IEEE Access
Multidisciplinary | Rapid Review | Open Access Journal

Received 24 September 2024, accepted 29 November 2024, date of publication 9 December 2024,
date of current version 17 December 2024.

Digital Object Identifier 10.1109/ACCESS.2024.3514375

SURVEY

Maximum Extractable Value (MEV) Mitigation Approaches in Ethereum and Layer-2 Chains: A Comprehensive Survey

ZEINAB ALIPANAHLOO[1], ABDELHAKIM SENHAJI HAFID[2], AND KAIWEN ZHANG[1], (Member, IEEE)
[1]Department of Software and IT Engineering, École de Technologie Supérieure, Montreal, QC H3C 1K3, Canada
[2]Department of Computer Science and Operational Research, University of Montreal, Montreal, QC H3C 3J7, Canada

Corresponding author: Zeinab Alipanahloo (zeinab.alipanahloo.1@ens.etsmtl.ca)

ABSTRACT Maximal Extractable Value (MEV) represents a pivotal challenge within the Ethereum ecosystem; it impacts the fairness, security, and efficiency of both Layer 1 (L1) and Layer 2 (L2) networks. MEV arises when miners or validators manipulate transaction ordering (e.g., front-running) to extract additional value, often at the expense of other network participants. This not only affects user experience by introducing unpredictability and potential financial losses but also threatens the underlying principles of decentralization and trust. Given the growing complexity of blockchain applications, particularly with the increase of Decentralized Finance (DeFi) protocols, it is crucial to address the issue and reduce the impact of MEV. This paper presents a comprehensive survey of MEV mitigation techniques as applied to both Ethereum's L1 and various L2 solutions. We provide a novel categorization of mitigation strategies. We also describe the challenges, ranging from transaction sequencing and cryptographic methods to reconfiguring decentralized applications (DApps) to reduce front-running opportunities. We investigate their effectiveness, implementation challenges, and impact on network performance. By synthesizing current research, real-world applications, and emerging trends, this paper aims to provide a detailed roadmap for researchers, developers, and policymakers to understand and combat MEV in an evolving blockchain landscape.

INDEX TERMS Blockchain technology, DeFi protocols, Ethereum, fair ordering, MEV, privacy-preserving methods.

I. INTRODUCTION

The landscape of blockchain technology, particularly Ethereum and its Layer 2 (L2) chains, has been significantly impacted by MEV (Miner Extractable Value) extraction strategies. These strategies, which include DEX (Decentralized Exchange) arbitrage, sandwiching, and liquidations, have emerged as profitable opportunities for miners and other participants in the network. However, these strategies also pose potential risks to the network's stability and security, leading to the development of various mitigation approaches. Blockchains have historically achieved data integrity and immutability through a network of nodes reaching consensus on the validity of transactions recorded in the distributed ledger. However, consensus protocols adopted in many blockchains do not enforce rules on the ordering of transactions produced by block producers. Miners or validators can exploit their capability to preview the network's upcoming state and manipulate transactions by reordering, including, or excluding them to generate profits. This profit-seeking behavior is known as MEV extraction; it can lead to significant issues for users, applications, and overall network robustness. This becomes significant with the rapid expansion of DeFi (Decentralized Finance) projects, leading to a notable increase in MEV extraction within the Ethereum network [1].

One prominent approach to extract MEV is through front-running attacks; this involves submitting a transaction with

The associate editor coordinating the review of this manuscript and approving it for publication was Massimo Cafaro.

higher transaction fees to surpass a pending transaction in the execution queue [2]. Front-running attacks form a dominant method to extract MEV; they enable individuals to leverage the transaction prioritization mechanism for their own financial gain. It is important to note that transaction ordering manipulation also occurs on off-chain or layer 2 (L2) networks, where single or multiple sequencer nodes are responsible for determining the final transaction order. L2 solutions are designed to improve the scalability and performance of blockchain networks by processing transactions off-chain [3], [4], [5].

Rollups are a specific type of L2 solutions with a single sequencer. The sequencer, under the control of the rollup operator, is a central authority responsible for the ordering and execution of transactions. Rollup chains employ a single sequencer to ensure deterministic transaction ordering, simplify system architecture, and enhance efficiency and security, although it may have certain drawbacks. Rollups can also have a network of sequencers, though operating their own network can be costly for the rollup chain [6]. A recent solution to this problem is the development of shared sequencer networks, which allow multiple rollups to utilize a common network of sequencers.

This paper aims to survey the current state of MEV mitigation approaches on Ethereum and L2 chains, with a focus on the strategies employed to address the challenges posed by MEV. By grouping these mitigation approaches into specific strategies and analyzing them, we aim to provide a comprehensive understanding of the landscape of MEV mitigation. To the best of our knowledge, this work is novel in its systematic categorization and detailed analysis of MEV mitigation strategies.

The rest of this paper is organized as follows. Section II provides a brief overview of existing Surveys and reviews on MEV mitigation methods. Section III presents background concepts related to MEV extraction issue. Section IV introduces a novel taxonomy of different MEV mitigation strategies. Sections V, VI, VII, and VIII review existing contributions for each mitigation strategy. Section IX compares all MEV mitigation strategies. Finally, Section X concludes the paper.

II. RELATED WORK

Previous research on MEV can be divided into two main categories. The first category involves the detection and quantification of MEV by analyzing recorded transactions or monitoring mempool activities, a topic extensively covered in the existing literature. The second category is concerned with strategies to prevent or mitigate MEV extraction within a network (e.g. [7], [8], [9], [10]). In this paper, we focus on the second category, specifically on MEV extraction mitigation and prevention methods on Ethereum and L2 chains.

Several studies have examined and evaluated various methods for mitigating MEV. Xu et al. [11] focus on Automated Market Maker (AMM)-based Decentralized Exchange (DEX) platforms. They assess the mechanics of various protocols (e.g., Uniswap [12], Curve [13], Balancer [14], and DODO [15]). They conclude with a comprehensive review of the AMM-based DEX platforms, focusing specifically on their design and associated vulnerabilities including front-running. The authors propose a taxonomy covering the economic risks and security issues associated with AMMs. The paper classifies front-running as a significant economic risk for AMM protocols within its risk taxonomy; it evaluates the impact of these attacks on the economic dynamics of DEXs. It also proposes potential mitigation methods for front-running, such as enforcing sequencing rules and concealing transaction details.

Alam et al. [16] explore the central role of DeFi in the financial ecosystem, particularly focusing on security issues. The authors concentrate on front-running attacks, with less emphasis on mitigation strategies.

Zhang et al. [17] focus on evaluating methods for detecting front-running vulnerabilities in smart contracts. The authors propose an algorithm to detect real-world front-running attacks in the Ethereum transaction history; the algorithm outperforms baseline methods. They introduce an approach to automatically collect vulnerable smart contracts that have been exploited in historical attacks. In addition, they put forward a benchmark of real-world front-running attacks, including the associated vulnerable smart contract code. The paper conducts an empirical evaluation of seven state-of-the-art front-running vulnerability detection tools using the constructed benchmark.

Eskandari et al. [2] propose a classification framework to understand different types of front-running attacks. They categorize these attacks into three main types: displacement, insertion, and suppression. The authors analyze various mitigation techniques (transaction sequencing, confidentiality, and smart contract design practices), evaluating their effectiveness and limitations. Additionally, the paper presents real-world examples of front-running attacks and their impact on blockchain systems and applications.

The authors in [18] classify and examine various schemes for mitigating transaction reordering manipulations. However, they do not cover recent advances and their categorization is not exhaustive.

After our review of existing surveys [2], [11], [16], [17], [18], we conclude that while they have looked into various strategies to mitigate MEV, few have taken the comprehensive and high-level approach that we aim to present in this paper. Most previous surveys focus on specific aspects of MEV mitigation, but they do not provide the complete picture. We strive to offer a more holistic perspective, covering a broader range of strategies and their interconnections.

III. BACKGROUND

In this section, we introduce background concepts that are necessary to understand the rest of the paper. We begin by discussing MEV, explaining its significance and impact on blockchain transactions. Next, we present the details of

front-running attacks, describing how they exploit transaction ordering to benefit malicious actors. Finally, we present L2 solutions highlighting how they enhance scalability in blockchain systems.

A. MAXIMAL EXTRACTABLE VALUE

MEV [1] refers to the maximum profit that a miner[1] can achieve from pending or confirmed transactions within a blockchain network. MEV represents the potential earnings validators can gain by strategically ordering transactions to their advantage. Validators possess the authority to include, exclude, or reorder transactions within a block. By leveraging this power, they can prioritize certain transactions that offer higher fees or manipulate transaction ordering to benefit from price movements, thereby maximizing their revenue. Currently, most actual MEV is extracted not only by validators but also by sophisticated users and automated bots, collectively known as MEV searchers. These searchers constantly monitor the blockchain for profitable opportunities, such as arbitrage, liquidations, and front-running. They employ advanced algorithms and high-frequency trading strategies to identify and capitalize on these opportunities before others can. This practice often occurs at the expense of other users, leading to concerns about fairness and market efficiency within the blockchain ecosystem. The implications for the users include: (1) financial loss: it happens through front-running on DEX platforms; (2) censorship: transactions can be censored or delayed by validators; and (3) high transaction fees: it is caused by network congestion.

MEV extraction also impacts the network layer, potentially destabilizing it through various attacks including (1) time-bandit attacks: validators attempt to rewrite blockchain history by reorganizing past blocks with profitable transactions; and (2) Priority Gas Auctions (PGA): high transaction fees are bid to prioritize transactions and capture opportunities [1].

B. FRONT-RUNNING

The most common method of extracting MEV is through front-running attacks. In these attacks, a transaction is placed ahead of another in the execution queue by bidding higher transaction fees. In Ethereum, transactions are temporarily stored in a public pool known as the mempool, which allows anyone to monitor and identify potential profit opportunities before they are included in a block. Malicious actors exploit this transparency by executing transactions with higher gas fees, effectively front-running the original transaction to capture profits before it is processed. When multiple front-runners compete for the same opportunity, it leads to a PGA [1], where the transaction with the highest gas fee wins the race to be included in the next block. This competitive bidding process can escalate quickly, causing network congestion as more transactions compete for limited block space. As a result, the overall transaction fees on the network increase, impacting regular users who are not participating in the auction. In certain situations, a validator might choose to censor or delay a specific transaction. This could be to prioritize their own transaction in order to profit, particularly if the potential profit exceeds the combined block reward and transaction fees.

Fig. 1 (a) shows an example of a front-running attack. Users submit transactions with specific fees to a blockchain node, where they are stored in a pool of pending transactions known as a mempool. The validator selects transactions from the mempool to create a block; typically, miners prioritize processing transactions that offer higher fees.

Fig. 1 (b), the user plans to buy a significant amount of a certain token on a DEX (e.g. 1000 token A). This large purchase will cause the token's price to rise after the order is executed. Before the transaction is broadcast to the network, a front-runner bot detects this large buy order in the mempool. The bot then places a buy order for the same amount of the token but with a higher transaction fee, allowing it to buy at a lower price. When the user's order is executed, it drives up the token's price, enabling the front-runner to sell their tokens at the increased price, thereby profiting from the user's transaction.

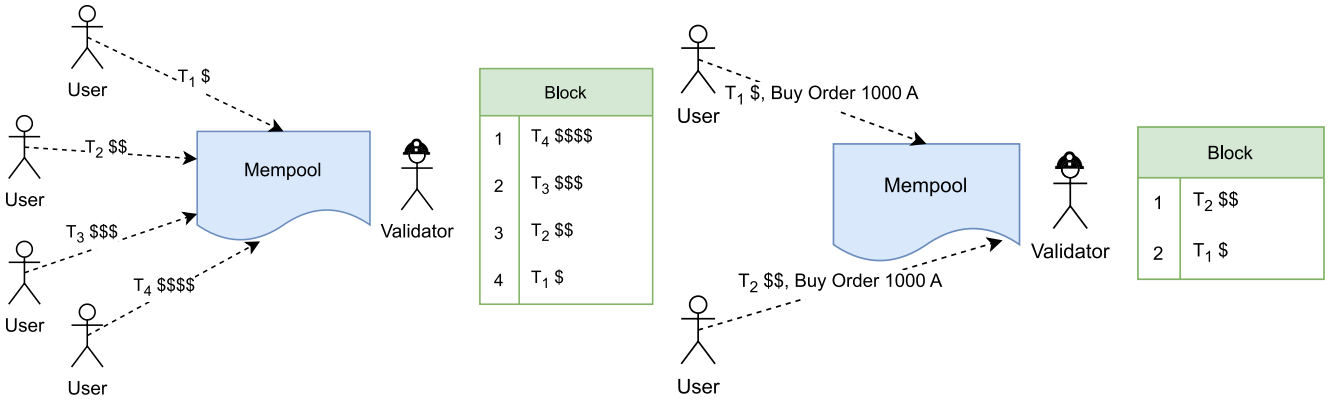
1) PGA EXAMPLE

In the previous example, there was one front-runner who only needed to set a higher transaction fee than the user's specified fee. Fig. 2 shows a scenario where multiple front-runner bots detect the user's transaction. If the user's fee is \$, the first bot sets a higher fee, say \$\$. The next bot sees this and sets an even higher fee, \$\$\$. This competition among bots drives the transaction fees higher, creating a situation known as a Priority Gas Auction. During this process, the validator, seeking to maximize profit, selects the transaction with the highest fee offer for inclusion in the block. Consequently, the user who sends the highest transaction fee for T_n is chosen by the validator and becomes the winner of the auction.

C. L2 SCALING SOLUTIONS

L2 solutions offer a way to scale the underlying chain or layer 1 (L1) chain. Each L2 protocol has its own execution environment, resembling the Ethereum virtual machine. The key concept involves deploying decentralized applications (DApps) on L2 protocols, enabling computations and storage to take place off-chain. Rollups are a type of the L2 solution, in which computations and the application's state are kept in an off-chain protocol, and the proof data of transactions are stored on L1. In this study, we focus on the rollup protocols as an efficient and general-purpose off-chain solution. Rollups aggregate multiple transactions into a single batch and perform most computation off-chain, using the underlying network for security. There are two main types of rollups. Optimistic rollups operate on the principle of assuming transactions are correct unless proven otherwise. In contrast, Zero-Knowledge rollups use cryptographic proofs to validate transactions. A crucial component in the architecture of rollups is the sequencer. The sequencer plays a vital role in

[1]Miner and validator will be used interchangeably in this paper.



(a) Rational decision-making by validators when choosing transactions for block inclusion.

(b) Example of a Front-Running Attack

FIGURE 1. **Front-Running attacks caused by arbitrary transaction selection and ordering.**

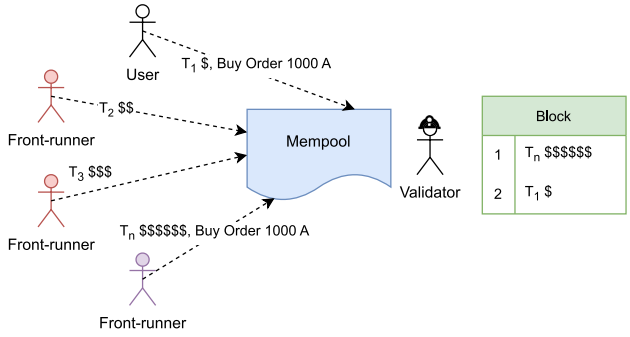


FIGURE 2. **Example of a priority gas auction (PGA) attack.**

maintaining the correct order of transactions and ensuring the consistency of the network state. When users submit transactions to a rollup chain, these transactions are initially processed and validated off-chain within the rollup network. The sequencer then collects these transactions and orders them based on their received time or other criteria, creating a batch or block of transactions. Periodically, the sequencer commits the batch to the underlying network, thereby ensuring the integrity of the transactions and the rollup's state. Transaction ordering and execution can be separated in rollup protocols, but in practice, they can be gathered in a single node [19], [20].

IV. TAXONOMY OF MEV MITIGATION STRATEGIES

In this section, we present a novel taxonomy outlining various methods to mitigate MEV (see Fig. 3). This taxonomy aims to offer a broad understanding of each mitigation technique, highlights its respective objectives and underlying strategies. The MEV mitigation taxonomy consists of two primary strategies, each addressing the issue from distinct perspectives: (1) prevention/reduction: it aims to eliminate MEV opportunities before they can be exploited; and (2) side-effect reduction: it focuses on mitigating the adverse impacts of MEV extraction rather than eliminating the opportunities themselves.

The prevention/reduction strategy involves removing or limiting the validator's ability to reorder transactions, thereby directly preventing them from extracting MEV. Conversely, MEV searchers are unable to leverage higher transaction fees to exploit the validators' capabilities. Prevention can also be achieved through **disincentivization**; this involves creating deterrents against engaging in MEV practices.

The side-effect reduction strategy focuses on mitigating the negative impacts of MEV extraction (PGA and time-bandit attacks) which can lead to network congestion and threaten the stability of the consensus protocol. Instead of eliminating MEV opportunities entirely, this strategy focuses on democratizing MEV. By making MEV extraction accessible to a broader group of participants, the current imbalance where validators have a disproportionate advantage is reduced.

In the following sections, we describe various proposed methods from the literature and industry, based on the strategies shown in Fig. 3. Table 1 presents key projects and articles discussed in the following sessions, highlighting their strategies, approaches, and main contributions.

V. FAIR ORDERING POLICY

A fair ordering policy is a method designed to eliminate a sequencer's ability to reorder transactions. It enforces a specific order for transactions to be included in a block by the sequencer. These policies are embedded into the protocol during its design by establishing explicit rules that the sequencer must adhere to when ordering transactions. These rules are enforced through the protocol's code and smart contracts, ensuring compliance. While this approach can significantly reduce the likelihood of front-running, the sequencer node can still censor or delay transactions. Additionally, in the case of rollups, where sequencer nodes have a private mempool, there is a risk that they could leak pool information to external front-runners, especially when there is only a single sequencer, which necessitates a strong trust assumption.

Ordering policies may be implemented in various settings. In a single sequencer scenario, a single node undertakes the

TABLE 1. Notable projects discussed in the survey.

Strategy		Work	Approach	Main Contribution	Reference
Prevention/Reduction	Removing Ability	TimeBoost	Fair ordering policy	Combines FCFS and auction ordering policies in a single sequencer setting.	[7]
		Metis	Fair ordering policy	Provides a decentralized sequencing rollup.	[21]
		Themis	Fair ordering policy	Applies batch-order-fairness definition in a decentralized setting.	[22]
		Wendy	Fair ordering policy	Applies timestamp-based ordering in a decentralized setting.	[23]
		Espresso	Fair ordering policy	Provides a shared sequencing network for all rollups.	[24]
	Disincentivizing	Shutter	Privacy-preservation	Provides an off-chain key management network for the mempool privacy through threshold encryption algorithm.	[25]
		F3B	Privacy-preservation	Provides an on-chain solution for mempool privacy through threshold encryption.	[26]
		Helix	Privacy-preservation	Combines the idea of threshold encryption and random selection of encrypted transactions.	[8]
		BlindPerm	Privacy-preservation	Proposes a mempool privacy solution with transaction encryption and ordering permutation.	[9]
		Radius	Privacy-preservation	Proposes a shared sequencing service with encrypted transactions through the delay encryption method.	[27]
		CoW Swap	Smart contract-level protection	provides a DEX platform with an off-chain component to protect pending trade requests.	[28]
		FairMM	Smart contract-level protection	provides a DEX platform with an off-chain component to protect pending trade requests.	[10]
Side-effects Reduction	Democratizing	MEV-Boost	Proposer-Builder separation	Provides an off-chain implementation for the PBS method.	[29]

sequencing task, with users directing their transactions to this node. Conversely, a multi-sequencer setup or decentralized sequencing involves multiple nodes collaborating to establish consensus on transaction order. The subsequent sections will delve into the challenges associated with fair ordering across these different configurations.

A. SINGLE SEQUENCER

The most straightforward ordering approach applicable to a single sequencer node is the First-Come-First-Serve (FCFS) algorithm. It orders transactions according to their arrival time at the sequencer node. It is reasonable to prioritize transactions based on their arrival time, ensuring fairness in processing. Furthermore, this approach also reduces front-running by limiting sequencers' ability to order transactions arbitrarily. However, implementing FCFS in practice presents some challenges and drawbacks. These include (a) Latency wars: users try to optimize their network latency with sequencer nodes to guarantee early inclusion of their transactions to a block; and (2) Spam attacks: malicious

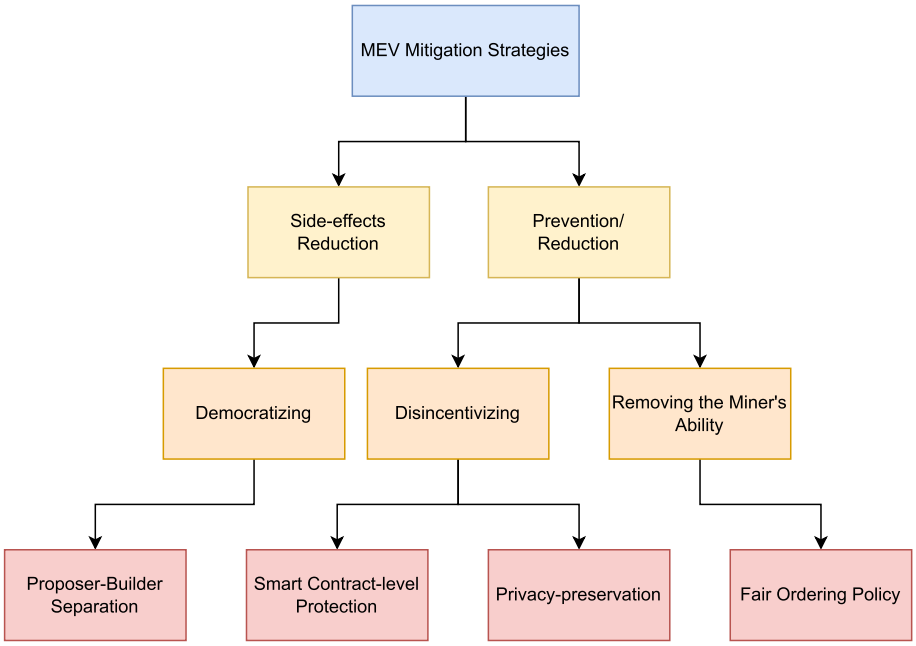


FIGURE 3. Taxonomy of MEV mitigation strategies.

TABLE 2. Prominent rollup protocols with their sequencing model.

Protocol	Rollup Type	Sequencing	Ordering Policy
Arbitrum	Optimistic	Single	FCFS
Optimism	Optimistic	Single	FCFS
Starknet	Zero-knowledge	Single	FCFS
Zksync	Zero-knowledge	Single	FCFS
Scroll	Zero-knowledge	Single	FCFS
Metis	Hybrid	Decentralized	FCFS

users flood the sequencer node with a large number of transactions with the intent to guarantee their priority in the queue. As a result, legitimate transactions may experience delays or even fail to be processed altogether.

Most notable rollups utilize a single sequencer with a FCFS ordering policy (see Table 2). FCFS ordering policy simplifies management and implementation processes but has certain implications. Firstly, users need to trust the sequencer node managed by the company. Secondly, there is a risk of a single point of failure; if the sequencer node encounters issues or is compromised, it could disrupt the entire rollup system, causing transaction delays or failures.

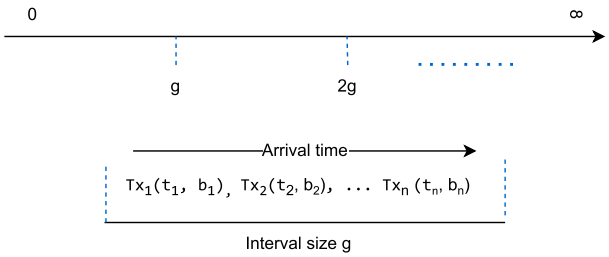


FIGURE 4. Granularity interval in TimeBoost algorithm.

Some protocols employ modified versions of the traditional FCFS algorithm to improve the fairness of transaction ordering. Although these solutions still contend with trust and single point of failure concerns, they can address issues associated with classic FCFS ordering, such as transaction spamming and latency optimization problems.

Kelkar et al. [7] introduce **TimeBoost**, a fair ordering protocol in a single sequencing setting. Fig. 4 shows the proposed algorithm, which orders transactions within a granularity interval g on rollup sequencers. It combines transaction timestamps and user bids to generate a score that determines the transaction order. Users can bid to effectively reduce their transaction's timestamp in the queue, effectively buying time. During the interval g, transactions are initially sorted based on their received time t_i; however, each user can bid b_i with their transaction to decrease the received time, thereby increasing the transaction's score. Ultimately, transactions are ordered based on their computed scores.

The score for a transaction $\text{tx}_i = (t_i, b_i)$ is computed as (1):

$$S(t_i, b_i) = \pi(b_i) - t_i. \tag{1}$$

The function $\pi(b_i)$ represents the priority achieved by bidding b_i. Transactions are ordered by descending scores; a higher score increases the likelihood of the transaction being processed sooner. The bidding function π is designed to satisfy the following properties:

1) $\pi(0) = 0$: Paying a bid of 0 provides no additional advantage.
2) $\pi'(b) > 0$: For all $b \in \mathbb{R}^+$, the priority increases with the bid.
3) $\lim_{b\to\infty} \pi(b) = g$: No transaction can outbid a transaction that arrived g time units earlier.
4) $\pi''(b) < 0$ for all $b \in \mathbb{R}^+$: The priority function is concave, meaning the cost of gaining priority increases with the bid.

The simplest bidding function that satisfies these constraints is expressed as (2):

$$\pi(b_i) = \frac{gb_i}{b_i + c}. \tag{2}$$

where c is constant. Unlike FCFS, TimeBoost avoids the inefficient latency competition inherent to the FCFS policy. Incorporating bids into the transaction ordering process encourages players to focus on bidding rather than investing heavily in low-latency infrastructure. In pure-latency approaches like FCFS, only the fastest players benefit, often leading to unfair advantages.

TimeBoost combines timestamps and bids, enabling bidding among transactions within a short time frame. This approach is superior to pure-bidding methods, which can be vulnerable to attacks (e.g., PGAs) that threaten network stability. Unlike PGAs, where any user across the entire network can participate in the bidding process, potentially causing network congestion, TimeBoost restricts bidding to a limited number of transactions within each granularity interval.

B. DECENTRALIZED SEQUENCING

Decentralized sequencing has emerged as a viable alternative by addressing the limitations associated with relying on a single sequencer (e.g., trust and single point of failure). However, it introduces a substantial challenge: accurately determining the transaction order. In decentralized systems, this frequently involves the need for synchronized clocks. Because of network delays, influenced by the geographical distances between nodes and the available network bandwidth for users, transactions might reach different nodes at different times and in various orders. As a result, the transaction order in each node's mempool may vary.

Metis [21] is one of the recent rollup chains pioneering decentralized sequencing, primarily addressing the single point of failure issue. It features a straightforward method for ordering transactions in a decentralized environment. All sequencer nodes receive transactions, and in each round, one sequencer is selected to construct a block. Metis uses a rotation mechanism for Sequencer selection, with all Sequencer Lists stored in a smart contract address controlled by multiparty computation (MPC). The selection is based on each sequencer node's voting weight (related to the staked amount of token) and a randomly generated hash value, ensuring a fairer process. If a sequencer stops operating, it is rotated out to maintain continuous network operation. The selected sequencer orders transactions based on the FCFS policy using the order of transactions in its local mempool. Metis has a permissioned pool of sequencers governed by a DAO (Decentralized Autonomous Organization). Prospective sequencers must submit proposals, subject to a voting process requiring majority approval from DAO members to join the network. Additionally, sequencers are mandated to stake a certain quantity of tokens as a deterrent against malicious behavior. Mechanisms for slashing exist to penalize sequencers, while rewards are offered for transaction processing and block production, thereby incentivizing honest behavior. The slashing mechanism addresses malicious actions or poor performance by Sequencer nodes. Offending Sequencers are immediately removed from the pool and subjected to a review process. Slashing cases, categorized by severity, include slow Sequencers (low severity), non-block production (medium severity), multiple node outages (high severity), and malicious execution result modification (critical severity). Programmatic detection and enforcement handle these issues, with additional penalties determined by governance proposals. Actions against offenders can include slashing a percentage of their stake, blacklisting, and removal from consensus. Transaction ordering manipulation is also considered a malicious activity in Metis. Currently, there is no automated detection process for this. The protocol encourages the community to identify and report such attacks. If a report is verified as accurate, the sequencer will be penalized accordingly [21], [30].

More complex forms of decentralized sequencing have been proposed in the literature including [22], [31], [32], [33]; however, they have not yet been implemented in a production blockchain. In these methods, all sequencer nodes receive transactions, and a leader sequencer is selected each round to establish the final transaction order. Instead of relying solely on its local mempool, the leader integrates the local orders from all sequencers based on a fair ordering definition. This definition determines the final order of transactions by deciding, for each transaction pair (Tx_0, Tx_1), which one should be processed first.

Kelkar et al. [22] introduced **Themis** which provides a fair ordering of transactions. Fig. 5 shows the Themis protocol ordering process. It is a leader-based protocol where all nodes transmit their transaction orders to the chosen leader, determined by their receive times (see Fig. 5 (1)). Subsequently, the leader compiles a fair-ordering proposal from the transaction orders received from other nodes (see Fig. 5 (2)). Finally, the leader sends the final ordering, accompanied by a SNARK proof of computation, to all other sequencer nodes involved in the sequencing process (see Fig. 5 (3)). The fair ordering definition used by Themis is

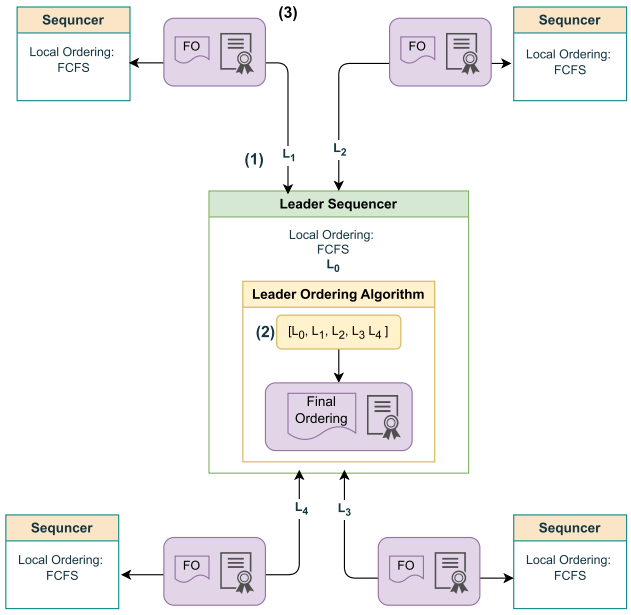


FIGURE 5. Themis network fair ordering protocol by the leader.

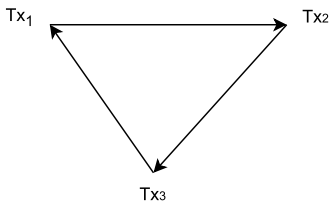


FIGURE 6. Example of a Condorcet Cycle.

known as Batch-Order-Fairness. According to this definition, if two transactions Tx_0 and Tx_1 are received by all sequencer nodes, and if a fraction (more than $\frac{1}{2}$) of these nodes received Tx_0 before Tx_1, then all honest nodes should order Tx_0 no later than Tx_1. A dependency graph is created where the vertices represent transactions and directed edges indicate the order between pairs of transactions. Additionally, a proof mechanism allows nodes to verify the accuracy of the final ordering proposed by the leader, based on the specified fair ordering algorithm [22]. This ordering policy can encounter the phenomenon known as the **Condorcet paradox or cycle**.

textbfFor example, let us consider three transactions received by three sequencer nodes, with each node receiving the transactions in a different order due to network latency and the nodes' geographical locations. The orders are presented as follows:

- **Node A**: $Tx_1 > Tx_2 > Tx_3$
- **Node B**: $Tx_2 > Tx_3 > Tx_1$
- **Node C**: $Tx_3 > Tx_1 > Tx_2$

Fig. 6 shows the resulting dependency graph. When cycles are present in the graph, it becomes impossible to determine the final order of transactions within the cycle. Moreover, the consensus process can be stalled. In protocols like Aequitas [32], this leads to a loss of liveness, where transactions are not processed in a timely manner or at all; the protocol gets stuck trying to resolve the cycle. High latency in a network can cause nodes to receive transactions in different orders, increasing the likelihood of conflicting preferences and thus cycles. Themis addresses this issue by using a method called deferred ordering, which helps manage the impact of cycles by ensuring that transactions do not have to wait indefinitely for ordering. In the following, we present the steps executed by Themis in the context of the example shown in Fig. 6. Let us consider that node *A* is selected as a leader.

1) **Partial Ordering:** Node *A* proposes a block that includes transactions Tx_1 and Tx_2 as fully ordered, but Tx_3 is only partially ordered. This means that the final position of Tx_3 in the block is not yet determined.
2) **Deferred Ordering Process:** The proposed block is broadcast to the other nodes. Nodes *B* and *C* receive the block from Node *A*. They agree on the order of Tx_1 and Tx_2 but still have their own views on Tx_3. The nodes defer the final ordering of Tx_3 to the next block proposed by the next leader.
3) **Final Ordering by the Next Leader:** Node *B*, the next leader, creates a new block. This block finalizes the position of Tx_3, considering the partially ordered state from node *A*'s block and any new transactions. Node *B* includes a complete ordering for Tx_3 along with new transactions (e.g., Tx_4, Tx_5).
4) **Consensus and Block Commitment:** The new block proposed by node *B* is broadcast and agreed upon by all nodes. The blockchain now has a consistent order: Block 1 (Tx_1, Tx_2, partially ordered Tx_3), Block 2 (Tx_3, Tx_4, Tx_5).

Kursawe [23] introduces Wendy which applies timestamp-based ordering in a decentralized setting; they leverage synchronized clocks to ensure that transactions are processed in the order they are received based on their timestamps. In Wendy, all nodes in the network maintain synchronized clocks. This synchronization can be achieved using protocols such as Network Time Protocol (NTP). When a node receives a transaction, it assigns a timestamp based on its local clock. The node then broadcasts the transaction along with its timestamp to the rest of the network. Each node has a mempool where it stores incoming transactions along with their timestamps. Nodes follow a predefined ordering policy based on the timestamps. Transactions with earlier timestamps are given higher priority for inclusion in the next block. During the consensus process, nodes propose and validate blocks of transactions. The validator must order the transactions in the block according to their timestamps, ensuring that transactions with earlier timestamps are included before those with later timestamps. When a node receives a proposed block, it verifies that the transactions are ordered correctly according to their timestamps. If the ordering is not correct, the block is rejected.

C. SHARED SEQUENCERS

Shared sequencing is a technique that has been discussed in recent years within the rollup ecosystem. It offers transaction sequencing for multiple rollups through a third-party service, enabling the sequencing process to be shared among various rollup chains. A shared sequencer operates independently of rollup chains by separating the executor role from the sequencer role. In this setup, the sequencer functions solely as a sequencer.

A shared sequencer can be either a single sequencer or a decentralized network of sequencers. In practice, because the shared sequencer handles requests from multiple rollups and prevents a single point of failure, it typically comprises a decentralized network of sequencers. Similar to a network of validators in a blockchain, this shared sequencing network forms a peer-to-peer network among sequencer nodes, which can use their own consensus mechanism to agree on the order of transactions. It can significantly reduce the cost of running rollups and accelerate the move towards decentralization. However, rollups will need to share transaction fees and MEV opportunities with other entities.

Espresso [24] is an example of a decentralized shared sequencing protocol. The architecture of the Espresso system is illustrated in Fig. 7. When a user sends a transaction to a rollup chain, following steps are executed:

1) The rollup receives the transaction and forwards it, along with a rollup ID, to the shared sequencer.
2) The sequencer network creates blocks containing an ordered list of transactions.
3) The rollup receives the block and executes transactions on its network.
4) The sequencer posts the log of executed transactions to L1 as a commitment (on its sequencer smart contract).
5) After executing the block of transactions, the rollup posts the updated state of the chain to L1 (on the rollup smart contract).
6) The rollup contract then reads the block commitment from the sequencer contract. The verification of state transition depends on the rollup type: a ZK-rollup verifies the proof, while an optimistic rollup waits for fraud proofs.

This process occurs simultaneously for multiple rollups. There is an incentive for rollups to forward transactions to the Espresso network, known as the sequencer marketplace. When a rollup receives a transaction from a user, it can choose to process it through its own sequencer node or sell its sequencing rights through an open marketplace. [34].

In a shared sequencing block that contains transactions from different rollups, one significant opportunity that arises is cross-domain MEV. In this context, transactions that span across multiple rollups can be strategically ordered or timed to exploit price differences, arbitrage opportunities, or other financial discrepancies between these domains.

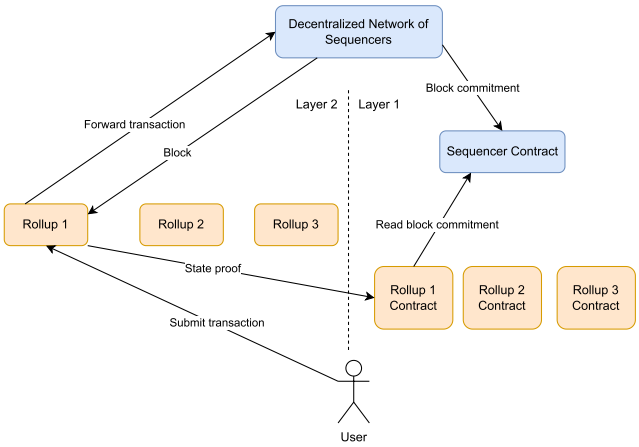


FIGURE 7. Espresso network architecture as a decentralized shared sequencer.

D. DISCUSSION

Table 3 presents a comparison of notable protocols discussed in the previous sections on fair ordering methods. Most protocols feature a set of permissioned sequencers, except for Espresso, [34] which allows anyone to join the network and run their own sequencer node. All the protocols are leader-based, with a single sequencer acting as the leader or, in the context of decentralized sequencing, one sequencer being selected as the leader for each block. All the protocols mentioned in the table, except Wendy [23], do not require a synchronized clock for their algorithms. Wendy's use of timestamps for each transaction increases communication overhead. Over time, clocks on different nodes can drift apart due to slight differences in their frequencies, leading to inconsistencies in timestamps and potential issues in the fair ordering process. Synchronizing clocks also adds network traffic, as nodes must periodically exchange time information, consuming additional computational and bandwidth resources that could be used for other operations.

Table 2 indicates that most rollups use the classic FCFS ordering policy. TimeBoost [7] is a proposed transaction ordering method for single sequencer rollups, combining bidding and time-based ordering methods. Metis is another rollup network that has proposed a decentralized sequencing method. While it addresses the single point of failure issue and reduces the trust required in a single sequencer, it still relies on the classic FCFS sequencing method. Themis [22] and Wendy [23] both propose decentralized sequencing methods but with different ordering policies. Themis operates without requiring a synchronized clock and maintains an $O(n)$ communication cost between sequencer nodes, utilizing the SNARK protocol. In contrast, Wendy requires synchronized clocks and has an $O(n^2)$ communication complexity. Shared sequencing protocols like Espresso [34] reduce the cost of running a network of sequencers for rollup chains. Rollups can either use their own sequencing policy or delegate the task to a third-party sequencing network through an auction mechanism.

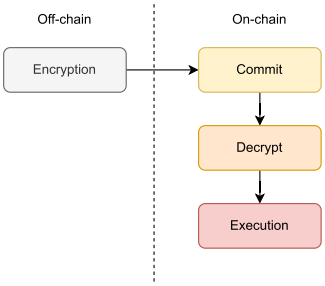


FIGURE 8. General approach in mempool encryption methods.

As Wendy uses a timestamp-based ordering mechanism; making it difficult for any single node to censor transactions because the order is determined by the timestamps, not by any single node's decision. Themis enforces batch-order fairness, meaning that transactions are ordered based on the order in which they were seen by a significant fraction of honest nodes. This reduces the ability of any single node to manipulate the order for censorship purposes. It employs SNARKs to achieve efficient and secure validation of the ordering process. This ensures that even if a node tries to censor transactions, it cannot tamper with the verification process without being detected.

VI. PRIVACY-PRESERVING MITIGATION METHODS

The main objective for the privacy-preserving methods is to hide the contents of transactions until their order is confirmed. These approaches can effectively eliminate MEV extraction opportunities and reduce the risk of transaction censorship by malicious sequencers. Since the transaction is not visible in the public mempool, it is less vulnerable to front-running and other types of attacks that exploit transaction visibility.

The primary focus of research in this domain is on **mempool privacy** or **encrypted mempool** (see Fig. 8), involving the encryption of pending transactions in the mempool and their decryption after their order is finalized, just before inclusion in a block [35]. Another privacy-preserving approach that helps prevent front-running involves **private transactions**. This is achieved by bypassing the mempool and directly sending your transaction to a validator, who then places the transaction in the newly created block ahead of other transaction [36].

A. ENCRYPTED MEMPOOL

The main techniques for encrypting the mempool include (1) **Threshold encryption**, which entails multiple trusted parties (referred to as key holders or keypers) decrypting transactions collaboratively without revealing the decryption key; (2) **Delay encryption**, where transactions remain encrypted for a designated duration before decryption; and (3) **Trusted Execution Environments (TEEs)** which guarantee security and privacy prior to transaction confirmation [37]. We will explore these three methods in detail.

1) THRESHOLD ENCRYPTION

In threshold encryption, the symmetric key is divided into several pieces (shares) and distributed to a key-holding committee during the encryption stage. To decrypt, a certain threshold of the total shares is required to reconstruct the symmetric key, making it impossible for a single attacker to attempt decryption before the transaction order is decided. This requires a degree of trust in the key-holding committee as a third party. The trust assumption here is that a certain threshold of committee members will share their portions of the key in a timely manner. The general process of the threshold encryption method is outlined in the following steps (see Fig. 9).

1) **Key generation:** Each member of the committee generates a share of the decryption key based on the Distributed Key Generation (DKG) algorithm [38]. This process ensures that no single member has the complete decryption capability, thereby distributing control over the decryption process among multiple parties.
2) **Encryption phase:** Transactions are encrypted using a public key from the trusted key management committee with an honest majority.
3) **Submission of encrypted transactions:** Encrypted transactions are submitted to the mempool. Since the transactions are encrypted, they cannot be tampered with or manipulated by validators.
4) **Transaction commitment:** The encrypted transactions are then posted on the blockchain as a commitment, but they can not be executed. At this point, the transactions are visible to all participants, but they are encrypted, and their contents are not accessible without the decryption key.
5) **Decryption phase:** Once a transaction is recorded on the blockchain, the key holders can initiate the decryption process. The contents of the transactions remain private until a threshold number of holders start the decryption process. This safeguard ensures that, even in the event of some participants being compromised, the integrity of the transactions is maintained as long as the majority of participants remain honest.
6) **Execution of Transactions:** Once the threshold decryption process concludes, the transactions are processed in a sequence based on the commitment already posted on the blockchain. This predetermined order is unrelated to the content of the transactions, effectively blocking any potential for MEV exploitation through strategies that rely on content-based ordering.

The key management committee can be chosen from validators on the main chain, who are responsible for consensus and block creation [39]. However, this approach may introduce overhead during the decryption phase and necessitate modifications to the protocol layer. Alternatively, an off-chain mechanism managed by trusted parties can

TABLE 3. Notable sequencing protocols.

Protocol	Sequencing Model	Permissioned	Leader based	Synchronized Clock	Ordering Policy	Verification	Censorship Resistance	Comm. Complexity
TimeBoost [7]	Single	Yes	Yes	No	Modified FCFS	Trust	No	—
Metis [21]	Decentralized	Yes	Yes	No	FCFS	DAO, Slash	No	—
Themis [22]	Decentralized	Yes	Yes	No	Batch-order fairness	SNARK	Yes	O(n)
Wendy [31]	Decentralized	Yes	Yes	Yes	Timestamp-based	Timestamps	Yes	O(n^2)
Espresso [34]	Shared	No	Yes	No	Auction-based	PoS consensus, Slashing Rules	Yes	O(n)

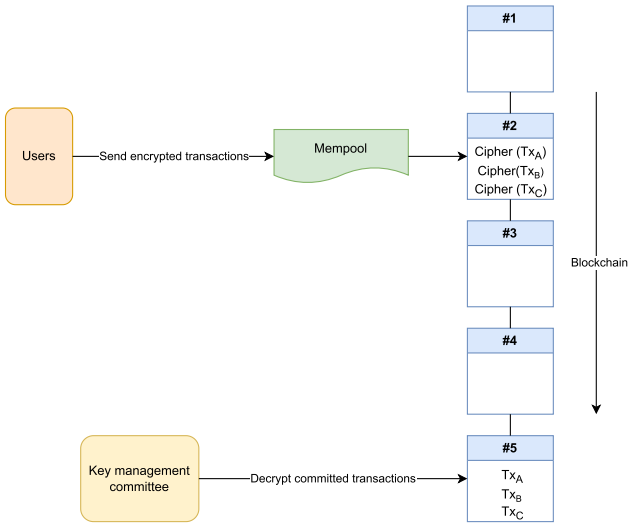


FIGURE 9. Users send encrypted transactions using the threshold encryption method.

handle key generation and decryption processes. An example of such a protocol is **Shutter** [25], which offloads the key management process to an off-chain environment. Another key advantage of Shutter, in terms of efficiency, is that it uses a single decryption key for each epoch in POS consensus (approximately for 32 blocks), whereas Ferveo [39] generates a decryption key for each transaction [40].

Yakira et al. [26] propose Flash Freezing Flash Boys (F3B), a novel blockchain architecture that employs a commit-and-reveal scheme. In this scheme, the contents of transactions are encrypted and only revealed by a decentralized secret-management committee after the underlying consensus layer has committed the transaction. The key components of F3B include: encrypted transaction senders, consensus group, and secret-management committee (set of trusted nodes which are responsible for keeping the encryption key). Although consensus nodes and management nodes are two separate entities in some situations (e.g., when a permissioned blockchain is used as the underlying blockchain) they can be the same or can be run on a single server. The process of the F3B threshold encryption method is outlined in the following steps:

1) **Key Generation Phase:** The secret-management committee generates a secret key for each member and a single public key using a DKG algorithm [38]. This process is repeated at specific intervals during the reconfiguration phase, which allows for the addition and removal of committee members.
2) **Key Selection and Encryption Phase:**
   - **Symmetric key generation:** The user or the sender of the transaction generates a random number on their side; it is considered as the symmetric key k.
   - **Encryption of the Symmetric Key:** The sender then encrypts the symmetric key k using the public key pk_{smc} of the secret-management committee to produce the encrypted symmetric key c_k.
   - **Transaction Encryption:** The sender encrypts the transaction tx using the symmetric key k, resulting in the encrypted transaction c_{tx}.
   - **Sending Encrypted Data:** The sender sends the pair (c_{tx}, c_k) to the consensus group, which validates the integrity of the data. Once validated, the pair is included in a new block of the blockchain.
3) **Waiting for Commitment:** At least m block confirmations are needed for the commitment phase.
4) **Key Release and Execution:**
   - **Retrieving the Encrypted Symmetric Key:** Each member in the secret-management committee

retrieves the encrypted symmetric key c_k from the blockchain. The key c_k was previously encrypted with the committee's public key pk_{smc} and recorded on the blockchain during the second phase. Each member individually decrypts their share of the symmetric key k. The decryption is done using the member's private key share, which corresponds to their share in the DKG algorithm used to generate pk_{smc}. This results in partial decrypted key shares.

- **Verification by Consensus Nodes:** To ensure that the decryption was done correctly, each member generates a non-interactive zero-knowledge (NIZK) proof. This proof verifies that their decrypted key share is correct without revealing the key share itself. The consensus nodes in the blockchain network verify the NIZK proofs provided by the trustees. If the proofs are valid, it confirms that the decrypted key shares are correct and trustworthy.
- **Reconstructing the Symmetric Key:** Once a sufficient number of correct key shares and corresponding proofs are collected, the consensus nodes use Lagrange interpolation to reconstruct the original symmetric key k. This is possible because the DKG algorithm ensures that the key can be reconstructed with a threshold number of shares.
- **Decrypting the Transaction:** With the symmetric key k reconstructed, the consensus nodes decrypt the encrypted transaction c_{tx} to obtain the original transaction tx.
- **Transaction Verification and Execution:** The decrypted transaction tx is then verified to ensure it meets all necessary conditions and is valid. Once verified, the transaction is executed according to the rules and logic defined in the blockchain protocol.

F3B requires writing data onto the blockchain only once, significantly reducing overhead. However, F3B requires modifications to the blockchain protocol to enable consensus nodes to verify and process encrypted transactions and key shares; Additional logic must be added to handle the decryption phase and the subsequent transaction validation [26].

Helix [8] is a hybrid protocol that combined the idea of fair ordering with threshold encryption. The protocol operates on a fully connected and synchronous network of nodes, ensuring that each transaction propagates to all nodes, resulting in all nodes having the same set of transactions. **Helix** is a leader-based protocol where the leader is selected among a committee of nodes, and the committee itself is established through an election process. Each node has a reputation score that increases with honest participation in block validation and proposals. The higher a node's reputation score, the greater its chances of being selected as a committee member or leader.

As the consensus and transaction ordering processes are confined to a limited number of committee nodes, the protocol scales with the number of nodes. The protocol selects a random set of pending encrypted transactions to include in a block using a Verifiable Random Function (VRF) [41]; the objective is to guarantee that the selection of pending transactions is done in a transparent, verifiable, and unbiased manner, enhancing the fairness and security of the blockchain network. It ensures transaction censorship resistance by employing a threshold encryption mechanism, preventing committee and leader nodes from knowing the content and the issuer address of a transaction until their order is finalized.

For example, if Node P is selected to propose a block in one round among the committee members, the process of block creation is as follows.

1) A random seed is generated for the round; it is used as the input for the VRF.
2) Node P computes a VRF output using the random seed and its private key share. It also generates a proof of VRF computation that can be verified by other nodes without knowing the private key.
3) Each transaction in the pool is hashed to generate a unique identifier. For each transaction Tx_i, node P concatenates the transaction's hash with the VRF output to form a combined string. The combined string is then hashed to produce a ranking score for the transaction.
4) The transactions are ordered based on their ranking scores in an ascending order. This ensures that each node, using the same VRF output and the same set of transactions, will compute the same order for the transactions.
5) Node P creates a block with the top k transactions; along with VRF proof, the block is proposed to other committee members.
6) The committee members verify the VRF proof to ensure that the transaction selection process was random and fair. They run a Byzantine Fault Tolerance (BFT) protocol to reach consensus on the proposed block.
7) Once consensus is reached, nodes begin the threshold decryption process for the transactions in the block. The decrypted transactions are then validated for correctness.
8) The validated transactions are included in the new block, which is then appended to the blockchain.

mempool Encryption can be improved with other techniques. For example, Kavousi et al. [9] introduce a framework called **BlindPerm**, which enhances encrypted mempool, with permutation techniques, providing multi-layer protection against MEV. It uses randomized permutations to shuffle the order of transactions within a committed block before they are executed. This framework is designed to work with BFT-style consensus mechanisms and is shown to be

efficient; it introduces essentially no overheads and requires no additional services. The process of the BlindPerm is outlined in the following steps.

1) The validator who is selected as the leader, proposes a block, containing encrypted transactions with an arbitrary order from the mempool.
2) Once a block is committed, validators decrypt the transactions, based on the used mempool encryption, and derive a seed for permutation.
3) Each validator uses the derived seed to permute the order of transactions by applying the permutation function Permute(seed_i, B_i) to shuffle the transactions in the block.
4) The permuted block B_i' is executed by all validators.

2) DELAY ENCRYPTION

Delay encryption, also known as **time-lock encryption**, is a cryptographic mechanism which provides a secure way to ensure that data remains confidential until a specified time has passed. It leverages cryptographic primitives like Verifiable Delay Functions (VDFs) and Time-lock Puzzles. This concept is particularly useful in scenarios where data should remain confidential until a specific future date or a condition is met. The core idea behind delay encryption is to tie the decryption key to a time-dependent condition (time-lock puzzle), making it inaccessible until that condition is satisfied.

VDF is a cryptographic primitive that allows a prover to convince a verifier that they have waited for a certain amount of time without revealing any information about the time they waited. This property is crucial for delay encryption; it allows the decryption key to be derived from a time-dependent condition without revealing the actual time waited. Time-lock puzzles are designed in such a way that solving them requires a certain amount of computational work or time. The solution to the puzzle, which is the decryption key, can only be obtained after the required time has passed [42], [43]. For better understanding, let us consider the following example.

- **Creating a Time-lock Puzzle:**
  1) Generate prime numbers: $p = 61, q = 53$
  2) Compute $N = pq$: $N = 61 \times 53 = 3233$
  3) Choose a generator number and an exponent: $g = 2, t = 1000$
  4) Compute puzzle $g^{2^t} \bmod N$: $2^{(2^{1000})} \bmod 3233$
  5) Drive a symmetric key K by hashing the result of the computation which is the puzzle solution S_o: K=H(S_o))
  6) Encrypt a transaction with K: C_{T_x} =Enc(K, T_x)
- **Solving a Time-lock Puzzle:**
  1) Solve the time-lock puzzle using public parameters N, g, t by raising g to the power of 2^t. This process takes a specific amount of time, thereby introducing a delay before the decryption phase.
  2) Drive the symmetric key K by hashing the computed solution from the previous step: K=H(S_o))
  3) Decrypt the transaction with key K: T_x = Dec(K, C_{T_x})

The process of Radius is outlined in the following steps.
Radius [27] is a shared sequencing protocol designed to enhance interoperability among rollup chains. It employs delay encryption to create an encrypted mempool within its sequencing layer. This approach helps prevent MEV extraction through front-running and other reordering attacks, while also ensuring censorship resistance. The process starts with generating a transaction by a user, the following steps are as follows.

1) The user generates a symmetric key using a time-lock puzzle set to a specific time T. While the user already knows the solution, it will take the sequencer layer time T to solve the puzzle and reconstruct the symmetric key.
2) The user encrypts the transaction using the generated symmetric key.
3) The user also generates a ZK-SNARK proof for the sequencing layer to verify the integrity of the time-lock puzzle and the encrypted transaction.
4) The user sends the encrypted transaction and the proof to the sequencing layer.
5) Upon receiving the encrypted transaction, the sequencing layer provides users with an order commitment, ensuring the transaction's sequence remains unchanged. The ordered list is based on the encrypted transactions because, before the time T has elapsed, the sequencing layer cannot solve the puzzle and decrypt the transactions.
6) After time T has passed, the sequencer solves the puzzle and decrypts the transaction using the derived symmetric key.

Using ZK proof in Radius is to check the validity of the time-lock puzzle before trying to solve the puzzle; this allows to prevent waste of resources and DOS attacks which is called Practical Verifiable Delay Encryption (PVDE) [27], [44].

3) TRUSTED EXECUTION ENVIRONMENT

In this approach, validators must operate Trusted Execution Environments (TEEs). These TEEs can be integrated with the threshold encryption technique. The concept involves each network node possessing a TEE, such as Intel's Software Guard Extensions (SGX) [45], and collectively maintaining an encryption key. Transactions are encrypted and sent to these TEEs, where they remain encrypted until they are included in a finalized block on the blockchain. The security of this method relies on the cost associated with compromising a specific hardware device [37], [39].
Node operators can run a TEE on their server if it supports SGX, or they can use a remote server with SGX support. SGX is an extension found in some Intel CPUs, providing a set of operations that enable the creation of a TEE, known

as an enclave. An enclave is a protected region in memory and the CPU where data and code are isolated and accessible only within this secure area. The encryption key is generated and stored within the CPU. During execution, data and code are automatically decrypted by the CPU, processed, and then re-encrypted to maintain security [45].
Using the TEE method with a public-key cryptography scheme, users have access to the public key of the enclave. They encrypt their messages using this public key and send them to the enclave. The enclave functions like a private mempool, keeping pending transactions hidden from view. Inside the enclave, transactions are decrypted, executed, and then published on the blockchain. One promising project that uses TEE is Suave [46], which has not yet been officially deployed.

4) DISCUSSION

Table 4 shows an overview of the three primary methods for mempool encryption. In this context, users are anyone who wants to send their transactions through these methods. With Threshold encryption, trust is placed in a committee of key holders. Delay encryption, on the other hand, is trustless as each user possesses their own encryption key independent of any specific committee. However, when employing TEEs, trust is required in the hardware to safeguard secret keys.

Regarding security, in Threshold encryption, the system remains secure and functional as long as at least $\frac{2}{3}$ of key holders are honest. In Delay encryption, security hinges on the safeguarding of users' symmetric keys. When utilizing hardware, security is contingent upon factors such as access controls, secure memory, and tamper resistance of the hardware.

Each method faces distinct challenges. With Threshold encryption, the risk lies in committee members colluding to disclose private keys to front-runners or ceasing decryption of transactions. In delay encryption, users are responsible for generating puzzles for each transaction and safeguarding their symmetric keys. Consequently, users may face problems such as losing their keys or having them stolen. Challenges in hardware usage include compatibility with other software and modules, as well as reliance on specific hardware devices.

The complexity of the threshold encryption method can be high compared to other methods because it requires managing multiple keys and coordinating among participants, necessitating robust key distribution and management protocols. In delay encryption, the user must generate a time-lock puzzle and manage their symmetric key for each transaction. With TEE, all key management is handled by the hardware.

The threshold encryption method lacks scalability because its complexity grows with the number of committee members. In contrast, delay encryption offers high scalability, as it can be uniformly applied across transactions. The scalability of the TEE method, however, is constrained by the availability and performance of TEE hardware.

B. PRIVATE TRANSACTIONS

Private transactions are sent directly to validators, bypassing the public mempool. This approach enhances privacy by avoiding exposure to front-running attacks. The process of sending a private transaction is as follows [36].

1) **Create the Transaction:** The user generates a transaction without any encryption, just like a standard Ethereum transaction.
2) **Direct Submission to Validators:** Instead of broadcasting the transaction to the public mempool, the user sends the transaction directly to one or more specific validators. This can be done through Private Remote Procedure Call (RPC) endpoints for Ethereum or using specialized services that facilitate private transactions (e.g., Flashbots API [47]).
3) **Validator Inclusion:** The targeted validator receives the private transaction and includes it in the next block they produce (with high priority). Since the transaction is not visible in the public mempool, it is less susceptible to front-running and other types of attacks that exploit transaction visibility.
4) **Broadcasting:** The block containing the private transaction is broadcast to the entire network. At this stage, other validators can see the transactions, but it is too late to be front-run.

Private transactions are not widely used to avoid MEV due to limited availability and trust issues. Not all validators support private transaction pools, which reduces the chances of transactions being quickly included in blocks. Additionally, users need to trust that validators won't leak or exploit the transaction for front-running, which makes users hesitant to adopt the method. Validators also have little incentive to support private transactions, as MEV represents a source of profit for them. Furthermore, using private transactions can reduce exposure to the broader market. The added technical complexity, along with the risk that private transactions may revert to the public mempool where they remain vulnerable to front-running further limits their widespread adoption.

VII. SMART CONTRACT-LEVEL PROTECTION

The goal of Smart Contract-level MEV extraction protection is to mitigate the potential for MEV extraction within the application layer while preserving the protocol of the underlying blockchain network. The challenge lies in designing smart contracts that can protect against MEV extraction without altering the fundamental protocol of the blockchain. Certain types of applications are particularly susceptible to front-running. These applications include: DEX, Gambling, Auction, Buying a Domain name, and rare NFTs.

Solutions fall into two primary categories: the first operates entirely within smart contracts, utilizing methods such as commit-and-reveal [2] and batch processing, while the second category involves off-chain components.

TABLE 4. Prominent mempool encryption methods to prevent MEV-extraction.

Privacy-preserving methods	Trust Level	Security	Challenge	Key Management	Scalability
Threshold Encryption	Committee of key holders	2/3 of key holders	Collusion of Committee members	Committee members	Committee members
Delay Encryption	Trustless	Users' symmetric key	Generating puzzles & keeping symmetric keys safe	Users	Highly scalable
TEE	Hardware device stores secret keys	Hardware device	Compatibility, Hardware dependency, Centralization issue	Hardware device	Performance of hardware device

FIGURE 10. CoW Swap protocol architecture (Adapted from docs.cow.fi).

CoW Swap [28] is a solution in the second category; it is the first DEX aggregator, designed to facilitate asset swaps and limit orders with a hybrid MEV protection solution. CoW Swap (see Fig. 10) sets up a private order flow marketplace where users can swap one asset for another at a specified price. Traders sign their trading intentions (intents), which are then aggregated off-chain and executed in batch settlements on-chain. CoW Swap introduces an uniform price clearing mechanism that assures traders receive the same price for an asset within the same block. It employs a decentralized network of solvers, or computer programs, that process the order-book to determine optimal prices and traded amounts. These solvers compete in batch auctions to find the best execution route across different liquidity sources, earning the right to execute the transactions and capturing any surplus from the trades [28], [48].

Ciampi et al. [10] propose an AMM cryptocurrency exchange named **FairMM**, which resists front-running attacks by incorporating off-chain components (second category). FairMM can prevent reordering trade transactions, censoring trades, and including specific trade requests. The authors use game theory along with some incentive mechanisms in the smart-contract-level to remove reordering attacks. FairMM has higher throughput and lower trade cost compared to Uniswap which is an AMM crypto-market exchange. In FairMM, traders and the market maker (MM) communicate off-chain via secure channels like TLS. Traders form a queue, and the MM issues a ticket (identified by a cryptographic hash and signed by MM) to the trader at the front of the queue. This ticketing system ensures the integrity of the trading history and prevents reordering attacks. The process of FairMM is outlined in the following steps (see Fig. 11).

1) A trader initiates a trade by sending a trade request, through an off-chain secure communication channel (e.g., TLS), to MM module which is responsible for managing and executing trades.
2) MM issues a ticket to the trader. This ticket includes a cryptographic hash; it is signed by MM to ensure the integrity and authenticity of the request. MM manages a queue of trade requests. The ticketing system helps in maintaining the order of requests and prevents reordering attacks.
3) The trader decides to proceed with or abort the trade and submit the response to MM
4) MM processes trade MMrequests by executing trades. This processing includes determining the appropriate price and executing the trade on behalf of the trader.
5) MM posts the details of these trades to a public ledger which is called bulletin board. This action is on-chain and ensures transparency and accountability by making the trading activities publicly accessible.
6) The trader can submit any complaint if they find any missing or invalid ticket to a specific smart contract. The contract handles complaints and enforces accountability by locking collateral and rewarding traders on valid complaints.

VIII. PROPOSER-BUILDER SEPARATION

The primary objective of Proposer-Builder Separation (PBS) is to foster greater diversity among participants in blockchain networks by separating the roles of block proposers from those who actually create the blocks. This separation is

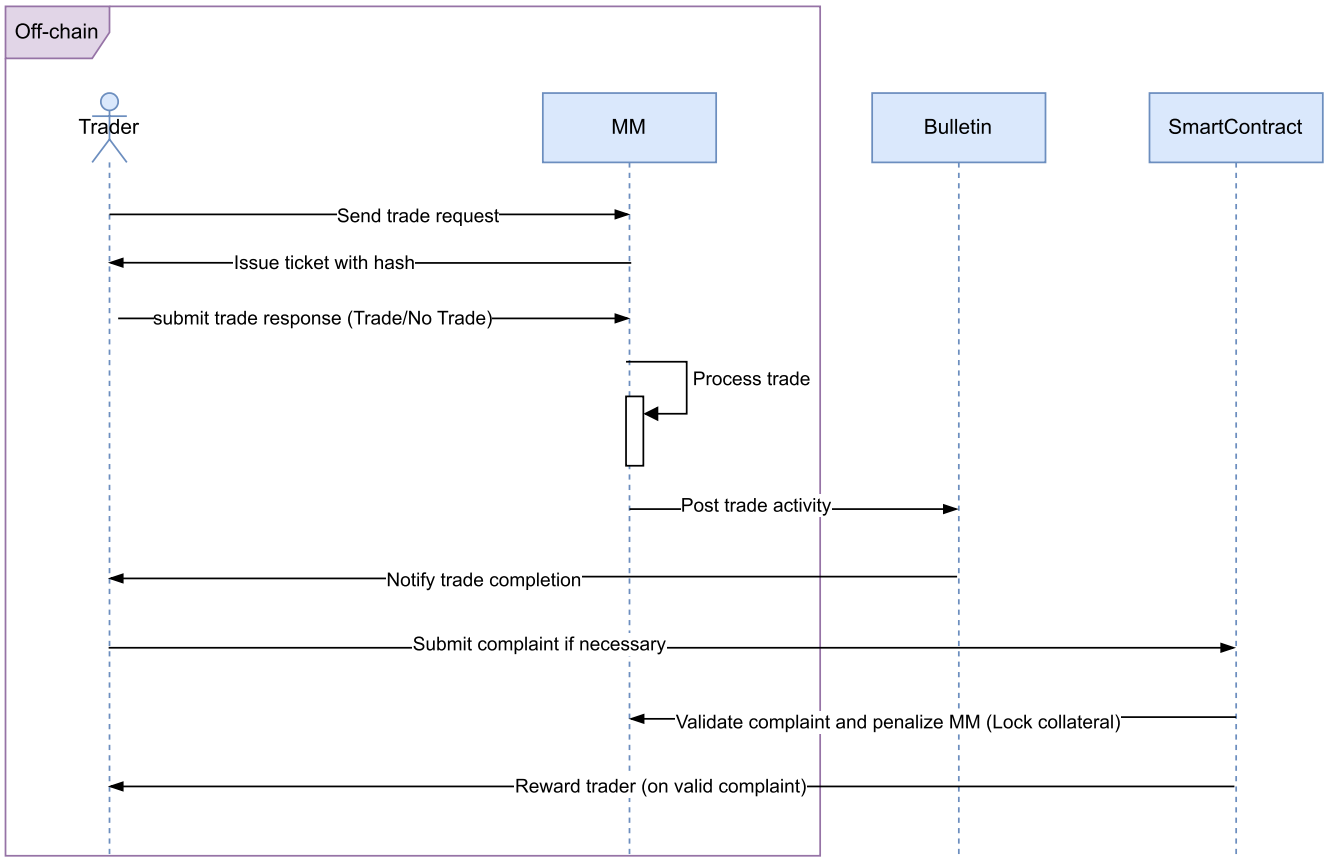


FIGURE 11. Fairmm exchange protocol sequential diagram.

intended to decentralize power and enhance network security by distributing responsibilities more widely among different parties.

PBS can be implemented entirely on-chain, requiring changes to Ethereum's consensus layer, which have not yet been implemented. Currently, MEV-Boost [49], an off-chain implementation of PBS, is widely used, creating about 90% of Ethereum's blocks under the Proof of Stake (PoS) consensus mechanism. This implementation specifically aims to mitigate issues related to MEV, promoting a more equitable and efficient transaction validation process by allowing separate entities to propose and construct blocks. The main idea is that instead of the block proposer trying to produce a revenue-maximizing block by themselves, they rely on an off-chain market where outside actors, called block-builders, produce bundles consisting of complete block contents and a fee for the proposer. The proposer chooses the bundle with the highest fee. MEV-Boost allows validators to benefit from MEV without directly involving themselves in the MEV extraction process; this ensures a fairer distribution of rewards [47], [50]. The process of MEVBoost is outlined in the following steps.

- MEV searchers create a bundle of their profitable transactions from the Ethereum mempool.
- MEV searchers send the bundles to a network of block builders in conjunction with a price bid to express their preferred position in a block.
- Block builders try to create the most profitable block with these bundles. The profitability of a block comes from both the MEV extracted from the transactions and the transaction fees. Builders are incentivized to include transactions that offer the highest MEV and fees.
- After constructing the blocks, builders submit them to a relay, which checks their validity and calculates the total profit. This ensures that only valid and profitable blocks are added to the Ethereum blockchain. However, there is a risk of malicious behaviors from the relay, including potential block censoring or front-running.
- Block proposers or validators on the Ethereum network use MEV-Boost to request blocks from the relay; it selects and sends the most profitable block header (without payload) back to the validator.
- Once the proposer signs the block header, the payload is released to the proposer.

Since the block proposer signs the block header without knowing the payload or transaction content, they are unable to engage in malicious activities such as front-running or censoring transactions. Additionally, if a block proposer attempts to propose another block other than the one they

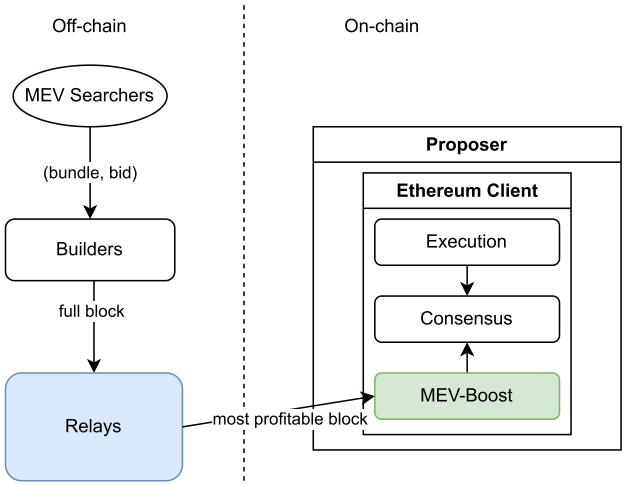


FIGURE 12. MEV-Boost architecture.

signed previously, they risk being penalized by the network due to the possibility of double signing.

Relays pose a centralization risk, as they have the potential for malicious actions such as censoring blocks or engaging in front-running activities. Currently there are a few relay providers that handles 90% of the block production on Ethereum PoS [51]. Providing incentives for running relay nodes can help prevent centralization. This could involve creating a token economy or other forms of compensation that encourage participation and discourage monopolization of relay services. Establishing regular monitoring and auditing procedures for relay nodes can help detect and respond to attempts at centralization or abuse of power. This could involve both automated systems and manual reviews to ensure that relay nodes are operating in accordance with the principles of PBS [52].

IX. SUMMARY

Table 5 provides an overview of various approaches aimed at mitigating the MEV extraction issue. The majority of these solutions fall under the prevention category, focusing on either reducing or eliminating the opportunities for MEV exploitation. Among the solutions, PBS stands out as the only method designed to minimize the negative impacts by making MEV extraction more democratic.

Implementing PBS through MEV-Boost does not add any latency to the network, as it operates off-chain primarily through relays. This ensures seamless operation without imposing additional load on the network. The communication overhead between sequencer nodes can lead to increased latency, especially in decentralized environments where all nodes must agree on a transaction order as defined by the network's ordering policy. This requires greater coordination between nodes. Additionally, the computational load grows as more sequencers participate in the ordering process. Privacy-preserving approaches generally involve encrypting the mempool, which can increase latency due to the encryption and decryption processes.

The proposed mitigation strategies are generally applicable across various transaction types and are not limited to specific applications. However, the smart contract-level protection approach requires modifications to the application's design, making it impractical for already deployed smart contracts.

Both the privacy-preserving and PBS methods offer an additional advantage in terms of censorship resistance. Since validators do not have access to transaction details prior to their inclusion in a block, they lack any incentive to censor specific transactions. This enhances the fairness and integrity of transaction processing. The other two strategies, smart contract-level protection and fair ordering, cannot prevent censorship by validators on their own. They need to incorporate additional methods, such as zero-knowledge proofs or slashing mechanisms, to effectively deter censorship.

Each method carries its own centralization risks. Fair ordering approaches might involve either a single sequencer or a permissioned group of sequencers. Privacy-preserving methods could encounter issues with a key management committee or dependencies on hardware devices. For PBS, the reliance on relay nodes introduces a potential point of centralization.

The fair ordering approach can be used on L2 chains without changing the Ethereum protocol. Privacy-preserving methods and PBS may need changes to the core protocol. This is particularly true if validators are involved in a key holder committee or if PBS is integrated directly into the network's infrastructure.

While fair ordering policies can significantly mitigate MEV, they add complexity to the network protocol and require trust in the sequencers. These policies cannot entirely eliminate censorship, as sequencers might not prioritize transactions with high gas fees. Therefore, substantial incentives are needed to ensure that sequencers act honestly.

Rollup chains are moving towards decentralization, with some proposing transitioning to a decentralized network of sequencers. However, this transition is costly for each rollup chain, and with the emergence of new rollup chains, incentivizing node operators to join each network becomes challenging. This leads to resource fragmentation. A promising solution to these issues is the concept of shared sequencing. This approach envisions a decentralized network of sequencers that can be used collectively by various rollup chains. Shared sequencing not only reduces the individual costs for each rollup chain but also streamlines resource allocation, thereby preventing resource fragmentation and improving overall efficiency. By pooling resources and creating a unified network of sequencers, shared sequencing enhances the viability and scalability of decentralized rollup chains. A potential area for future research on shared sequencing is to develop mechanisms that prevent cross-MEV attacks. Preventing these attacks would require designing advanced protocols and robust

TABLE 5. A High-level comparison between different MEV mitigation strategies.

MEV Mitigation Strategies	Strategy	Latency	Application Dependent	Censorship Resistance	Centralization Risk	L1 Protocol Change
Fair ordering	Prevention	Communication cost	No	No	Sequencer nodes	No
Smart contract-level protection	Prevention	Additional computation	Yes	No	—	No
Privacy preserving	Prevention	Encryption/ Decryption	No	Yes	Key management Committee/ Hardware	No/Yes
PBS	Democratizing	—	No	Yes	Relay nodes	No/Yes

security measures that ensure sequencers cannot manipulate or reorder transactions across chains for profit. This could involve implementing cross-chain communication standards, enforcing stricter ordering policies, or integrating with privacy-preserving mechanisms.

Privacy-preserving methods, particularly those that enhance mempool privacy, show great promise in preventing front-running and ensuring censorship resistance. These methods can also be integrated with other strategies to mitigate MEV. Among these, delay encryption stands out as especially promising due to its trustless nature, which eliminates the need for a key management committee. Future work on delay encryption involves making time-lock puzzles more secure, reducing the computational load on users, and exploring the use of asymmetric or public key cryptography. This will help improve the efficiency and usability of delay encryption, making it a more viable solution for enhancing privacy and security in blockchain networks.

Witness encryption [53] is a cryptographic technique designed to enhance mempool privacy by encrypting transactions such that they can only be decrypted when a specific computational proof (the witness) is provided. This ensures that transaction details remain hidden from validators until certain conditions, such as block confirmation or inclusion in a block, are satisfied. Unlike delay encryption, which is time-bound, witness encryption supports a wider variety of decryption conditions, making it applicable to a broader range of scenarios. Although witness encryption offers promising potential, it is still a relatively new concept in cryptography, and practical implementations are in the early stages. While the theory behind it is promising, creating efficient, scalable implementations of witness encryption has proven to be challenging due to the complexity of constructing the necessary cryptographic proofs and the associated computational costs.

There appears to be a strong incentive for validators to gain profit from MEV blocks, as only 10% of blocks are built by validators not using MEV-Boost. Therefore, it is important to focus on addressing the challenges associated with the MEV-Boost solution, particularly by working towards the decentralization of relay networks and implementing incentives or slashing mechanisms to encourage honest behavior among relays.

X. CONCLUSION

In conclusion, our research has yielded valuable insights into various MEV mitigation strategies applicable to Ethereum and L2 chains. As highlighted throughout the article, each approach presents its own set of challenges and advantages. Some methods offer security and efficiency but entail trust-related issues, while others aim to offload certain processes off-chain to reduce protocol overhead or mitigate MEV extraction side effects.

Notably, mempool privacy utilizing Delay Encryption and PBS emerges as promising approaches. Ongoing research in areas like shared sequencing and witness encryption shows considerable potential. Further exploration into these topics is warranted, as new challenges may arise, necessitating continued investigation into the advancements made in these domains.

REFERENCES

[1] P. Daian, S. Goldfeder, T. Kell, Y. Li, X. Zhao, I. Bentov, L. Breidenbach, and A. Juels, "Flash boys 2.0: Frontrunning in decentralized exchanges, miner extractable value, and consensus instability," in *Proc. IEEE Symp. Secur. Privacy (SP)*, May 2020, pp. 910–927.

[2] S. Eskandari, S. Moosavi, and J. Clark, "SoK: Transparent dishonesty: front-running attacks on blockchain," in *Proc. Int. Conf. Financial Cryptography Data Secur.*, Feb. 2019, pp. 170–189.

[3] G. Konstantopoulos. (2021). *(Almost) Everything You Need to Know About Optimistic Rollup*. [Online]. Available: https://www.paradigm.xyz/2021/01/almost-everything-you-need-to-know-about-optimistic-rollup

[4] T. Schaffner and F. Schaer, "Scaling public blockchains—A comprehensive analysis of optimistic and zero-knowledge rollups," in *A Comprehensive Analysis of Optimistic and Zero-Knowledge Roll-Ups*. Basel, Switzerland: Univ. Basel, 2021.

[5] A. Gangwal, H. R. Gangavalli, and A. Thirupathi, "A survey of layer-two blockchain protocols," *J. Netw. Comput. Appl.*, vol. 209, Jan. 2023, Art. no. 103539.
[6] L. T. Thibault, T. Sarry, and A. S. Hafid, "Blockchain scaling using rollups: A comprehensive survey," *IEEE Access*, vol. 10, pp. 93039–93054, 2022.
[7] A. Mamageishvili, M. Kelkar, J. C. Schlegel, and E. W. Felten, "Buying time: Latency racing vs. bidding for transaction ordering," in *Proc. 5th Conf. Adv. Financial Technol. (AFT)*, 2023, pp. 23:1–23:22.
[8] D. Yakira, A. Asayag, G. Cohen, I. Grayevsky, M. Leshkowitz, O. Rottenstreich, and R. Tamari, "Helix: A fair blockchain consensus protocol resistant to ordering manipulation," *IEEE Trans. Netw. Service Manage.*, vol. 18, no. 2, pp. 1584–1597, Jun. 2021.
[9] A. Kavousi, D. V. Le, P. Jovanovic, and G. Danezis, "BlindPerm: Efficient MEV mitigation with an encrypted mempool and permutation," Cryptol. ePrint Arch., Tech. Paper 2023/1061, 2023. [Online]. Available: https://eprint.iacr.org/2023/1061
[10] M. Ciampi, M. Ishaq, M. Magdon-Ismail, R. Ostrovsky, and V. Zikas, "FairMM: A fast and frontrunning-resistant crypto market-maker," in *Proc. Int. Symp. Cyber Secur., Cryptol., Mach. Learn.*, Jan. 2022, pp. 428–446.
[11] J. Xu, K. Paruch, S. Cousaert, and Y. Feng, "SoK: Decentralized exchanges (DEX) with automated market maker (AMM) protocols," *ACM Comput. Surveys*, vol. 55, no. 11, pp. 1–50, Nov. 2023.
[12] H. Adams, N. Zinsmeister, M. Salem, R. Keefer, and D. Robinson, "Uniswap v3 core," Uniswap, USA, Tech. Rep., 2021.
[13] M. Egorov and C. Finance, "Automatic market-making with dynamic PEG," Retrieved, Curve Finance (Swiss Stake GmbH), Switzerland, Dec. 2021.
[14] F. Martinelli and N. Mushegian. (2019). *A Non-custodial Portfolio Manager, Liquidity Provider, and Price Sensor*. [Online]. Available: https://balancer.finance/whitepaper
[15] DODO. *What is DODO*. Accessed: Jul. 8, 2024. [Online]. Available: https://docs.dodoex.io/en/home/what-is-dodo
[16] T. Alam, M. A. Ali, and M. H. Rahman, "Front-running attack in decentralized finance in the metaverse: A systematic review," *Int. J. Sci. Res. Arch.*, vol. 11, no. 1, pp. 2315–2324, Feb. 2024.
[17] W. Zhang, L. Wei, S.-C. Cheung, Y. Liu, S. Li, L. Liu, and M. R. Lyu, "Combatting front-running in smart contracts: Attack mining, benchmark construction and vulnerability detector evaluation," *IEEE Trans. Softw. Eng.*, vol. 49, no. 6, pp. 3630–3646, Jun. 2023.
[18] L. Heimbach and R. Wattenhofer, "SoK: Preventing transaction reordering manipulations in decentralized finance," in *Proc. 4th ACM Conf. Adv. Financial Technol.*, Sep. 2022, pp. 47–60.
[19] P. McCorry, C. Buckland, B. Yee, and D. Song, "SoK: Validating bridges as a scaling solution for blockchains," Cryptol. ePrint Arch., Tech. Paper 2021/1589, 2021. [Online]. Available: https://eprint.iacr.org/2021/1589
[20] L. Bousfield, R. Bousfield, C. Buckland, B. Burgess, J. Colvin, E. W. Felten, S. Goldfeder, D. Goldman, B. Huddleston, and H. Kalodner, "Arbitrum nitro: A second-generation optimistic rollup," Offchain Labs, Inc., NJ, USA, Tech. Rep., 2018.
[21] Metis. (2024). *Decentralized Sequencer: Sequencer Node*. Accessed: Apr. 26, 2024. [Online]. Available: https://docs.metis.io/dev/decentralized-sequencer/overview/sequencer-node
[22] M. Kelkar, S. Deb, S. Long, A. Juels, and S. Kannan, "Themis: Fast, strong order-fairness in Byzantine consensus," in *Proc. ACM SIGSAC Conf. Comput. Commun. Secur.*, 2023, pp. 475–489.
[23] K. Kursawe, "Wendy grows up: More order fairness," in *Proc. Int. Conf. Financial Cryptography Data Secur.*, Jan. 2021, pp. 191–196.
[24] E. Systems. (2022). *Decentralizing Rollups: Announcing the Espresso Sequencer*. Accessed: Jul. 2, 2024. [Online]. Available: https://medium.com/@espressosys/decentralizing-rollups-announcing-the-espresso-sequencer-81c4c7ef6d97
[25] S. Network. (2024). *Introducing Shutter—Combating Front Running and Malicious Mev Using Threshold Cryptography*. Accessed: Jun. 1, 2024. [Online]. Available: https://blog.shutter.network/introducing-shutter-network-combating-frontrunning-and-malicious-mev-using-threshold-cryptography/
[26] H. Zhang, L.-H. Merino, Z. Qu, M. Bastankhah, V. Estrada-Galinanes, and B. Ford, "F3B: A low-overhead blockchain architecture with per-transaction front-running protection," 2022, *arXiv:2205.08529*.
[27] A. Park. (2024). *Decentralizing Rollup Sequencers: Towards a Rollup-Centric Ethereum*. Accessed: Apr. 21, 2024. [Online]. Available: https://hackmd.io/@zeroknight/radiusdecentralizingrollupsequencers#Order-Validation
[28] Z. Shi, S. Sweck, and O. Zaki, "From CoWs to multi-chain AMMs: A strategic optimization model for enhancing solvers," in *Proc. IEEE Int. Conf. Blockchain (Blockchain)*, Aug. 2024, pp. 97–104.
[29] A. Wahrstätter, L. Zhou, K. Qin, D. Svetinovic, and A. Gervais, "Time to bribe: Measuring block construction market," 2023, *arXiv:2305.16468*.
[30] Metis. (2024). *Governance Proposal: Decentralized Sequencer Governance*. Accessed: Apr. 26, 2024. [Online]. Available: https://ceg.vote/t/governance-proposal-decentralized-sequencer-governance/1922
[31] K. Kursawe, "Wendy, the good little fairness widget: Achieving order fairness for blockchains," in *Proc. 2nd ACM Conf. Adv. Financial Technol.*, Oct. 2020, pp. 25–36.
[32] M. Kelkar, F. Zhang, S. Goldfeder, and A. Juels, "Order-fairness for Byzantine consensus," in *Proc. 40th Annu. Int. Cryptol. Conf.*, Santa Barbara, CA, USA. Cham, Switzerland: Springer, 2020, pp. 451–480.
[33] Y. Zhang, S. Setty, Q. Chen, L. Zhou, and L. Alvisi, "Byzantine ordered consensus without Byzantine oligarchy," in *Proc. 14th USENIX Symp. Operating Syst. Design Implement. (OSDI 20)*, Oct. 2020, pp. 633–649.
[34] E. Systems. (2024). *Espresso Architecture: System Overview*. Accessed: Jun. 5, 2024. [Online]. Available: https://docs.espressosys.com/sequencer/espresso-architecture/system-overview
[35] J. Charbonneau. (2023). *Encrypted Mempools*. Accessed: Apr. 21, 2024. [Online]. Available: https://joncharbonneau.substack.com/p/encrypted-mempools
[36] X. Lyu, M. Zhang, X. Zhang, J. Niu, Y. Zhang, and Z. Lin, "An empirical study on Ethereum private transactions and the security implications," 2022, *arXiv:2208.02858*.
[37] A. Rondelet and Q. Kilbourn, "Mempool privacy: An economic perspective," 2023, *arXiv:2307.10878*.
[38] J. Canny and S. Sorkin, "Practical large-scale distributed key generation," in *Proc. Int. Conf. Theory Appl. Cryptograph. Techn.*, Jan. 2004, pp. 138–152.
[39] J. Bebel and D. Ojha, "Ferveo: Threshold decryption for mempool privacy in BFT networks," *Cryptol. ePrint Arch.*, 2022.
[40] A. R. Choudhuri, S. Garg, J. Piet, and G.-V. Policharla, "Mempool privacy via batched threshold encryption: Attacks and defenses," *Cryptol. ePrint Arch.*, 2024.
[41] S. Micali, M. Rabin, and S. Vadhan, "Verifiable random functions," in *Proc. 40th Annu. Symp. Found. Comput. Sci.*, Oct. 1999, pp. 120–130.
[42] J. Burdges and L. D. Feo, "Delay encryption," in *Proc. Annu. Int. Conf. Theory Appl. Cryptograph. Techn.*, Jan. 2021, pp. 302–326.
[43] J. Liu, T. Jager, S. A. Kakvi, and B. Warinschi, "How to build time-lock encryption," *Des., Codes Cryptogr.*, vol. 86, no. 11, pp. 2549–2586, Nov. 2018.
[44] Ethresearch. (2024). *Mev-Resistant ZK-Rollups With Practical VDE/PVDE*. Accessed: Jun. 25, 2024. [Online]. Available: https://ethresear.ch/t/mev-resistant-zk-rollups-with-practical-vde-pvde/12677
[45] F. McKeen, I. Alexandrovich, I. Anati, D. Caspi, S. Johnson, R. Leslie-Hurd, and C. Rozas, "Intel software guard extensions (Intel SGX) support for dynamic memory management inside an enclave," in *Proc. Hardw. Architectural Support Secur. Privacy*, 2016, pp. 1–9.
[46] Flashbots. (2024). *The Future of Mev is Suave*. Accessed: Aug. 2, 2024. [Online]. Available: https://writings.flashbots.net/the-future-of-mev-is-suave#iv-suave-in-the-blockchain-stack
[47] B. Öz, D. Sui, T. Thiery, and F. Matthes, "Who wins Ethereum block building auctions and why?" 2024, *arXiv:2407.13931*.
[48] Shoal. (2024). *Mev Protection: Dex and Aggregator Anti-Mev Mechanisms*. Accessed: Apr. 23, 2024. [Online]. Available: https://www.shoal.gg/p/mev-protection-dex-and-aggregator
[49] Flashbots. *Mev-boost*. Accessed: Jul. 19, 2024. [Online]. Available: https://mevboost.pics/
[50] A. Unger, "From searchers to proposers: User behaviour in Ethereum's proposer-builder separation," M.S. thesis, Distributed Comput. Group Comput. Eng. Netw. Lab. ETH Zürich, 2023.
[51] R. Labs. (2024). *Rated Network Explorer—Relays*. Accessed: Apr. 22, 2024. [Online]. Available: https://explorer.rated.network/relays?network=mainnet&timeWindow=all
[52] EigenPhi. (2024). *From Censorship to Centralization: 4 Dimensions of Ethereum Relay's Public Goods Problem*. Accessed: Jul. 20, 2024. [Online]. Available: https://eigenphi.substack.com/p/eth-relay-public-goods-problem

[53] J. Stearn, "Cryptographic approaches to complete mempool privacy," Flashbots Res., Cayman Islands, Tech. Rep. FRP-18, 2022.

ZEINAB ALIPANAHLOO received the bachelor's degree in software engineering from IKIU, Iran, the master's degree in artificial intelligence from AUT, Iran. She is currently pursuing the Ph.D. degree with ÉTS University, Montreal, Canada. Her current research interests include decentralized technologies, blockchain, cryptography, and security.

ABDELHAKIM SENHAJI HAFID received the M.S. and Ph.D. degrees in computer science. He spent several years as a Senior Research Scientist with Bell Communications Research (Bellcore), Piscataway, NJ, USA, working in the context of major research projects on the management of next-generation networks. He was also an Assistant Professor with Western University (WU), London, ON, Canada, the Research Director of the Advance Communication Engineering Center (venture established by WU, Bell Canada, and Bay Networks), London, a Researcher with CRIM, Montreal, QC, Canada, a Visiting Scientist with GMD-Fokus, Berlin, Germany, and a Visiting Professor with the University of Évry, Évry-Courcouronnes, France. He is currently a Full Professor with the University of Montreal, Montreal. He is also the Founding Director of the Network Research Laboratory, Montreal, and the Montreal Blockchain Laboratory, Montreal. He is also a Research Fellow with CIRRELT, Montreal. He co-founded Tipot Technologies Inc., Ottawa, ON, Canada (research and development platform for IoT). He consulted for a number of telecommunication companies and startups in North America. He has extensive academic and industrial research experience in the area of the management and design of next-generation networks. He has supervised to graduation over 50 graduate and postgraduate students. He has authored or co-authored over 250 journals and conference papers. He also holds three U.S. patents. His current research interests include the IoT, fog/edge computing, blockchain, and intelligent transport systems. He also gave talks/keynotes at a number of international conferences.

KAIWEN ZHANG (Member, IEEE) received the B.Sc. and M.Sc. degrees from McGill University, Montreal, and the Ph.D. degree from the University of Toronto. He was an Alexander von Humboldt Postdoctoral Fellow in computer science with TU Munich. He is currently a Professor with the Department of Software and IT Engineering, ÉTS. His research interests include blockchain technologies, publish/subscribe systems, massive multiplayer online games, performance modeling, and software-defined networking.