemes*. Epoch-bound schemes make the user commit each ciphertext to a target block, but the user cannot reliably predict the landing slot, so a meaningful fraction miss their target and become undecryptable, and the slot-collision attack of [14] recreates the censorship vector encryption was meant to remove. Epochless schemes shift the work from the user to the committee, which aggregates partial decryptions across a batch on demand. The key-homomorphic puncturable PRF construction of Bormet et al. [14] requires $O(B^2)$ pairings to aggregate at batch size B. The FFT-in-the-exponent decryption of Agarwal et al. [2] first brought the cost to $O(B \log B)$, in the weighted setting, and the partial-fraction technique of Boneh et al. [12] together with the punctured-powers constructions of [55, 3] reach the same quasi-linear cost in the collision-free setting. The most recent numbers, from BTX [3], put end-to-end decryption at $B = 512$ at roughly 600 ms on a single core against 1,200 ms for the partial-fraction baseline of [12], both on BTX's AVX-512 C++ codebase; Policharla's earlier evaluation (Section 2.5) is about twice as large with the same ordering. Viable batch sizes today therefore sit in the few hundreds. Finding an epochless BTE scheme with $O(B)$ decryption-aggregation cost is an explicit open problem, necessary to lift the batch-size ceiling toward L2 throughput targets of thousands of transactions per slot. The post-quantum setting compounds the challenge, since the partial-fraction and FFT techniques relied on by current solutions exploit pairing structure that does not transfer to lattice constructions.

6 Conclusion

The convergent properties our taxonomy identifies, silent setup, batched decryption, and epochless collision-free encoding, describe a design point optimised for a binary-decryption interface that this paper argues is the wrong target. The cryptographic gaps and the economic costs compound rather than conflict: closing the four-property gap leaves blind ordering in place, and a TEE-based programmable-privacy alternative captures the auctions only by shifting trust from the cryptographic committee to the hardware supply chain. The four open problems we identify sit ahead of any deployable design, so the prudent path is off-protocol experimentation: on the TEE substrate where programmable privacy is deployable today, and on the cryptographic substrate where the building blocks are maturing. Premature enshrinement would cost more than a privacy primitive that misses its target and fresh technical debt for wallet and client teams. Above all, it would disrupt the same-block auctions that today route value back to users.

References

1 Ben Adams, Dankrad Feist, Maria Inês Silva, and Paul Harris. EIP-7782: Reduce block latency. Ethereum Improvement Proposals, 2024. Case post: https://ethereum-magicians.org/t/eip-7782-the-case-for-2x-shorter-slot-times-in-glamsterdam/24616. URL: https://eips.ethereum.org/EIPS/eip-7782.

2 Amit Agarwal, Kushal Babel, Sourav Das, Babak Poorebrahim Gilkalaye, Arup Mondal, Benny Pinkas, Peter Rindal, and Aayush Yadav. Weighted batched threshold encryption with applications to mempool privacy. Cryptology ePrint Archive, Paper 2025/2115, November 2025. To appear, IEEE S&P 2026. URL: https://eprint.iacr.org/2025/2115.

3 Amit Agarwal, Sourav Das, Babak Poorebrahim Gilkalaye, Peter Rindal, and Victor Shoup. BTX: Simple and efficient batch threshold encryption. Cryptology ePrint Archive, Paper 2026/754, 2026. URL: https://eprint.iacr.org/2026/754.

4 Artyom Bakhtin, Frieder Erdmann, Angela Lu, Moe Mahhouk, and Mateusz Morusiewicz. Searching in TDX. Flashbots Collective forum, September 2024. Describes the "bob" (bottom-of-block) construction. URL: https://collective.flashbots.net/t/searching-in-tdx/3902.

5 Joseph Bebel and Dev Ojha. Ferveo: Threshold decryption for mempool privacy in BFT networks. Cryptology ePrint Archive, Paper 2022/898, 2022. URL: https://eprint.iacr.org/2022/898.

6 Amos Beimel. Secret-sharing schemes: A survey. In *Coding and Cryptology (IWCC 2011)*, volume 6639 of *Lecture Notes in Computer Science*, pages 11–46. Springer, 2011. doi:10.1007/978-3-642-20901-7_2.

7 Daniel Benarroch, Bryan Gillespie, Ying Tong Lai, and Andrew Miller. SoK: Programmable privacy in distributed systems. Cryptology ePrint Archive, Paper 2024/982, 2024. URL: https://eprint.iacr.org/2024/982.

8 Dan Boneh. Cryptography for encrypted mempools. a16z crypto research talks, YouTube, November 2025. URL: https://www.youtube.com/watch?v=0NNDX63lySQ.

9 Dan Boneh, Craig Gentry, and Brent Waters. Collusion resistant broadcast encryption with short ciphertexts and private keys. In *Advances in Cryptology – CRYPTO 2005*, volume 3621 of *Lecture Notes in Computer Science*, pages 258–275. Springer, 2005. doi:10.1007/11535218_16.

10 Dan Boneh, Evan Laufer, and Ertem Nusret Tas. Batch decryption without epochs and its application to encrypted mempools. Cryptology ePrint Archive, Paper 2025/1254, 2025. URL: https://eprint.iacr.org/2025/1254.

11 Dan Boneh, Ben Lynn, and Hovav Shacham. Short signatures from the Weil pairing. *Journal of Cryptology*, 17(4):297–319, 2004. Preliminary version in ASIACRYPT 2001, LNCS 2248, pp. 514–532. doi:10.1007/s00145-004-0314-9.

12 Dan Boneh, Rohit Nema, Arnab Roy, and Ertem Nusret Tas. Efficient batch threshold encryption using partial fraction techniques. Cryptology ePrint Archive, Paper 2026/674, 2026. URL: https://eprint.iacr.org/2026/674.

13 Jan Bormet, Arka Rai Choudhuri, Sebastian Faust, Sanjam Garg, Hussien Othman, Guru-Vamsi Policharla, Ziyan Qu, and Mingyuan Wang. BEAST-MEV: Batched threshold encryption with silent setup for MEV prevention. Cryptology ePrint Archive, Paper 2025/1419, 2025. URL: https://eprint.iacr.org/2025/1419.

14 Jan Bormet, Sebastian Faust, Hussien Othman, and Ziyan Qu. BEAT-MEV: Epochless approach to batched threshold encryption for MEV prevention, August 2025. URL: https://www.usenix.org/conference/usenixsecurity25/presentation/bormet.

15 Vitalik Buterin. Proposer/block builder separation-friendly fee market designs. ethresear.ch, June 2021. URL: https://ethresear.ch/t/proposer-block-builder-separation-friendly-fee-market-designs/9725.

16 Ankit Chiplunkar and Stephane Gosselin. Introducing the CAKE framework. Frontier Research, February 2024. URL: https://frontier.tech/the-cake-framework.

17 Arka Rai Choudhuri, Sanjam Garg, Julien Piet, and Guru-Vamsi Policharla. Mempool privacy via batched threshold encryption: Attacks and defenses. In *33rd USENIX Security Symposium (USENIX Security 24)*, pages 3513–3529, Philadelphia, PA, 2024. USENIX Association. URL: https://www.usenix.org/conference/usenixsecurity24/presentation/choudhuri.

18 Arka Rai Choudhuri, Sanjam Garg, Guru-Vamsi Policharla, and Mingyuan Wang. Practical mempool privacy via one-time setup batched threshold encryption. Cryptology ePrint Archive, Paper 2024/1516, 2024. To appear, USENIX Security 2025. URL: https://eprint.iacr.org/2024/1516.

19 CoW DAO. MEV Blocker. cow.fi. Accessed 2026. URL: https://cow.fi/mev-blocker.

20 Philip Daian, Steven Goldfeder, Tyler Kell, Yunqi Li, Xueyuan Zhao, Iddo Bentov, Lorenz Breidenbach, and Ari Juels. Flash Boys 2.0: Frontrunning in decentralized exchanges, miner extractable value, and consensus instability. In *2020 IEEE Symposium on Security and Privacy (SP)*, pages 910–927, 2020. doi:10.1109/SP40000.2020.00040.

21 Francesco D'Amato, Nico Flaig, Barnabé Monnot, Michael Neuder, Potuz, Justin Traglia, and Terence Tsao. EIP-7732: Enshrined proposer-builder separation. Ethereum Improvement Proposals (draft), June 2024. Created 28 June 2024; status: draft. Accessed 2026-05-22. URL: https://eips.ethereum.org/EIPS/eip-7732.

22 Dataalways. Twitter thread on the 0x841 wide-sandwich bot. Twitter/X, 2024. Pseudonymous author; archival via web.archive.org recommended before final submission. URL: https://x.com/dataalways/status/2047693168791928906.

23 Stefan Dziembowski, Sebastian Faust, and Jannik Luhn. Shutter network: Private transactions from threshold cryptography. Cryptology ePrint Archive, Paper 2024/1981, 2024. URL: https://eprint.iacr.org/2024/1981.

24 Anders Elowsson. LUCID: Encrypted mempool with distributed payload propagation. Ethereum Research forum (ethresear.ch), February 2026. Posted 5 February 2026. Accessed 2026-05-22. URL: https://ethresear.ch/t/lucid-encrypted-mempool-with-distributed-payload-propagation/24042.

25 Anders Elowsson, Julian Ma, and Justin Florentine. Hegota headliner: LUCID encrypted mempool. Ethereum Magicians forum (ethereum-magicians.org), February 2026. Posted 5 February 2026. Accessed 2026-05-22. URL: https://ethereum-magicians.org/t/hegota-headliner-lucid-encrypted-mempool/27658.

26 Rex Fernando, Guru-Vamsi Policharla, Andrei Tonkikh, and Zhuolun Xiang. TrX: Encrypted mempools in high performance BFT protocols. Cryptology ePrint Archive, Paper 2025/2032, 2025. URL: https://eprint.iacr.org/2025/2032.

27 Flashbots. Flashbots Auction overview. Online documentation. Accessed 2026. URL: https://docs.flashbots.net/flashbots-auction/overview.

28 Flashbots. Introducing BuilderNet. Flashbots / BuilderNet, November 2024. Decentralised, TEE-based collaborative block-building network for Ethereum L1. Migration write-up at https://writings.flashbots.net/migrating-to-buildernet. URL: https://buildernet.org/blog/introducing-buildernet.

29 Flashbots. Introducing Rollup-Boost: Launching on Unichain. Flashbots Writings, October 2024. URL: https://writings.flashbots.net/introducing-rollup-boost.

30 Flashbots. Flashbots protect settings guide. Flashbots documentation, 2025. Accessed 2026-08-08. URL: https://docs.flashbots.net/flashbots-protect/settings-guide.

31 Nils Fleischhacker, Mateusz Morusiewicz, and Mark Simkin. Panetière: Efficient anonymous broadcast. Draft, Flashbots, July 2026. Formerly known as Flashnet. Accessed 2026-08-07. URL: https://net.flashbots.net/panetiere-draft.pdf.

32 Sanjam Garg, Mohammad Hajiabadi, Dimitris Kolonelos, Abhiram Kothapalli, and Guru-Vamsi Policharla. A framework for witness encryption from linearly verifiable SNARKs and applications. Cryptology ePrint Archive, Paper 2025/1364, 2025. CRYPTO 2025. URL: https://eprint.iacr.org/2025/1364.

33 Sanjam Garg, Dimitris Kolonelos, Guru-Vamsi Policharla, and Mingyuan Wang. Threshold encryption with silent setup. In Leonid Reyzin and Douglas Stebila, editors, *Advances in Cryptology – CRYPTO 2024, Part VII*, volume 14926 of *Lecture Notes in Computer Science*, pages 352–386, Cham, 2024. Springer. Also available as Cryptology ePrint Archive Paper 2024/263, https://eprint.iacr.org/2024/263. doi:10.1007/978-3-031-68394-7_12.

34 Pranav Garimidi, Joseph Bonneau, and Lioba Heimbach. On the limits of encrypted mempools. a16z crypto, 2025. URL: https://a16zcrypto.com/posts/article/limits-encrypted-mempools/.

35 Krzysztof M. Gogol, Manvir Schneider, and Claudio J. Tessone. When priority fails: Revert-Based MEV on fast-finality rollups, 2025. URL: https://arxiv.org/abs/2506.01462, arXiv:2506.01462.

36 Stephane Gosselin and Ankit Chiplunkar. The orderflow auction design space. Frontier Research, March 2023. URL: https://frontier.tech/the-orderflow-auction-design-space.

37 Mathias Hall-Andersen, Mark Simkin, and Benedikt Wagner. Silent threshold encryption with one-shot adaptive security. Cryptology ePrint Archive, Paper 2025/1384, 2025. URL: https://eprint.iacr.org/2025/1384.

38 Amir Herzberg, Stanisław Jarecki, Hugo Krawczyk, and Moti Yung. Proactive secret sharing or: How to cope with perpetual leakage. In Don Coppersmith, editor, *Advances in Cryptology – CRYPTO 1995*, volume 963 of *Lecture Notes in Computer Science*, pages 339–352. Springer, 1995. Cite key includes "jakobsson"; PSS 1995 author list is HJKY only. If you intended the related Herzberg-Jakobsson-Jarecki-Krawczyk-Yung paper "Proactive Public Key and Signature Systems" (ACM CCS 1997), swap this entry. doi:10.1007/3-540-44750-4_27.

39 Charanjit S. Jutla, Rohit Nema, and Arnab Roy. Partial fraction techniques for cryptography. In *Advances in Cryptology – EUROCRYPT 2026*, Lecture Notes in Computer Science, pages 423–453. Springer, 2026. Cryptology ePrint Archive, Paper 2025/2081.

40 Aniket Kate, Gregory M. Zaverucha, and Ian Goldberg. Constant-size commitments to polynomials and their applications. In Masayuki Abe, editor, *Advances in Cryptology – ASIACRYPT 2010*, volume 6477 of *Lecture Notes in Computer Science*, pages 177–194, Berlin, Heidelberg, 2010. Springer. doi:10.1007/978-3-642-17373-8_11.

41 Alireza Kavousi, Duc V. Le, Philipp Jovanovic, and George Danezis. BlindPerm: Efficient MEV mitigation with an encrypted mempool and permutation. In *29th International Conference on Principles of Distributed Systems (OPODIS 2025)*, volume 361 of *Leibniz International Proceedings in Informatics (LIPIcs)*, pages 36:1–36:21. Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2025. doi:10.4230/LIPIcs.OPODIS.2025.36.

42 Quintus Kilbourn, Sylvain Bellemare, Andrew Huang, and Michael Gao. ZTEE: Trustless supply chains. Flashbots Writings, November 2024. Accessed 2026-08-08. URL: https://writings.flashbots.net/ZTEE2-Supply-Chains.

43 Quintus Kilbourn, Sylvain Bellemare, Jonathan Passerat-Palmbach, and Andrew Miller. Zero trust execution environments. Flashbots Writings, October 2024. Accessed 2026-08-08. URL: https://writings.flashbots.net/ZTEE.

44 Evan Kim and Murat Akdeniz. Trading in the dark: Measuring reordering slippage in MEV. Primev, in collaboration with Frontier Research, January 2024. URL: https://frontier.tech/measuring-reordering-slippage-in-mev.

45 Jannik Luhn. EIP-8105: Universal enshrined encrypted mempool. Ethereum Improvement Proposals (draft), December 2025. Created 17 December 2025; status: draft. Accessed 2026-05-22. URL: https://eips.ethereum.org/EIPS/eip-8105.

46 Julian Ma. Why Ethereum needs an encrypted mempool. Julian's Research (Substack), March 2026. Posted 12 March 2026. Accessed 2026-05-22. URL: https://julianma.substack.com/p/why-ethereum-needs-an-encrypted-mempool.

47 Bruno Mazorra, Burak Öz, Christoph Schlegel, and Fei Wu. The free option problem of ePBS, 2025. Submitted 29 September 2025; last revised 8 January 2026 (v3). URL: https://arxiv.org/abs/2509.24849, arXiv:2509.24849.

48 Bruno Mazorra, Christoph Schlegel, and Akaki Mamageishvili. Timing games: Probabilistic backrunning and spam, 2026. URL: https://arxiv.org/abs/2602.22032, arXiv:2602.22032.

49 Robert Miller. MEV and the limits of scaling. Flashbots Writings, June 2025. URL: https://writings.flashbots.net/mev-and-the-limits-of-scaling.

50 Mike Neuder and Maryam Bahrani. Proprietary AMMs and Ethereum. Ethereum Research forum (ethresear.ch), July 2026. Accessed 2026-08-08. URL: https://ethresear.ch/t/proprietary-amms-and-ethereum/25543.

51 Charlie Noyes and Guru Vamsi Policharla. How to remove the relay. Paradigm Research, October 2024. Posted 8 October 2024. Accessed 2026-05-22. URL: https://www.paradigm.xyz/2024/10/removing-the-relays.

52 Vabuk Pahari and Andrea Canidio. How exclusive are Ethereum transactions? Evidence from non-winning blocks, 2025. Submitted 19 September 2025. URL: https://arxiv.org/abs/2509.16052, arXiv:2509.16052.

53 Jonathan Passerat-Palmbach. Leveraging homomorphic encryption for maximally extractable value (MEV) mitigation: Enabling blind arbitrage on decentralised exchanges. In *Financial Cryptography and Data Security 2025*, 2025. Paper 238 in preproceedings. URL: https://fc25.ifca.ai/preproceedings/238.pdf.

54 Jonathan Passerat-Palmbach and Sarisht Wadhwa. Differentially private aggregate hints in MEV-Share, 2025. URL: https://arxiv.org/abs/2508.14284, arXiv:2508.14284.

55 Guru-Vamsi Policharla. A simple batched threshold encryption scheme. Cryptology ePrint Archive, Paper 2026/760, 2026. URL: https://eprint.iacr.org/2026/760.

56 Leonid Reyzin, Adam D. Smith, and Sophia Yakoubov. Turning HATE into LOVE: Compact homomorphic ad hoc threshold encryption for scalable MPC. In Shlomi Dolev, Oded Margalit, Benny Pinkas, and Alexander Schwarzmann, editors, *Cyber Security Cryptography and Machine Learning (CSCML 2021)*, volume 12716 of *Lecture Notes in Computer Science*, pages 361–378, Cham, 2021. Springer. Also available as Cryptology ePrint Archive Paper 2018/997, https://eprint.iacr.org/2018/997. doi:10.1007/978-3-030-78086-9_27.

57 Antoine Rondelet and Quintus Kilbourn. Mempool privacy: An economic perspective, 2023. URL: https://arxiv.org/abs/2307.10878, arXiv:2307.10878.

58 Michael Rosenberg, Maurice Shih, Zhenyu Zhao, Rui Wang, Ian Miers, and Fan Zhang. ZIPNet: Low-bandwidth anonymous broadcast from (dis)Trusted Execution Environments. Cryptology ePrint Archive, Paper 2024/1227, 2024. Published version: Proceedings on Privacy Enhancing Technologies, 2025(2):211–225, https://doi.org/10.56553/popets-2025-0058. URL: https://eprint.iacr.org/2024/1227.

59 Caspar Schwarz-Schilling, Fahad Saleh, Thomas Thiery, Jennifer Pan, Nihar Shah, and Barnabé Monnot. Time is money: Strategic timing games in proof-of-stake protocols. In *5th Conference on Advances in Financial Technologies (AFT 2023)*, volume 282 of *LIPIcs*, pages 30:1–30:17, 2023. doi:10.4230/LIPIcs.AFT.2023.30.

60 Shutter Network. Introducing EIP-8105 universal enshrined encrypted mempool. Shutter Blog, December 2025. Posted 17 December 2025. Accessed 2026-05-22. URL: https://blog.shutter.network/introducing-the-universal-enshrined-encrypted-mempool-eip/.

61 Shutter Network and Tatu. Why encryption is the only real protection against MEV. Shutter Blog, May 2025. Posted 15 May 2025. Accessed 2026-05-22. URL: https://blog.shutter.network/why-encryption-is-the-only-real-protection-against-mev/.

62 Thomas Thiery. Encrypted frame transactions. Ethereum Research forum (ethresear.ch), March 2026. Posted 18 March 2026. Accessed 2026-05-22. URL: https://ethresear.ch/t/encrypted-frame-transactions/24440.

63 Michiel Van Beirendonck, Jan-Pieter D'Anvers, Furkan Turan, and Ingrid Verbauwhede. FPT: A fixed-point accelerator for torus fully homomorphic encryption. *arXiv preprint*, 2022. Also published at CCS 2023. URL: https://arxiv.org/abs/2211.13696, arXiv:2211.13696.

64 Benedikt Wagner and Julian Ma. Hybrid encrypted mempools. Ethereum Research forum (ethresear.ch), October 2025. Posted 27 October 2025. Accessed 2026-05-22. URL: `https://ethresear.ch/t/hybrid-encrypted-mempools/23360`.

65 Brent Waters and David J. Wu. Silent threshold cryptography from pairings: Expressive policies in the plain model. In *Advances in Cryptology – EUROCRYPT 2026, Part V*, volume 16545 of *Lecture Notes in Computer Science*, pages 517–547. Springer, 2026. Cryptology ePrint Archive, Paper 2025/1547. `doi:10.1007/978-3-032-25330-9_18`.

66 Sen Yang, Kartik Nayak, and Fan Zhang. Decentralization of Ethereum's builder market, 2024. URL: `https://arxiv.org/abs/2405.01329`, `arXiv:2405.01329`.

67 Sen Yang, Burak Öz, Fei Wu, and Fan Zhang. Geographical centralization resilience in Ethereum's block-building paradigms. *Proceedings of the ACM on Measurement and Analysis of Computing Systems*, 10(2):39:1–39:42, 2026. Presented at ACM SIGMETRICS 2026. Preprint: arXiv:2509.21475. `doi:10.1145/3805637`.

68 Zama. A systemVerilog implementation of the Homomorphic Processing Unit (HPU) targeting AMD Alveo V80 FPGA board, 2025. `https://github.com/zama-ai/hpu_fpga`.

A Deployment-budget summary

Table 2 collects the numerical constraints and per-construction figures cited across Section 2 and Section 3, so that the deployment argument can be inspected as a single budget. Slot and storage figures are protocol-fixed. The validator-set size reflects the May 2026 beacon-chain census. Per-construction figures are at $B = 512$ where applicable, on a single core unless noted, and silent constructions report per-ciphertext rather than per-batch figures since they have no batched interface.

Table 2 Deployment-budget summary at Ethereum scale, drawn from the figures cited in the body of the paper.

Quantity	Value	Source
Slot and storage budget		
Ethereum slot duration	12 s	[21]
Single EIP-4844 blob capacity	128 KiB	EIP-4844
Validator-set size	≈1 million	beacon chain
Empirical block scale		
CEX-DEX searcher transactions per block (mean)	≈3.6	[47]
Silent-setup hint storage		
Hints per validator (committee size 128)	3 KB	[51]
Total hint accumulation across the validator set	≈3 GB	[51]
Per-construction ciphertext size		
HSW STE classical (per ciphertext, non-batched)	1.072 MB	[37]
HSW STE post-quantum (per ciphertext, non-batched)	1.431 MB	[37]
BEAST-MEV per-ciphertext partial decryption (batched)	48 B	[13]
CGPW one-time DKG per partial decryption (batched)	48 B	[18]
Policharla 26 / BTX shortest ciphertext (batched)	≈700 B	[55, 3]
Decryption latency at $B = 512$		
CGPW one-time DKG, partial + reconstruction ($B = 500$)	3.2 + 3.0 s	[18]
BEAST-MEV silent batched reconstruction	129.5 s	[13]
TrX end-to-end overhead (50-node WAN)	27 ms (≈14%)	[26]
Policharla 26 (independent codebase)	2.80 s	[55]
BNRT FFT-optimised partial-fraction (AVX-512 codebase)	≈1.2 s	[12, 3]
BTX (AVX-512 codebase)	≈0.6 s	[3]

B Figure 1 data and methodology

Figure 1 is drawn from the public Dune query 3397259 (`https://dune.com/queries/3397259`) over Flashbots Protect transactions between April 2023 and May 2026. The query buckets landed transactions by month, RPC endpoint, and hint configuration, and records the p50, p75, and p95 of the delay from submission to on-chain inclusion. Per-tier percentiles in the figure are volume-weighted averages across the monthly buckets, and (endpoint, hint tier) cells with fewer than 100,000 landed transactions over the window are omitted, which is why tier shares within an endpoint do not sum to 100%. Percentiles are conditional on inclusion: the public data does not expose submissions that never landed, so land-rate differences across tiers, and any selection induced by users choosing their own hint tier, are not addressable from this dataset.