


RESEARCH

Open Access



Mitigating MEV attacks with a two-tiered architecture utilizing verifiable decryption

Mustafa Ibrahim Alnajjar^{1*} , Mehmet Sabir Kiraz¹, Ali Al-Bayatti¹ and Suleyman Kardas²

*Correspondence:
mustafa.alnajjar@my365.dmu.
ac.uk

¹Cyber Technology Institute, De
Montfort University, Leicester, UK

²Computer Engineering, Batman
University, Batman, Turkey

Abstract

A distributed ledger is a shared and synchronized database across multiple designated nodes, often referred to as miners, validators, or peers. These nodes record, distribute, and access data to ensure security and transparency. However, these nodes can be compromised and manipulated by selectively choosing which user transactions to include, exclude, or reorder, thereby gaining an unfair advantage. This is known as a miner/maximal extractable value (MEV) attack. Existing solutions can be classified into various categories, such as MEV auction platforms and time-based ordering properties, which rely on private transaction Mempools. In this paper, we first identify some architectural weaknesses inherent in the latest proposals that divide the block creation and execution roles into separate functions: block builders and block executors. The existing schemes mainly suffer from the verifiability of the decryption process, where a corrupted builder or executor can simply deny the inclusion of specific targeted transactions by exploiting the fact that all transactions are in plain format. To address this, we propose an enhanced version that incorporates a verifiable decryption process. On a very high level, within our proposal, whenever an Executor or a Builder performs a decryption, the decrypted values must be broadcasted. This enables any entity in the network to publicly verify whether the decryption was executed correctly, thus preventing malicious behavior by either party from going undetected. We also define a new adversary model for MEV and conduct a comprehensive security analysis of our protocol against all kinds of potential adversaries related to MEV. Finally, we present the performance analysis of the proposed solution.

Keywords: Blockchain, Miner/maximal extractable value (MEV), Cryptographic, Frontrunning, Backrunning

1 Introduction

The introduction of blockchain technology, shown by Bitcoin [1] and Ethereum [2], has started in a new era of financial transactions. This technology allows users to send transactions to a decentralized network of peers, known as miners or validators, who try to earn transaction fees and block rewards [3]. The core of blockchain is its consensus mechanisms, which ensure the process of validating transactions is honest and fair. These mechanisms minimize the chance of any single miner benefiting too much from the fees compared to the resources they put into to the network. Despite the robustness of these consensus algorithms, they typically do not dictate the precise ordering of

transactions within a block. This flexibility stems from the practical challenges in ensuring that validators generate both precise and manipulation-resistant timestamps. Consequently, individual validators may be incentivized to sequence transactions in a manner that maximizes their profits, a phenomenon that is not fully mitigated by the trustless model of blockchain or by performance considerations. This issue persists even with the advent of Layer 2 solutions, such as Optimistic Rollups [4], Zero-Knowledge Rollups (zkRollups) [5], and a Zero-Knowledge Ethereum Virtual Machine (zkEVM) [6] are also subjected to such attacks which are also susceptible to similar vulnerabilities. A key result of this flexibility is the development of MEV, which is the extra profit a miner can make by smartly changing the order of user transactions [3, 7].

The ecosystem of MEV involves three main actors: miners, network users, and MEV searchers. Miners validate transactions in exchange for fees, network users submit transactions for inclusion in the blockchain, and MEV searchers identify profitable opportunities by manipulating the order of transactions in their mempool. These searchers employ various strategies to optimize the sequence of transactions within the constraints of a block's maximum capacity. When a profitable sequence is identified, they participate in auctions organized by miners to secure specific transaction slots. This situation has transformed decentralized finance (DeFi) into a competitive where a few players can gain a significant value at the expense of less knowledgeable participants. The challenge of MEV has prompted a lot of research efforts aimed at finding ways to reduce its impact. For example, an architectural solution, called the Flashbots Auction [8], introduces a structured network of searchers, relays, and builders, to protocol-level innovations like the Eden Network's [9] transaction ordering protocol and Ethereum's Danksharding scheme [10]. However, despite these advancements, existing solutions have not fully addressed the threat posed by malicious block proposers or builders.

1.1 MEV attacks: impact and motivation of this study

A new issue called MEV has emerged as a significant concern for blockchain technology, especially affecting cryptocurrencies like Bitcoin and Ethereum. MEV attacks exploit the inherent flexibility in transaction ordering within blockchain systems, allowing miners (or validators) to potentially change the order of transactions to give themselves an unfair advantage. This capability not only poses a risk to the fairness and transparency of blockchain operations but also has broader implications for market stability and security. That's why it's important to come up with ways to stop MEV attacks. This will protect the core ideas of blockchain and keep the cryptocurrency market healthy.

Firstly, the potential for MEV attacks to disrupt market equilibrium cannot be overstated. By enabling actors to alter transaction outcomes and manipulate prices, these attacks introduce a significant source of market instability. Addressing MEV attacks is crucial for preserving the integrity of cryptocurrency markets, ensuring that they function as efficient and fair platforms for financial exchange. Implementing robust solutions to counteract MEV attacks can significantly contribute to maintaining market stability, thereby protecting the interests of all market participants from dishonest behavior and manipulative practices.

Furthermore, MEV attacks contribute to volatility in the cryptocurrency market by allowing for the strategic manipulation of transaction outcomes and pricing [11]. This

volatility undermines the broader adoption and acceptance of cryptocurrencies as stable financial instruments. By developing and deploying mechanisms to mitigate the impact of MEV attacks, we can foster a more stable and predictable market environment, encouraging greater participation and investment in the cryptocurrency space.

Another critical concern is the effect of MEV attacks on the security of the blockchain itself. These attacks can incentivize miners to engage in practices such as transaction restriction or reshuffling, directly undermining the blockchain's security framework and the immutability of transaction records. By addressing the root causes and mechanisms of MEV attacks, we can bolster the blockchain's resilience against such security threats, ensuring the protection and integrity of user transactions.

Lastly, the presence of MEV attacks introduces inefficiencies into the blockchain ecosystem, as users may be forced to adopt expensive and complex strategies to safeguard their transactions from being front-run or altered. This not only increases the operational costs for users but also detracts from the blockchain's promise of providing a cost-efficient and transparent mechanism for conducting transactions. Developing solutions to neutralize the threat of MEV attacks can significantly enhance the efficiency of blockchain systems, reducing the necessity for costly countermeasures and improving the overall user experience.

In light of these considerations, this research paper seeks to explore innovative approaches to mitigating the challenges posed by MEV attacks. By examining the underlying mechanisms that enable these attacks and proposing effective solutions, we aim to contribute to the ongoing efforts to secure blockchain networks, preserve market stability, and ensure a fair and equitable digital economy.

1.2 Our contributions

In this paper, we first evaluate current mitigation strategies against MEV attacks and their effects on blockchain security and decentralization. We demonstrate that these techniques do not actually offer full protection against malicious block proposers or builders. To address this gap, we introduce a new mechanism to enhance a recently proposed scheme, called Mangata's Proposer-Builder Separation (PBS) [12], which was designed to resist MEV attacks. Our solution addresses the architectural weaknesses of existing approaches by employing a verifiable decryption mechanism. In particular, we reduce the block proposer's power to reject an auctioned block. Furthermore, we also present a comprehensive new adversary model for MEV attacks and prove that our proposed architecture is indeed secure against all types of attackers defined within this model. Through our research, we aim to contribute to ongoing efforts toward securing blockchain networks against MEV vulnerabilities and promoting a more equitable and decentralized ecosystem. The contributions of our paper can be summarized as follows:

- We first revisit existing proposals to mitigate MEV attacks, particularly focusing on the recently proposed Danksharding [10]. This approach introduces sharding, dividing the network into smaller sections, and implements PBS and Censorship Resistance Scenarios (crList) to reduce MEV. However, we show that in its basic form, this design does not prevent a proposer from ignoring the auctioned block from the builder and creating their own blocks, hence potentially enabling a malicious pro-

poser to exploit MEV. We also review various strategies that have been proposed to secure MEV attacks, including Flashbots [8], Mangata [12], and Eden Network [9] and show that these solutions do not actually provide full security against malicious block proposers or block builders.

- We then introduce our improved scheme to enhance Mangata's PBS-based MEV resistance protocol using a verifiable decryption process, which mitigates the ability of block proposers to decline auctioned blocks.
- We also present a new adversary model for MEV attacks and show that the proposed architecture is secure against all the forms of attackers defined in the model. We believe this to be the first adversary model presented in the space MEV.
- Finally, we evaluate the performance of our proposal by presenting our implementation results and showing its efficiency and scalability.

1.3 Roadmap

In Section 1, we detail the design and implementation of our computational simulations, including the specific cryptographic operations, experimental setup, and the comprehensive benchmarking of our proposed two-tiered architecture aimed at mitigating MEV attacks. In Sect. 2, we first highlight the methods used in the construction of the proposal, gives the most common consensus algorithms behind PoA chains which are used in the proposed constructions for securing MEV attacks, and then provide the necessary cryptographic background for MEV mitigation techniques. In Sect. 3, we present and discuss previous attempts and proposals aimed at mitigating MEV attacks. Section 4 presents the definition of MEV and the related attacks such as frontrunning, back-running, and sandwich. It also introduces several different strategies that can be used to mitigate the MEV attacks. It also reviews the related literature on MEV attacks through a number of mechanisms. Section 5 first describes the Mangata Finance protocol and then presents its potential weaknesses. In Sect. 6, we propose our new scheme which is potentially an improved version of the Mangata architecture by providing a verifiable decryption process to mitigate their weaknesses. In Sect. 7, we present a detailed security analysis according to the predefined adversarial model and give the performance analysis. In Sect. 8, we present the results evaluating the performance and efficiency of our proposal. In Sect. 9, it compares the security and transparency of the proposed architecture with existing methods for mitigating MEV attacks, highlighting scalability challenges and the need for ongoing testing and optimization. Finally, Sect. 10 concludes the paper.

2 Methods/experimental

2.1 Study design

The study employs a detailed computational simulation to evaluate cryptographic operations aimed at mitigating MEV within a blockchain environment. Our model simulates real-world operations using RSA and symmetric cryptographic algorithms under a proposed two-tiered architectural framework, focusing on their effectiveness and efficiency in reducing potential MEV attacks.

2.2 Setting

The research was conducted using Python 3.12.2 on an Intel(R) Core(TM) i5-10210U CPU. This setup provided a controlled and replicable environment, ensuring that the performance metrics and benchmarks accurately reflect the computational demands of typical blockchain operations. All cryptographic operations and performance evaluations were performed within this environment, allowing for precise control over variables and consistency in data collection.

2.3 Participants or materials

Our simulations utilized artificial blockchain transactions that were created to represent a range of typical network activities. These transactions were used to test the integrity and performance of RSA decryption, symmetric decryption, and their combination within our blockchain architecture. The GitHub repository¹ publicly hosts the implemented cryptographic algorithms, allowing for external validation and replication of our results.

2.4 Interventions and comparisons

This subsection outlines the cryptographic operations at each stage of our two-tiered architecture. It highlights how these operations improve security and efficiency compared to traditional cryptographic methods.

Client-side calculations

- Transactions were initially signed with the user's private key.
- Two types of encryptions were performed per transaction: RSA encryption for keys designated for the builder and executor and symmetric encryption for the transaction data itself.

Builder-side calculations

- Transactions were selected from a simulated mempool, involving decryption processes to retrieve keys and messages and verify the transaction integrity through hash comparisons.

Executor-side calculations

- Post builder decryption, the executor verified the block's integrity, performed further decryptions, and prepared the transaction for final block inclusion.

These interventions were systematically compared to baseline models that utilize conventional single-tier cryptographic practices, highlighting the enhancements in security and efficiency brought about by our proposed model.

¹ <https://github.com/Mustafa25022022/Mitigating-MEV-Attacks-with-a-Two-Tiered-Architecture-Utilizing-Verifiable-Decryption.git>

2.5 Type of analysis used

Comprehensive statistical analysis was utilized to assess the performance benchmarks. Metrics such as mean, standard deviation, and range were calculated for:

- Encryption and decryption times, capturing the speed and efficiency of cryptographic operations.
- Key generation and integrity check durations, emphasizing the system's capability to maintain security under operational stress.

Detailed results included:

- Average encryption time of 0.002555 s and decryption time of 0.006267 s over 1,000 trials, indicating high efficiency suitable for real-time applications.
- Key generation and integrity checks demonstrated rapid execution, essential for maintaining high-security standards in dynamic network environments.

2.6 Power calculation

While a traditional power calculation was not applicable due to the non-statistical nature of the primary outcomes (system performance metrics), the sample size of 1,000 trials was chosen to ensure the robustness and reliability of the benchmark results, providing a comprehensive view of system performance across various scenarios.

3 Exploring blockchain architecture and cryptographic primitives in our proposal

This paper proposes a new architecture to mitigate MEV attacks. It also analyses previously proposed architectures which were designed against MEV; such as Flashbots [8, 13], Mangata Finance [12], Eden Network [9], and Danksharding [10, 14]. However, we show that they do not actually offer sufficient protection against both adversarial block builders and executors. These existing architectures have also a common vulnerability that could allow a malicious proposer to reject an auctioned block from the builder and independently create their own blocks, thereby facilitating MEV extraction. In this respect, Danksharding introduces a blockchain framework characterized by the division of the network into smaller units referred to as shards.

The proposed architecture aims to mitigate MEV exploitation by utilizing the concept of PBS. The proposed architecture addresses the aforementioned issues by introducing novel and effective enhancements to Mangata's PBS-based MEV resistance protocol. The proposed architecture incorporates a verifiable decryption process to target inherent weaknesses in the PBS protocol. Unlike other solutions, it also eliminates the problem of block bidders rejecting blocks put up for auction. Furthermore, this paper first introduces adversarial model for the proposals and then presents a formal analysis model against MEV attacks for the first time. The security of our proposed system has been shown within the framework of this model. Finally, the paper includes performance calculations of the proposed system and highlights its scalability.

In the following subsections, we start by presenting the necessary cryptographic primitives which have been utilized in our proposed protocol.

3.1 Most common consensus algorithms behind PoA chains

Proof-of-Authority (PoA) is a permission consensus algorithm that provides a practical and effective solution for current blockchain systems, especially consortium blockchains. PoA relies on several authority nodes, called sealers, to carry out procedures that allow consensus to be reached. These eligible nodes have to check that blocks and transactions are correct. The design of a small committee enables fast confirmation of transactions and easy management of involved members. Thus, PoA is used as the underlying consensus algorithm for many blockchain projects. Ethereum is the most well-known application. It comes in two forms: 1) Aura (short for Authority Round) in Parity and 2) Clique in Geth [15].

3.1.1 Clique

The Ethereum client, written in GoLang and Geth, employs Clique as its PoA algorithm. Clique [16] is the PoA algorithm, which uses Geth [17], while the Ethereum client is written in GoLang. The method works in epochs, distinguished by a fixed sequence of committed blocks. A specific transition block is sent when a new epoch begins, and which defines the set [18]. While Aura uses UNIX time, Clique uses a formula that combines the block number and the number of authorities to calculate the current step and related leader. Most importantly, other authorities and the existing leader are permitted to suggest blockages during each phase. To prevent a single Byzantine authority from wreaking havoc on the network by imposing an excessive number of blocks, each authority is limited to proposing a block every $N/2 + 1$ blocks. As a result, no more than $N - (N/2 + 1)$ authorities are permitted to propose a block at any given time. Similar to before, authorities who act intentionally (for example, proposing a block when it is not permitted) can be voted out. Specifically, at each step, a vote against an authority can be cast, and if a majority is obtained, the authority is removed from the list of valid authorities.

3.1.2 AURA (authority round)

Mangata's protocol has been built into the substrate framework and will be joined as a parachain in the Polkadot network using the AURA consensus [12]. Patterns of Aura have a first-round where the current leader proposes a new block (block proposal). Aura needs a second round where the new block is accepted (block acceptance), but by contrast Clique does not. The protocol depends on the assumption of a synchronous network (UNIX time synchronization), and thus, there may be times when it does not work correctly because the network validators' clocks need to be in sync.

The network is assumed to be synchronous and all authorities to be synchronized to the same UNIX time [19]. The index s for each step is deterministically computed by each authority as $s = t/step_duration$, where step duration is a constant, thus determining the duration of a given step. The leader of a step s is the authority identified by the id $l = s \bmod N$.

3.1.3 BABE (Blind assignment for blockchain extension)

BABE is usually used in proof-of-stake blockchains because it allows for slot-based block authoring with a known set of validators. Slot assignment, unlike Aura, is based on a verifiable random function (VRF). For each epoch, each validator is given a weight. This epoch is divided into slots, and at each slot, the validator checks its VRF. It can write a block for each slot where the validator's VRF output is lower than its weight. Even when the network is working well, forks occur more frequently in BABE than in Aura because more than one validator might be able to make a block at the same time. The way substrate uses that BABE provides a backup plan for when no authorities are chosen for a given slot. With these secondary slot assignments, BABE can keep the time it takes to block the same [20].

3.2 Cryptographic hash functions

A hash function takes an input of any length and returns an output of a fixed size. The values that a hash function gives back are called hash values, hash codes, digests, or just hashes. Let

$$H : \{0, 1\}^* \rightarrow \{0, 1\}^k$$

be a hash function. H needs to satisfy the following properties:

- *Collision resistance*: It is computationally infeasible to find a collision, i.e., two distinct inputs that hash to the same result. More concretely, it is hard to find two inputs x, y with $x \neq y$ such that $H(x) = H(y)$.
- *Preimage resistance (one way)*: It is computationally infeasible to find any input which hashes to any pre-specified output. More concretely, if a hash function H produced a hash value z , it should be hard to find an input value x such that $z = H(x)$.
- *Second preimage resistance*: It is computationally infeasible to find any second input which has the same output as any specified input. More concretely, if a hash function H takes an input value x and comes up with a hash value $H(x)$, it should be hard to find any other input value y for which $H(y) = H(x)$ [21].

3.3 Symmetric encryption - secret key encryption

A symmetric key encryption scheme uses a single private key K to encrypt a given message M as $C = \text{SymEnc}(K, M)$ and also to decrypt C as $M = \text{SymDec}(K, C)$. AES is the most common algorithm is used in practice where different key sizes can be used such as AES-128, AES-192, or AES-256 [22].

3.4 Asymmetric encryption

An asymmetric encryption (also called public key cryptography) uses a pair of public and private key (pk, sk) that are mathematically linked to encrypt and decrypt

sensitive information, respectively [23]. Rivest–Shamir–Adleman(RSA) and ElGamal are the most common algorithms used in practice.

3.4.1 RSA

- *Key generation*

1. Generate two random large primes p and q .
2. Compute $n = pq$.
3. Compute $\phi(n) = (p-1)(q-1)$.
4. Select a public exponent $e \in \{1, 2, \dots, \phi(n) - 1\}$ such that $\gcd(e, \phi(n)) = 1$.
5. Compute the private key d such that

$$de \equiv 1 \pmod{\phi(n)}.$$

6. Output $(pubkey, privkey)$ where $pubkey = (n, e)$ and $privkey = d$.

- *Encryption of a message M*

$$C = \text{AsymEnc}(pubkey, m) \equiv M^e \pmod{n}.$$

- *Decryption of a ciphertext C*

$$M = \text{Dec}(privkey, C) \equiv C^d \pmod{n}.$$

3.4.2 ElGamal encryption

ElGamal encryption scheme is applicable to any cyclic group \mathbb{G} with a large prime order q and a generator g [24].

- *Key generation*

1. Select a large prime q .
2. Select g to be a primitive root (generator) in \mathbb{G} .
3. Select $x \in_R \mathbb{Z}_q^*$.
4. $h = g^x \pmod{q}$.
5. Output $(pubkey, privkey)$ where $pubkey = (g, h, q)$ and $privkey = x$.

- *Encryption of a message M*

1. Select $r \in_R \mathbb{G}$.
2. $C_1 = g^r \pmod{q}$.
3. $C_2 = Mh^r \pmod{q}$.
4. Output (C_1, C_2) .

- *Decryption of a ciphertext C*

$$M = C_2 C_1^{-x} \pmod{q}.$$

3.5 Digital signatures

A digital signature scheme is a cryptographic method for proving the authenticity and integrity of a digital communication or a document. Below, we present elliptic curve digital signature algorithm (ECDSA) and Schnorr signatures, though it should be noted that Edwards-curve digital signature algorithm (EdDSA) (a variant of Schnorr signatures) and BLS are also used in different blockchains such as Cardano and Ethereum 2.0 [25, 26].

Schnorr signatures support batch verification, allowing multiple signatures to be verified simultaneously, resulting in improved efficiency. Unlike Schnorr, ECDSA does not naturally support batch verification. Verifying multiple ECDSA signatures requires individual computations for each signature. Moreover, Schnorr signatures are generally more efficient in terms of computation and signature size compared to ECDSA. Schnorr signatures require fewer computational operations and result in shorter signatures. As a result, both Schnorr signatures and ECDSA are widely used and offer secure digital signature algorithms. Schnorr signatures present efficiency and security advantages over ECDSA; however, ECDSA has a broader history of adoption and standardization. The selection between them may depend on specific use cases and compatibility requirements with existing systems.

3.5.1 ECDSA

The ECDSA is the elliptic curve equivalent of the DSA and is one of the most widely used algorithms [27].

- *Key generation*

1. Select an elliptic curve E with modulus p , and coefficients a and b . Let P be a point on the curve generating the prime order of the cyclic group \mathbb{G} .
2. Choose a random integer d with $0 \leq d \leq q$.
3. Compute $Q = d \cdot P$.
4. Output $(pubkey, privkey)$ where

$$pubkey = (p, a, b, q, P, Q) \text{ and } privkey = d.$$

- *Signing on a message M*

1. Select a random value k with $2 \leq k \leq q$.
2. Compute $R = k \cdot P$
3. Let $r = x_R \bmod q$ where x_R is the x -coordinate of R .
4. Compute $s = k^{-1}(\text{Hash}(M) + d \cdot r) \bmod q$.
5. Output the signature pair (r, s) .

- *Verification of a signed message (r, s) for a given M*

1. Verify if r and s are integers in $[1, q]$.
2. Compute $w = \bmod q$.

3. Compute $u_1 = Hash(M)s^{-1} \pmod q$.
4. Compute $u_2 = rs^{-1} \pmod q$.
5. Compute a point $A = u_1P + u_2Q$.
6. The signature is valid if $r = x \pmod q$.

3.5.2 Schnorr signatures

The Schnorr signature scheme has had a significant impact on the way cryptographic protocols are created. The signature scheme is based on an identification scheme that is a three-move honest-verifier zero-knowledge proof of knowing a discrete logarithm. This is achieved with the help of the Fiat-Shamir transform. The Schnorr signature as digital signature scheme which is composed a tuple of algorithms $Signature = (KeyGen, Sign, Verify)$ [28].

- *Key generation* ($KeyGen(s) \rightarrow (pk, sk)$ for a security parameter s)

1. Pick a randomly generated private key $sk = x$.
2. The public key is $pk = y = g^x$.

- *Signing* ($Sign(sk, m) \rightarrow \sigma$ on a message m)

1. Pick a random k .
2. Calculate $r = g^k$.
3. Calculate $e = H(r||M)$.
4. Calculate $s = k - xe$.
5. The signature is the pair (s, e) .

Note that If $s, e \in \mathbb{Z}_q$ and $q < 2^{160}$ then the signature can fit into 40 bytes.

- *Verification* ($Verify(pk, M, (s, e)) \rightarrow \{0, 1\}$)

1. Calculate $r_v = g^s y^e$
2. Calculate $e_v = H(r_v||M)$
3. If $e_v = e$ then the signature is valid.

3.6 Xoshiro256++ (XOR/shift/rotate)

Blackman and Vigna presented xoshiro256++, which is a Pseudorandom Rumber Generator (PRNG) with 64 bits that uses a specific linear transformation [29]. Xoshiro256++ has a large enough state space for any concurrent application and passes all tests. Theoretically, xoshiro256++ performs simpler processes and is easily parallelizable utilizing Intel's extended and is three-dimensionally equally distributed. Only a shift and a rotation are required for the xoshiro linear transformation. Since it updates the whole state at each iteration, it is only practical for states of modest sizes.

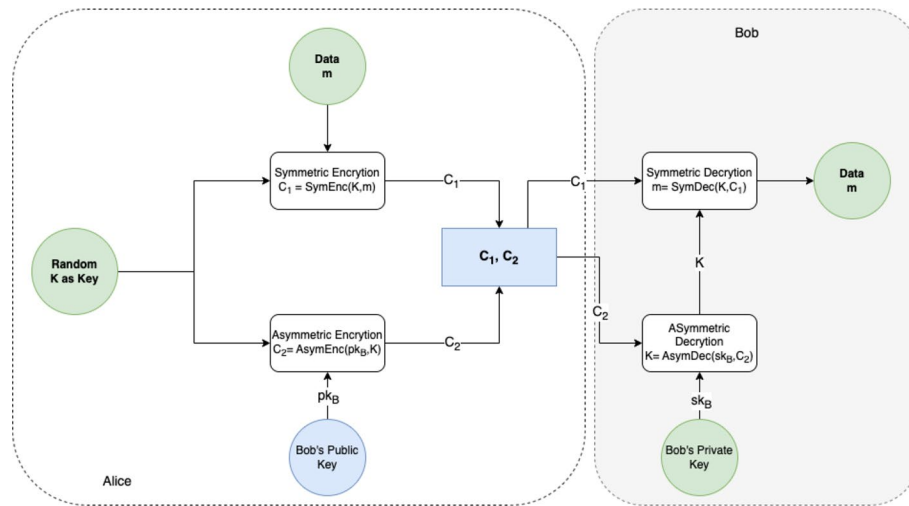


Fig. 1 Hybrid encryption architecture: This figure illustrates the hybrid encryption architecture, which combines symmetric and asymmetric encryption methods to optimize both security and performance. The process starts by generating a symmetric key (K), which is used to encrypt the message (M) using a symmetric encryption function $\text{SymEnc}(K, M)$ to produce the ciphertext (C_1). Concurrently, K is encrypted with the recipient's public key using an asymmetric encryption function $\text{AsymEnc}(pk, K)$ to generate (C_2). The recipient uses their private key to decrypt (C_2), retrieves K , and then decrypts (C_1) to obtain M . This dual encryption approach ensures data confidentiality while maintaining efficient decryption by the receiver

3.7 Hybrid encryption scheme

Hybrid encryption is commonly used to swiftly secure data communication through a combination of symmetric and asymmetric encryption schemes. On a high level, the sender generates a symmetric key, encrypts the key using a public key, and then encrypts the entire data with the symmetric key. The ciphertext can only be decrypted if the receiver knows the sender's symmetric key. To illustrate this process more concretely, let (pk_B, sk_B) be Bob's public and private key pair. If Alice intends to send an encrypted message to Bob using a hybrid encryption scheme, then she performs the following steps (see Fig. 1):

1. Request Bob's public key pk_B .
2. Generate a new symmetric key K .
3. Encrypt the data m as $C_1 = \text{SymEnc}(K, m)$.
4. Use Bob's public key to encrypt the symmetric key as $C_2 = \text{AsymEnc}(pk_B, K)$.
5. Send the two ciphertexts C_1 and C_2 to Bob.
6. Bob uses his own private key sk_B to decrypt C_2 and obtains K as $K = \text{AsymDec}(sk_B, C_2)$.
7. Bob decrypts C_1 using K and obtains the data as $m = \text{SymDec}(K, C_1)$.

4 Related work

In this section, we present and discuss previous attempts and proposals aimed at mitigating MEV attacks.

4.1 Flashbots 2.0: frontrunning in decentralized exchanges

Daian et al. [11] developed the MEV concept and related risks, showing that the blockchain revolution and the use of smart contracts failed to provide a purely peer-to-peer version of digital cash. As they were compared with traditional exchanges in centralized systems, as happened for Wall Street stock but which is not a valid comparison. However, they present priority gas auctions (PGAs) and define them as arbitrage bots that compete with each other by bidding up higher gas fees. Depending on their model for the bot, PGA behavior causes heavy traffic on the network and raises gas prices. Additionally, the same research presents an auction model that allows for a Nash equilibrium for players to take turns bidding, which is consistent with the observed behavior in Ethereum. It also focuses on MEV as it measures how much value miners can derive from users by the way transactions are ordered; for example, miners can decide which Mempool transactions go into blocks and in what order. The research of Daian et al. [11] led to a new research direction into MEV strategies for both miners and bots, as well as possible ways to stop MEV (Flashbots [8] and Eden Network [9]); all of these projects targeted solving MEV. In general, all these projects are based on ideas around creating private transaction pools, which are based on private agreements with miners and that let users send transactions without going through the public Mempool.

Remark 1 If any participant in the private network used in Flashbots 2.0 is malicious, then Flashbots 2.0 based solutions do not mitigate Frontrunning attack.

4.2 MEV protection on a DAG

Malkhi et al. [30] introduced a new line of research showing that Byzantine fault-tolerant (BFT) protocols use Directed Acyclic Graphs (DAGs, which are graphs made up of vertices and edges that connect pairs of vertices and have varied uses in science and computing) [31] to mitigate MEV. BFT with DAG provides high throughput by keeping network utilization high, separating the spread of transactions from the order of their metadata, and efficiently encoding consensus logic over a DAG that shows the causal order of messages that have been spread. They discuss this by introducing a DAG-based protocol called Fino, which adds MEV resistance features to DAG-based BFT without slowing down the steady spread of transactions by the DAG transport and with no message overhead.

4.3 SGX protection against MEV attacks

Intel's Software Guard Extensions (SGX) is a set of extensions to the Intel architecture that aims to provide integrity and confidentiality guarantees to security-sensitive computation performed on a computer where all the privileged software (kernel, hypervisor, etc) is potentially malicious. The enclave is the foundation of SGX, and it is where all the data and instructions for a secure computation are kept [32]. In MEV-SGX, the nodes participating in the auction are required to run their software in a secure enclave, such as Intel SGX, to ensure the integrity of the software used for the auction [13].

MEV-SGX could help Flashbots to realize the design objective of developing a truly private and permissionless system. Searchers would generate blocks containing their bundles, validate, and encrypt those blocks in their SGX, and deliver them to miners along with block truncated header hashes. Miners receive reduced header hashes and encrypted blocks that they recognize as valid and profitable to mine. They use the truncated header hashes to do proof-of-work on blocks without viewing them, and after discovering a proof-of-work solution, they can decode and seal blocks. Since SGX does not protect against cache timing attacks, the authors of the privileged enclave cannot employ data-dependent memory accesses. Cache attacks on the Quoting Enclave, which computes attestation signatures, would, for instance, allow an assault with a processor's enhanced privacy ID (EPID) signing key and entirely compromise SGX [32].

Remark 2 If one of the SGX CPU leaks the private key of the participant, SGX based solutions would not provide security against MEV.

4.4 Threshold encryption against MEV attacks

A (t, n) threshold encryption scheme is used to distribute the decryption process between n participants where at least t members are required to decrypt a given ciphertext. This is generally used to prevent single points of failure. Threshold encryption contains expensive asymmetric operations such as exponentiation (or multiplication in elliptic curve operation). Furthermore, the threshold decryption process requires multiple parties to be involved, and this brings additional significant overhead to the blockchain consensus that requires high transactions per second (TPS). Adapting threshold encryption would require a committee of block producers to decrypt encrypted transactions submitted by searchers. Each miner would receive a portion of a decryption key, and some threshold (for example, n of m) would be required to decrypt transactions. While this technique provides some additional privacy and validity assurances, it is challenging to join the set of critical holders via a permissionless procedure because it is based on a fair majority assumption that the key holders will not collude to break the encryption. Threshold encryption by committee also introduces a bandwidth-intensive step to block output that may become unsustainable. Threshold encryption could be a potential next step if these challenges can be addressed [13].

Remark 3 If threshold number of participants in the threshold encryption is malicious, threshold-based solutions do not provide security against MEV.

4.5 Multi-party computation against MEV attacks

Multi-party computation (MPC) is a cryptographic tool that allows many participants to perform calculations on their combined data without revealing their individual contribution. In particular, let $f : X^n \rightarrow Y$ be a function and let P_1, \dots, P_n be n parties such that P_i has a private input $x_i \in X$ [33]. An MPC for a functionality f is a protocol between the parties who mutually compute and output $y_i = f(x_1, \dots, x_n)$ without disclosing their inputs to each other. Informally, the protocol is secure if it reveals nothing except y_i to each participant [34].

MPC has the following advantages [35]:

1. Data are immune to intelligence and third-party access: MPC reduces reliance on third-party service providers by securing data and calculations within the internal networks of businesses.
2. MPC preserves data accessibility and confidentiality: MPC makes it straightforward to perform collaborative calculations without hiding any variables. Without losing precision, data confidentiality is maintained in its entirety.
3. MPC complies with regulatory and privacy standards: In MPC, data are split into bits to increase security and is never transferred in its entirety across international boundaries, ensuring compliance with the various data protection standards.

However, MPC has the following drawbacks that impede its practical usage.

1. Costs of computing and communication are high: MPC techniques generate a lot of random numbers, which takes a lot of computing power. In addition, many different kinds of server computers and storage devices can slow down MPC protocols. MPC stores pieces of data in different places. These pieces are then reconstructed to form the final result. To bring people together, you need communication tools, which can add to the cost of deployment.
2. Malicious participants must be assumed to be taking part: Therefore, to implement MPC safely, one needs to be able to make accurate predictions about how many malicious parties will be involved.

Remark 4 If threshold number of participants in the MPC network are malicious, MPC-based solutions do not provide security against MEV.

4.6 Danksharding against MEV attacks

Sharding is the act of breaking up a blockchain network into smaller pieces called shards. Danksharding [10] is a shard design that uses PBS and crList [36] to reduce MEV.

In PBS, block builders and proposers are two significant players. Block builders have blocked constructors, while block proposers choose the constructed block, take the transaction from it, and send it to the Ethereum network. Proposers choose the block transactions with the highest bid (priority fee) and send that information to the chain. Builders have more power in this system. They get a list of transactions from the proposer called the crList [36]. They can then rearrange transactions on the crList to make the most money for themselves. Since builders have control over block data, crList helps stop data censorship by forcing builders to include txns in a block. This ensures that builders remain honest and makes the network trust them less. Because MEV consumption is shifting from miners to builders and proposers, centralized MEV consumption is no longer a problem. Danksharding intends to make Ethereum Layer 1 a rollup, data availability, and settlement layer.

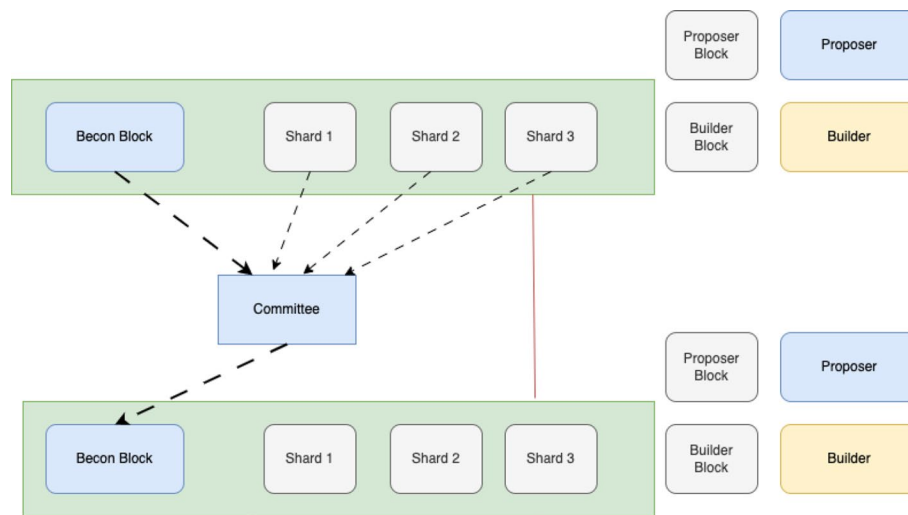


Fig. 2 Danksharding [10]: This figure highlights the Beacon Chain as a critical component of Ethereum 2.0, essential for coordinating shards and managing the consensus algorithm within the Danksharding framework. Danksharding enhances transaction throughput and data availability by using shard chains to handle specific transactions within Ethereum 2.0, each responsible for processing a subset of transactions and generating blocks. Additionally, Danksharding introduces “dank ordering,” a new transaction ordering technique that sorts transactions by their miner extractable value (MEV), prioritizing those that contribute significantly to the network’s benefit. This strategy not only connects execution and sharded blocks but also ensures efficient validation of data through aggregation, minimizing delays in shard block confirmation. The increased capacity to process larger data volumes under Danksharding supports rollups and synchronous calls between ZK Rollups and Layer 1, simplifying the rollup architecture and enhancing network efficiency

According to Fig. 2, the Beacon chain is a key component of Ethereum 2.0, coordinating shards and managing the consensus algorithm. Danksharding heavily relies on the Beacon chain to coordinate transaction ordering across shards.

- *Shard chains*: These are unique chains in the Ethereum 2.0 network that handle transactions for specific shards. Danksharding requires each shard chain to handle a subset of transactions and generate blocks.
- *Transaction ordering*: Danksharding introduces a novel technique to transaction ordering known as “dank ordering.” Transactions are sorted according to their MEV, with higher MEV transactions given priority. This is intended to incentivize miners to include transactions that benefit the network rather than solely maximizing their own profits. As a result, the execution and sharded blocks are connected. Validation of data is achieved through aggregation. This ensures no delays in shard block confirmation, and Danksharding enables Ethereum to process significantly larger amounts of data than it could previously. This facilitates rollups by allowing synchronous calls between ZK Rollups and Layer 1, thus simplifying rollup design.

Remark 5 Danksharding does not show any resistance if the proper declined to take the auctioned block from the builder and keep making his blocks, so this might need more development on the design. Also, the size increase should be considered as dependent on such a sharding process.

4.7 Blockchain with vehicle ad-hoc network (VANETs)

In VANETs, the utilization of blockchain technology could significantly mitigate the risk of malicious participants exploiting MEV attacks, ensuring a more secure and transparent vehicular communication environment. This section explores various innovative approaches that integrate blockchain into VANETs to enhance their security and operational efficiency.

4.7.1 An adaptive real-time malicious node detection framework using machine learning in vehicular ad hoc networks (VANETs)

Rashid et al. [37] introduce an adaptive, real-time framework for detecting malicious nodes in VANETs, leveraging advanced machine learning techniques to enhance network security. The paper emphasizes the urgency of addressing security threats such as distributed denial of service (DDoS) attacks within VANETs and proposes a comprehensive solution incorporating a distributed multi-layer classifier (MLPC). This system employs a variety of machine learning models, including gradient boosted trees (GBT), logistic regression (LR), MLPC, random forest (RF), and support vector machine (SVM). The dataset used encompasses normal and attacking vehicles to validate the effectiveness of the proposed model in real-time scenarios. By implementing a distributed system, the computational load is efficiently distributed among vehicles, thereby enhancing the speed and accuracy of malicious node detection. Furthermore, the adoption of a neural network architecture with multiple layers significantly strengthens the framework's ability to classify and detect malicious activities accurately. The accurate simulation and testing phase ensures that the framework is robust and can be effectively integrated into real-world VANET environments, providing a reliable defense mechanism against potential cyber threats.

4.7.2 BBSF: blockchain-based secure weather forecasting information through routing protocol in VANET

Sohail et al. [38] propose the blockchain-based secure forecasting (BBSF) technique, aimed at enhancing the safety and efficiency of VANETs through secure and efficient spreading of weather forecasting information. Utilizing blockchain technology, the BBSF framework not only secures weather data but also optimizes routing processes to ensure rapid and reliable information delivery. This approach significantly reduces hop counts and network latency, while improving packet delivery ratios and minimizing network overhead. Key to the framework's success is highly efficient weather forecasting servers that employ the Hyperledger Sawtooth transaction mechanism, ensuring the integrity and security of data across the network. The paper details a routing strategy that maximizes packet delivery rates and minimizes end-to-end delays by considering node connectivity, channel reliability, and the number of hops involved. Moreover, the integration of blockchain technology provides a decentralized and manipulation-proof system, enhancing the trustworthiness and security of data spreading. Secure routing protocols, coupled with the use of public and private

keys for data decryption and encryption, respectively, protect the privacy and accuracy of the transmitted weather forecasting information, thereby strengthening the overall efficacy and security of vehicular communication networks.

4.7.3 VABLOCK: a blockchain-based secure communication in V2V network using ICN network support technology

Ali et al. [39] propose VABLOCK, an innovative framework that integrates blockchain technology and information-centric networking (ICN) to address security and trust challenges in vehicle-to-vehicle (V2V) communications within VANETs. The framework employs a blockchain-based method for secure message spreading, enhancing the integrity and non-repudiation of communication data. Furthermore, it leverages ICN's content-centric approach to improve the efficiency and reliability of content delivery, eliminating location dependencies through enhanced caching mechanisms. The paper also introduces a cluster-based communication strategy where vehicles are organized into clusters with designated cluster heads (CH) that manage communication and content caching. To ensure robust security, the framework incorporates a trust management mechanism that assesses vehicle trustworthiness to mitigate risks caused by malicious nodes and utilizes blockchain for content validation to protect against data tampering. The efficacy of the proposed solution is validated through simulations in Network Simulator-2 (NS-2), demonstrating significant improvements in cache hit ratio, one-hop count, malicious node detection, and delivery ratios over existing methods.

4.8 Comparison of proposed solution and existing approaches

Table 1 provides a comparative analysis of the proposed two-tiered architecture utilizing verifiable decryption against existing approaches in the field such as Flashbots [11], Dank Sharding [40], DAG [41], Intel SGX [42], Threshold Encryption [43], and MPC [44]. This comparison highlights the unique advantages and addresses the limitations of our approach compared to other significant methodologies in the field.

5 MEV attacks and our adversary model

5.1 Miner (or maximal) extractable value (MEV)

MEV refers to the maximum value for a miner from block production by including, excluding, or reordering the transactions in that block (i.e., frontrunning, backrunning, or sandwiching transactions). The value produced from MEV is a separate value to the block rewards or transaction fees of the miners [7, 11, 45–47].

In frontrunning [48], a signed transaction is sent to the miners, who are paid to add it to a block in the chain. For example, in Ethereum, there is no way to know who will mine the next block ahead of time, and a transaction is usually shared with the whole network. The Mempool is where nodes keep track of these pending transactions. How long it takes for a transaction to be added to a block depends on how much gas was paid and how much space is in the block. Anyone accessing an Ethereum node can look at the Mempool to see what transactions have been made. If a pending transaction (called a victim transaction Tx_{victim}) meets certain conditions, an attacker can send a new transaction $Tx_{attacker}$ with a slightly higher gas price. If a miner orders transactions based on the gas price, the attacker's transaction $Tx_{attacker}$ will occur prior to the transaction

Table 1 Detailed technical comparison of existing approaches

Approach	Key features	Technical advantages	Technical limitations
Flashbots [11]	Transaction ordering marketplace	Reduces negative externalities such as gas price spikes and network congestion by providing a separate system for miners and traders	Introduces a semi-centralized component, increasing coordination complexity and potential power imbalances
Dank sharding [40]	Shard chains and data availability sampling	Increases throughput by distributing transactions across shards and secures data without full downloads.	Implementation complexity, changes to consensus mechanisms, and potential new security vulnerabilities
Directed acyclic graph (DAG) [41]	Parallel transaction processing	Highly scalable and reduces transaction confirmation latency, suitable for high-throughput applications.	Difficulties in achieving and maintaining consensus, increased complexity in global state maintenance
Intel SGX [42]	Trusted execution environments	Secures code and data from external attacks, crucial for data confidentiality and secure computation	Vulnerable to side-channel attacks; effectiveness depends on hardware integrity and enclave software
Threshold encryption [43]	Multi-party key cooperation for decryption	Enhances data security by requiring multiple parties for decryption, preventing unauthorized access.	Increases coordination latency, complexity in key management, and overhead
Multi-party computation (MPC) [44]	Privacy-preserving data analysis	Allows deriving data insights without exposing underlying data, ideal for collaborative scenarios	Computationally intensive, significant communication overhead, complexity grows with participant number
Proposed solution	Robust security measures	Ensures all system actions are auditable and verifiable, reducing potential for manipulations or attacks.	Performance impact due to complex cryptographic measures like RSA encryption with large keys

Tx_{victim} . Frontrunning has been looked at for a long time in traditional markets; Daian et al. [11] wrote a lot about it for the first time in Ethereum. They saw that bots not only make new transactions based on what is happening in the Mempool, but also bid against each other on gas prices for better block placement.

Backrunning is like frontrunning in that you use what you know about a transaction to place an order immediately after the transaction you want to copy. Most of the time, a transaction changes the exchange rate on one exchange but not others. This is called “*price slippage*”. Backrunners use this information to profit from the price difference between different DEX exchanges. In this work, backrunning takes MEV out of the system by exchanging the same cryptocurrency on multiple DEXs simultaneously, maybe in different amounts [47]. Frontrunning on Ethereum is commonly executed through “*Sandwich*” attacks, where a user’s transaction is sandwiched between two other transactions, resulting in a loss for the user and a gain for the attacker. This attack involves placing the two transactions before and after the user’s transaction, thereby intercepting and exploiting it. Sandwich attacks are widely used and are considered one of the most prevalent forms of Frontrunning on Ethereum. Numerous research papers [45, 47, 48] have explored the potential profitability of frontrunning and backrunning tactics. Various attack strategies exist, ranging from replay attacks on arbitrage to complex plans involving collectibles. The success of frontrunning attacks is largely determined by the sequence in which transactions are processed within a block. As miners possess full control over transaction ordering, profits from such attacks are known as miner extractable values. This refers to the monetary gain that miners can obtain by arranging pending transactions in a favorable order or by adding new transactions to a block.

To ensure MEV resistance, Flashbots [8] defined the following goals:

- *Pre-trade privacy*: Transactions are only known to the public after they have been added to a block. This leaves out intermediaries such as relays and block builders.
- *Failed trade privacy*: Losing bids are never added to a block and are therefore never shown to the public.
- *Efficiency*: The design must be efficient while extracting MEVs without incurring needless network or network congestion.
- *Bundle merging*: Multiple incoming bundles can be combined without causing problems.
- *Finality protection*: Flashbots blocks with Flashbots bundles cannot be changed once they have been sent to the network. This stops time-bandit chain re-org attacks.
- *Complete privacy*: Relays and validate worthy intermediaries that can censor transactions.
- *Permissionless*: There are no trustworthy intermediaries that can censor transactions.

5.2 Our adversary model

There are a number of different strategies that can be used to mitigate MEV attacks. In this paper, we fully focused on the PBS technique where the miner or validator who proposes a block (called as Builder) is separate from the miner or validator who builds the block (called as Executor). This makes it more difficult for miners or validators to extract MEV from the

network. In this kind of scheme, users may submit doubly encrypted transactions by using public key of both builder and executor. This may prevent malicious builder or executor to deny including transactions. In this context, we introduce three potential adversary scenarios for PBS-based schemes. Note that builder and executors can also be used by an end user to gain profit from any arbitrage.

Definition 1 (*Corrupted builder (CB)*): A corrupted builder may behave arbitrarily. In particular, they may not decrypt transactions correctly and may also try to avoid the transactions to be included into the block.

In the presence of corrupted builders, the decryption process done by the builders must be verifiable by either users or executors.

Definition 2 (*Corrupted executor (CE)*): A corrupted executor may act in an arbitrary manner, including the possibility of incorrect decryption of encrypted transactions.

In the presence of corrupted executors, we have to make sure that decryption has been computed correctly and that they can be verified by users including builders.

Definition 3 (*Corrupted builder and executor (CBE)*): Both builder and executor could be malicious and colluding parties.

The scenario involving colluding builders and executors is the most challenging case. This is due to the potential for malicious offline communication and decryption to gain an advantage, which might require observation on the blockchain during calculations. Furthermore, they could opt to reject transactions when they could get any advantage.

Remark 6 (*A Condition for CBE Attack*): CBE attack could occur if the Builder and Executor are in the subsequent (adjacent) block production.

Remark 7 (*Mitigation against CB and CE Attacks*): To mitigate both CB and CE attacks, the decryption should be done in such a way that anyone would be able to do the encryption process again and compute the same ciphertext in a deterministic manner.

Remark 8 (*Mitigation against CBE Attack*): To mitigate CBE attack, the distribution of validators for block production must be randomized. This can be solved by the underlying consensus mechanism. For example, Aura consensus already provides a random selection of block producers.

6 The Mangata proposal through two-layered architecture

Mangata is blockchain protocol for decentralized exchange built on the Polkadot network [49] and bridged via Ethereum. Mangata DEX runs on proof of liquidity consensus, providing fixed-fees for trading and is the first layer 1 that aims to prevent MEV. The Mangata team built a decentralized crypto exchange to try to protect it from a variety of different attacks, especially MEV attacks, through modifying block execution by:

- Separating block construction and block execution.
- Introducing extrinsic shuffling.
- Introducing the concept of an encrypted transaction.

The Themis protocol was also proposed by the Mangata team as a solution for MEV. This protocol focuses on removing the abilities that create MEV from any network participant, hence making all network participants more equal [12]. Mangata proposes that all value extractions come down to two primary abilities [12]: the power to change the order of transactions, and the power to deny transactions, as defined through value extraction by reordering (VER) and value extraction by denial (VED).

6.1 Value extraction by reordering (VER)

Mangata splits block production into two consecutive steps:

- *Block building*: Transactions are accepted into a block.
- *Block execution*: The transaction execution order is reshuffled using information that did not exist at the time of block building.

Block building phase.

- The block builder gathers transactions from the Mempool and builds the block during the first stage.
- Then, they provide a seed to the subsequent miner who uses it to shuffle the transaction execution order and, ultimately, achieve the same blockchain state as the miner. The Mangata teams use Schnorr's signature [50] variant for seeding, both to generate and validate seeds.

Block execution phase.

- The block executor will sign a seed using the private key. This seed will later be used in the shuffling process to mix up the transactions in block n subsequent to the Fisher–Yates shuffle [51], which is used to form a random permutation of a finite sequence. To shuffle an array a of n elements, the following algorithm is executed. For $i = n - 1$ until $i = 1$, do:

1. Select a random integer j such that $0 \leq j \leq i$.
2. Swap $a[j]$ with $a[i]$.

In addition, Xoshiro256++ algorithms are used, which are random number generators via shift-register generators that use rotations in addition to shifts [52].

- Private keys are utilized because they are unknown to the block builder and cannot be altered by the block executor. The signature is immutable, and the scheme is predetermined. At the same time, a public key is available to everyone because it is

stored on the blockchain. This means that anyone can check whether the signature was made correctly.

- The executor then builds a new block (performing the first step for the subsequent block) and supplies the private key-signed seed. This successfully creates a seed chain that can be used to shuffle transactions. This generates a seed chain for transaction shuffling.

This separation of concerns ensures that the block builder cannot influence the execution order and that the block executor cannot alter the block content and must shuffle the transactions. This creates a two-block head of the blockchain and doubles the duration required to execute each block. That means there will be overhead because that block builder needs to pre-execute a transaction to assess whether these are valid ones, then state modifications are discarded, and the block is propagated. This effectively reduces the number of transactions that can fit into the block by half. This VED is built on top of the substrate, which is a programming framework to allow users to create a blockchain by picking the features from various “pallets” [53]. A blockchain developer, for instance, picks the palette for their preferred consensus algorithm to establish how consensus will operate on their blockchain. In addition, existing pallets allow blockchain developers to rapidly add functionality to a blockchain without having to build it themselves [54].

6.2 Value extraction by Denial (VED)

In this attack scenario, if an attacker has the power to decide whether or not to include a transaction, then they can choose to either not to include the transaction or to replace it with their own. We contend that if miners alone have this opportunity, the design is inherently unfair because users will never gain access to such an opportunity to gain pure profit. To prevent MEV, we need to stop these two powers from occurring or minimize their effects to the greatest extent possible [12]. There are three levels of VED solutions, from the least resilient to the most robust:

1. Miners should not reject transactions.
2. Miners do not know which transactions to reject.
3. Miners cannot reject transactions.

The current state of the VED solution has achieved the second level of robustness, in which the Miner (or any transaction relay) does not know what to deny because they have no knowledge of the transaction’s purpose. To accomplish this, the Mangata team has proposed the VED architecture as a proof of concept (see Fig. 3). In this proposal, transactions are encrypted by the user using both the builder’s and executor’s public keys. This double encryption ensures that the transaction executor can only decrypt if the block builder has already decrypted it. This process is described (see also Fig. 1).

1. Take transactions $m_E = (Tx_1, \dots, Tx_n)$.
2. Compute $C_1 = SymEnc(K_E, m)$, where K_E is a new fresh symmetric key.
3. Encrypt $C_2 = AsymEnc(pk_v, K_E)$, where pk_E is the public key of the executor.

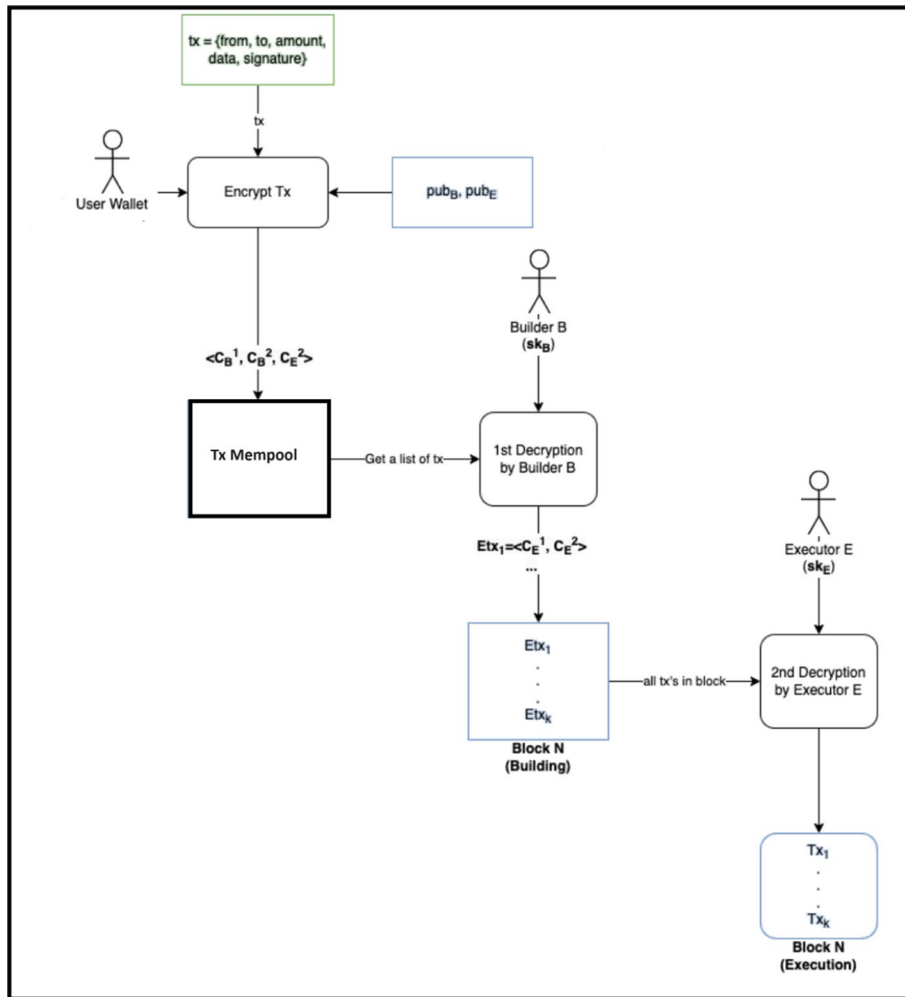


Fig. 3 Enhanced value extraction by Denial (VED) based on our proposal. This figure presents the enhanced value extraction by Denial (VED) framework aimed at mitigating MEV through cryptographic assurances. The VED model uses a two-layer cryptographic approach where transactions are first encrypted by the transaction initiator using the builder’s public key, denoted as $Enc_B(tx)$, and again by the executor’s public key, represented as $Enc_E(Enc_B(tx))$. This nested encryption ensures that only the appropriate executor and builder can decrypt and execute transactions in their respective orders, supported by hash functions to ensure integrity and prevent unauthorized alterations

4. Compute $C_3 = SymEnc(K_B, m_B)$, where K_B is a new fresh symmetric key and $m_B = (C_1, C_2)$.
5. Encrypt $C_4 = AsymEnc(pk_B, K_B)$, where pk_B is the public key of the builder.
6. Send C_3 and C_4 to the Mempool.

In the subsequent steps, block builders are required to decrypt the transaction, but this does not reveal the content because the block executor must itself perform a final decryption. Each transaction is encrypted twice: once during block building, and again during block execution. In this method, the transaction builder is unable to become aware the transaction’s contents and the executor is forced to decrypt and execute the transaction. This implies that the block builder and block executor must be known beforehand, and hence, submission into the Mempool cannot be node-agnostic, but

must include this information. It is important to remember that encryption is optional and should only be used when it makes sense. This is because processing an encrypted transaction is costly, and there is no guarantee that the value created by processing the transaction will be sufficient to cover the cost of processing. Since no one knows what is in the encrypted transaction, it may even be impossible to execute. This is why, unlike exchange transactions that are not encrypted, all encrypted transactions must have fixed gas costs.

6.3 Informal security analysis of the Mangata scheme

According to architecture presented before, the scheme could potentially have the following weaknesses.

Theorem 1 *Mangata's VER proposal is not secure against a corrupted builder.*

Proof A malicious or CB can deny any targeted transaction from being included in the block because all the transactions are in plain format (i.e., they are readable by the CB). □

Theorem 2 *Mangata's VER proposal is also not secure against in the presence of a corrupted executor.*

Proof A malicious executor has only one way of attacking, and it is a derivation of a signed seed that will be used to shuffle and execute transactions in the building block. The seed has already been derived by the builder, and the signature for the seed is deterministically computed via the executor's private key. Therefore, there is no apparent opportunity for a malicious adversary to obtain an expected signed seed. The only attack could be happen by the corrupted executor is not to produce any block. Hence, VER is not secure against a CE-type of adversary if further mitigations are not provided. □

Theorem 3 *Mangata's VER proposal is not secure if the builder and executor collude maliciously.*

Proof A corrupted builder can deny any targeted transaction from being included in the block because all the transactions are in plain format. The malicious builder can clone any transaction that has profitable arbitrage and can deny the original one to be included in the current building block. □

The VED proposal was offered to overcome those weaknesses, and it needs to be clearly defined so the security analysis can be achieved fairly.

7 Our proposal: how to mitigate MEV attacks through verifiable decryption

In this section, we describe our proposed scheme, which improves upon the Mangata architecture by providing a verifiable decryption process to mitigate their weaknesses. We provide a security analysis for each class according to our adversarial model defined in Sect. .

We extend and improve the VED proposal introduced by Mangata Finance by providing a verifiable decryption scheme. In the following, we introduce how an end user can sequentially encrypt a transaction with executor and, subsequently, builder public keys. The main idea underlying the proposal is that whenever a decryption is performed by one of them, the deciphered values will be broadcasted. Any entity in the network can then verify whether the decryption has been performed correctly.

Let $pk_B = (e_B, N_B)$ be the RSA public key of a builder and $pk_E = (e_E, N_E)$ be the RSA public key of an executor. Let $sk_B = d_B$ and $sk_E = d_E$ denote the RSA private keys of the builder and executor, respectively. Assume that the hash function H is SHA256 and the length of a symmetric key is 256 (i.e., AES-256). Let us also assume that the message m_1 in $C_1 = SymEnc(K, m_1)$ is already padded according to the PKCS7 padding standard [55]. Similarly, the message m_2 in $C_2 = AsymEnc(pk_E, m_2)$ is padded according to OAEP standard [56].

7.1 Encryption by users

For a given transaction tx , the user performs the following steps before submitting the transaction:

1. Randomly selects two symmetric keys

$$K_E, K_B \in_{\mathcal{R}} \{0, 1\}^{256}.$$

2. Computes the hash of the transaction $h_E = H(tx)$.
3. Concatenates the hash with the transaction as

$$m_E = (tx, h_E).$$

4. Encrypts the transaction with K_E

$$C_E^1 = SymEnc(K_E, m_E).$$

5. Computes the hash for the builder as $h_B = H(C_E^1)$.
6. Concatenates the hash and the cipher as

$$m_B = (C_E^1, h_B).$$

7. Re-encrypts the message m_B with K_B

$$C_B^1 = SymEnc(K_B, m_B).$$

8. Encrypts the key K_B with the builder's public key

$$C_B^2 = \text{AsymEnc}(pk_B, K_B).$$

9. Encrypts the key K_E with the executor's public key

$$C_E^2 = \text{AsymEnc}(pk_E, K_E).$$

Now, the user can broadcast the triple (C_B^1, C_B^2, C_E^2) to the network.

7.2 Decryption by builders

Once the builder selects a list of transactions from Mempool, the builder performs the followings for each transaction:

1. Decrypts C_B^2 with his private key sk_B . Let us denote K_B' , which is the decrypted value after padding. The user also removes the padding and obtains the key K_B .
2. Decrypts C_B^1 through K_B and obtains m_B , which is the concatenation of C_E^1 and h_B .
3. Verifies if $h_B \stackrel{?}{=} H(C_E^1)$ to check if the integrity is ensured.
4. If the verification holds, include (C_E^1, C_E^2) in the building block and broadcast $(K_B', C_B^1, C_B^2, pk_B)$ for further verification by the community (i.e., other validator nodes).

7.3 Decryption by executors

Once the building block is broadcasted, the executor does the following

1. Decrypts C_E^1 with his private key sk_E . Let's denote K_E' which is the decrypted value after padding. The user also removes the padding and obtains the key K_E .
2. Decrypts C_E^2 with the recovered secret key K_E and obtain m_E , which is concatenation of tx and h_E .
3. Verifies if $h_E \stackrel{?}{=} H(tx)$ to check if the integrity of the transaction is ensured.
4. If verification holds, executes the transaction tx and broadcast $(K_E', C_E^1, C_E^2, pk_E)$ for further verification by the community (i.e., other validator nodes).

7.4 Public verification by the community

Anyone can verify the correctness of the whole computation as follows:

- *Verification of builder's calculations*

```

Function verify_builder( $K'_B, C_B^1, C_B^2, pk_B$ ):
 $K'_B = \text{remove\_padding}(K_B)$ 
 $(C_E^1, h_B) = \text{symmetric\_decrypt}(K_B, C_B^1)$ 

if  $h_B \neq H(C_E^1)$  then return False
if not verify_modular_equation( $K_B'^{e_B}, N_B \stackrel{?}{=} C_B^2$ ) then return False
return True

```

- *Verification of executor's calculations*

```

Function verify_executor( $K'_E, C_E^1, C_E^2, pk_E$ ):
 $K'_E = \text{remove\_padding}(K_E)$ 
 $(tx, h_E) = \text{symmetric\_decrypt}(K'_E, C_E^1)$ 

if  $h_E \neq H(tx)$  then return False
if not verify_modular_equation( $K_E'^{e_E}, N_E \stackrel{?}{=} C_E^2$ ) then return False
return True

```

7.5 Liveness protection and slashing

There are several cases to be considered as forms of attack, and these scenarios should be considered as shown:

- *Liveness attack*: This attack can delay the transaction acknowledgment timings of their targets and provide two instances of such attacks against Bitcoin and Ethereum. The liveness attack proceeds in three phases: preparation, transaction denial, and blockchain delay. This attack delays the confirmation of a transaction. The attacker attempts to obtain a possible edge over honest participants during the preparation phase to construct their private chain. This is followed by the transaction denial phase, in which the attacker attempts to delay the transaction's authentic block. If the attacker determines that the delay is not convincing, they move on to the blockchain render phase, in which they attempt to slow the rate at which the chain transaction grows [57].
- *Slashing*: A malicious actor can disrupt a staking pool through either slashing or reputation loss. In Ethereum, validators who are deemed to have acted against the chain are penalized with monetary penalties. These penalties are designed to deter attacks on the currency. Slashing is the process by which a significant portion of a validator's stake is "burned" if that validator is deemed to have behaved inappropriately. Slashing is a mechanism for policing behavior collectively. Importantly, a malicious actor who has compromised a validator's signing key would be able to intentionally commit actions that would result in severe slashing penalties [58].

The slashing penalties for various blockchains vary. The two primary reasons for enforcing the slashing penalty are:

- To encourage validators to behave responsibly.
- To make network attacks costly and unattractive.

The two most typical instances in which the validator can be charged are:

- During downtime (when the validator is not present to sign transactions).
 - Double signing (when the validator signs two or more blocks at the same height) [59].
- *The “Nothing at Stake” attack* In spite of the variety of PoS protocols, validators have an incentive to work on multiple forks because generating a block in PoS is equivalent to generating a single signature. In other words, validators could generate conflicting blocks on all possible forks with nothing at stake in order to maximize the benefits. This issue is generally known as the nothing at stake attack. This attack reduces the network’s consensus time and, consequently, the system’s efficiency. In addition, it results in blockchain forks, which compromise the blockchain’s ability to defend against double spending attacks and other threats. Specifically, validators can disregard the algorithm for fork resolution and generate blocks on top of multiple forks. In addition, because the eligibility proof for each account is deterministic, it is straightforward to anticipate which validators will generate valid blocks in the future. This is commonly referred to as “transparent forging” and adds a new attack surface to the blockchain, allowing attackers to choose the next leader to compromise with precision [60].

8 Security analysis of proposal

In this section, we prove that our scheme presented in Sect. is secure against the adversary models presented in Sect. by considering each adversary separately.

8.1 Cryptographic and security definitions

Cryptographic primitives are defined and their security is assessed under standard cryptographic models. This forms the basis for the upcoming proof of theorem, which go through into the robustness of our system when facing attacks from corrupted builders and executors (CBE).

- *Symmetric encryption* (SymEnc, SymDec) Assumed secure under the Chosen Plaintext Attack Chosen Plaintext Attack(CPA) model, which is defined as: \forall probabilistic polynomial-time adversaries \mathcal{A} ,

$$\Pr[k \leftarrow \text{KeyGen}_{\text{sym}}(), c \leftarrow \text{SymEnc}_k(m), \mathcal{A}(c) = m] \leq \frac{1}{|\mathcal{M}|} + \text{negl}(n)$$

where $\text{negl}(n)$ is a negligible function in the security parameter n , and $|M|$ is the size of the message space.

- *Asymmetric encryption* (AsymEnc, AsymDec) Assumed IND-CCA (Indistinguishability under Chosen Ciphertext Attack) secure. The security definition is formalized as the inability of any efficient adversary \mathcal{B} to distinguish between ciphertexts of chosen plaintexts under an adaptive chosen ciphertext attack:

$$\Pr \left[b' = b : \begin{array}{l} (pk, sk) \leftarrow \text{KeyGen}_{\text{asym}}(), \\ (m_0, m_1, \text{state}) \leftarrow \mathcal{B}^{\text{Dec}_{sk}(\cdot)}(pk), \\ b \leftarrow \{0, 1\}, c \leftarrow \text{AsymEnc}_{pk}(m_b), \\ b' \leftarrow \mathcal{B}^{\text{Dec}_{sk}(\cdot)}(c, \text{state}) \end{array} \right] - \frac{1}{2} \leq \text{negl}(n)$$

- *Hash function* H Assumed to provide collision resistance, where it is computationally infeasible for any efficient algorithm \mathcal{A} to find two distinct inputs x and x' such that $H(x) = H(x')$:

$$\Pr[x, x' \leftarrow \mathcal{A}(), x \neq x', H(x) = H(x')] = \text{negl}(n)$$

Theorem 4 Assume that the underlying encryption algorithms (SymEnc, SymDec), (AsymEnc, AsymDec) and the hash function H are secure. Then, our proposal described in Sect. is secure against CB attack.

Proof The security of the protocol is demonstrated through two primary vectors: the protocol behavior and its reduction to the underlying primitives.

More concretely, each plain transaction is encrypted with the executor and builder’s public keys subsequently. Once the builder performs the decryption, the decrypted message is still the encryption of original transaction with the executor’s public key. Therefore, the builder will not be able to see any arbitrage value in the original transaction. More concretely, each transaction is defined as a triple value (C_B^1, C_B^2, C_E^2) . Even if a builder decrypts a transaction $(\text{AsymDec}(sk_B, C_B^2) \rightarrow K_B, \text{SymDec}(K_B, C_B^1) \rightarrow (C_E^1, h_B))$, it cannot see the plain form because the transaction is also encrypted (C_E^1, C_E^2) by the next Executor. Hence, a malicious builder cannot deny any targeted transaction from being included in the block since all the transactions are in plain format and they are not readable by the builder.

Assume now an adversary \mathcal{A} can exploit information from $C = C_y^x$ where $x = 1$ or $x = 2, y = E$. This implies \mathcal{A} can derive information about tx , effectively breaking the CPA security of SymEnc, contradicting our initial security assumption. Consider the probability that \mathcal{A} succeeds:

$$\Pr[\mathcal{A}(C) = tx] = \Pr[\mathcal{A}(\text{SymEnc}_{k_B}(tx)) = tx]$$

By the security of SymEnc, this probability should be negligible:

$$\Pr[\mathcal{A}(C) = tx] \approx \frac{1}{|M|} + \text{negl}(n)$$

For IND-CCA security of AsymEnc, even with access to a decryption oracle, the security property guarantees that:

$$\left| \Pr[b' = b] - \frac{1}{2} \right| \leq \text{negl}(n)$$

indicating that C does not reveal information about tx to the Builder, even if corrupted. Under the assumption that SymEnc is CPA-secure and AsymEnc is IND-CCA secure, the protocol's design ensures that a Corrupted Builder, who has access only to C , cannot compromise the security or confidentiality of the transaction tx . \square

Theorem 5 *Assume that the underlying encryption functions (SymEnc, SymDec), (AsymEnc, AsymDec), signing algorithm Sign, and the hash function H are secure. Then, our proposal is secure against CE attack.*

Proof The security of the proposal against CE attacks can be illustrated through a comprehensive analysis of the transaction handling and cryptographic operations. First of all, all transactions in the block are encrypted by the current executors. An encrypted transaction (C_E^1, C_E^2) is decrypted by the executor $(AsymDec(s_E, C_E^2) \rightarrow K_E, SymDec(K_E, C_E^1) \rightarrow (tx, h_E))$, and it can easily be seen whether a transaction has any advantage. The order of transaction executions is a vital problem. Recall from the Mangata protocol, the builder signs a random value for ordering. If the executor signs the signed seed value in a deterministic way, the result would determine the order of transactions. Hence, a malicious executor has only one way of attacking, which is a derivation of the signed seed that will be used to shuffle and execute transactions in the building block. The seed has already been derived by the builder and its signature is deterministically calculated through the executor's private key. Therefore, it is not possible for the malicious adversary to obtain an expected signed seed. The only attack that could occur via the corrupted executor is not producing a block. However, this would also lead to a liveness attack and it could be prevented via a slashing mechanism. Hence, VER is secure against CE-types of adversary. More formally, the executor decrypts the doubly encrypted transaction using sk_E . Let us denote $C_1 = \text{SymEnc}_{k_B}(tx) = \text{AsymDec}_{sk_E}(C_2)$. Then, the executor should perform another decryption using k_B to access tx . However, without k_B , tx remains confidential.

- *Step 1:* The probability of decrypting C_1 without k_B does not compromise tx :

$$\Pr[\mathcal{A}(C_1) = tx] \leq \frac{1}{|M|} + \text{negl}(n)$$

- *Step 2:* Analysis of the integrity of transaction execution order:
 - Recall from the Mangata protocol, the builder signs a random value for ordering. If the executor signs the signed seed value in a deterministic way, the result would determine the order of transactions. Assume an executor attempts to reorder trans-

actions. Given the signatures involved and the deterministic nature of the protocol, any deviation would be detectable unless:

$$\Pr[\mathcal{B}(\text{forge signature})] \leq \text{negl}(n)$$

Hence, the protocol is secure against CE-type adversaries. The only attack that could occur via the corrupted executor is not producing a block. However, this would also lead to a liveness attack and it could be prevented via a slashing mechanism. \square

Theorem 6 *Assume that the underlying encryption functions (SymEnc , SymDec), (AsymEnc , AsymDec), signing algorithm Sign , and the hash function H are secure. Then, our proposal is secure against CBE attack with probability $2/n$ where n denotes the number of nodes.*

Proof Assume that Builder_{*i*} and Executor_{*i*} have colluded maliciously. In this case, once an encrypted transaction is inserted into the Mempool, they can recover the plaintext transaction in advance (offline) and check whether the plaintext transaction has any advantage. However, the main difficult of occurring this attack is the likelihood that these two nodes will be assigned to subsequent slots. Since our proposal uses randomness to periodically determine the nodes responsible for building and executing blocks, the probability that two colluding nodes will be assigned to the same slot is $2/n$. To calculate the probability that two randomly selected nodes out of n nodes in the system are consecutive, we can consider the total number of possible pairs and then count the number of pairs where the users are consecutive. Let's assume that the users are labeled with consecutive integers from 1 to n . The total number of possible pairs of users is given by combinations of n choose 2, denoted as $C(n, 2)$, and calculated as

$$C(n, 2) = n! / [(2!(n-2)!)]$$

Now, we need to count the number of pairs where the users are consecutive. If you choose any user numbered from 1 to $(n-1)$ as the first user in the pair, then the second user will be the next consecutive integer. Therefore, there are $(n-1)$ pairs of consecutive nodes. So, the probability that two randomly selected nodes are consecutive is:

$$\begin{aligned} \text{Probability} &= (\text{Number of Consecutive Pairs}) / (\text{Total Number of Pairs}) \\ &= (n-1) / [n! / (2!(n-2)!)] \\ &= (n-1) / [(n * (n-1)) / 2] \\ &= 2/n \end{aligned} \tag{1}$$

Hence, the probability of two randomly selected users being consecutive is $2/n$. Since separate CB and CE attacks cannot succeed, we can conclude that our proposal is also secure against the CBE attack with probability $2/n$. \square

Remark 9 We would like to highlight that even if the Mangata VED proposal utilizes a verifiable encryption scheme, it would not be still secure against CBE attack. The attack would work on the assumption that two subsequent nodes are malicious. Assume that Builder_{*i*} and Executor_{*i*} have colluded maliciously. Once an encrypted transaction is

Table 2 Computation complexity of the proposal where a block contains ℓ transactions

Cryptographic methods	Client side calculations	Builder side calculations	Executor side calculations
RSA encryption	2ℓ	–	–
RSA decryption	–	ℓ	ℓ
Symmetric encryption	2ℓ	–	–
Symmetric decryption	–	ℓ	ℓ
ECDSA signing	ℓ	1	1
ECDSA verification	–	ℓ	$\ell + 1$

inserted into Mempool, they can recover the plain transaction in advance and can check whether the plain transaction has any advantage. Since the Mangata consensus utilizes the AURA algorithm and the order of the validator is deterministic and known through the lifetime of all block generations, the possibility of having two subsequent malicious nodes is very high. Therefore, their VED proposal is not secure against CBE attack.

9 Implementation and benchmarking

This section outlines the cryptographic operations for each party and evaluates the overall system performance through detailed benchmarking results. We have implemented the cryptographic algorithms, RSA decryption and symmetric decryption to illustrate the system's effectiveness and efficiency (see our GitHub repository at²). Performance benchmarks were conducted on an Intel(R) Core(TM) i5-10210U CPU, with comprehensive metrics provided.

9.1 Overview of cryptographic operations

This subsection provides an overview of the cryptographic roles of the client, builder, and executor, setting the stage for more in-depth analysis in subsequent sections. It outlines the scope for evaluating the performance and efficiency of the proposal with comparisons highlighted in Table 2. Assuming each block contains ℓ transactions, we focus on the cryptographic computations involved: encryptions, decryptions, and signature verifications under both symmetric and asymmetric schemes.

- *Client side calculations:* Initially, each transaction within a block is signed using the user's private key. Subsequently, the client performs two RSA encryptions, denoted by 2ℓ , as follows:

- Encryption of the key K_B using the builder's public key:

$$C_B^2 = \text{AsymEnc}(pk_B, K_B).$$

- Encryption of the key K_E using the executor's public key:

² <https://github.com/Mustafa25022022/Mitigating-MEV-Attacks-with-a-Two-Tiered-Architecture-Utilizing-Verifiable-Decryption.git>

$$C_E^2 = \text{AsymEnc}(pk_E, K_E).$$

Additionally, two symmetric encryptions, also denoted as 2ℓ , are performed:

- Encryption of the transaction using K_E :

$$C_E^1 = \text{SymEnc}(K_E, m_E).$$

- Re-encryption of the message m_B using K_B :

$$C_B^1 = \text{SymEnc}(K_B, m_B).$$

- *Builder side calculations*: The builder selects ℓ transactions from the Mempool and for each transaction performs the following RSA and symmetric decryption operations:

- RSA decryption to retrieve K_B :

$$K_B' = \text{Decrypt}_{sk_B}(C_B^2), \quad K_B = \text{RemovePadding}(K_B'),$$

- Symmetric decryption to retrieve the original message:

$$m_B = D(K_B, C_B^1) \quad (\text{where } D \text{ denotes the decryption operation})$$

- Verification of transaction integrity:

$$\text{Verification: } h_B \stackrel{?}{=} H(m_B) \quad (\text{hash comparison})$$

- The builder finalizes by signing the block using ECDSA.

- *Executor side calculations*: Upon receiving the proposed block, the executor verifies the block's integrity and performs the following calculations for each transaction:

- Verification of the RSA decryption and removal of padding:

$$K_E' = \text{Decrypt}(sk_E, C_E^1), \quad K_E = \text{RemovePadding}(K_E')$$

- Symmetric decryption to recover the transaction data:

$$m_E = D(K_E, C_E^1) \quad (\text{retrieving the concatenated transaction data and hash})$$

The Executor completes the process by signing the finalized block using ECDSA.

Existing proposals suffer from serious security vulnerabilities, and our proposal is the first construction that deals with certain malicious adversaries in the MEV adversary model. Therefore, it could be crucial to extend the proposed security model with dynamic adversaries as well as UC (Universal composability) framework considering environmental attacks.

Table 3 Summary of encryption and decryption times

Statistic	Encryption time (s)	Decryption time (s)
Count	1000.000000	1000.000000
Mean	0.002555	0.006267
SD	0.000660	0.001097
Minimum	0.000968	0.003989
25% Percentile	0.001995	0.005955
50% Percentile	0.002568	0.005988
75% Percentile	0.002992	0.006688
Maximum	0.012267	0.023261

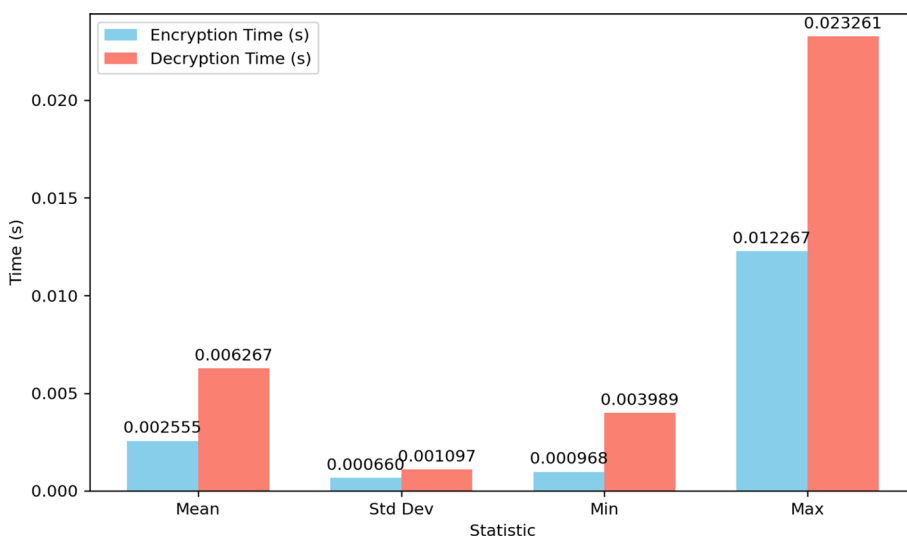


Fig. 4 Key statistics of encryption and decryption time. This figure quantitatively analyzes the encryption and decryption times within our proposed architecture. It demonstrates the distribution of computational times across multiple test instances, applying statistical metrics such as mean, median, and standard deviation. The mathematical formulation of the encryption ($E(m, k)$) and decryption ($D(c, k)$) functions is also presented, focusing on their computational complexity and impact on overall system performance

Table 4 Statistical summary of key generation and integrity check times

Statistic	Key generation time (s)	Integrity check time (s)
Count	1000.000000	1000.000000
Mean	2.844310	0.000336
SD	1.436287	0.000484
Minimum	0.414737	0.000000
25% Percentile	1.813997	0.000000
50% Percentile (Median)	2.602540	0.000000
75% Percentile	3.584175	0.000996
Maximum	10.695417	0.001511

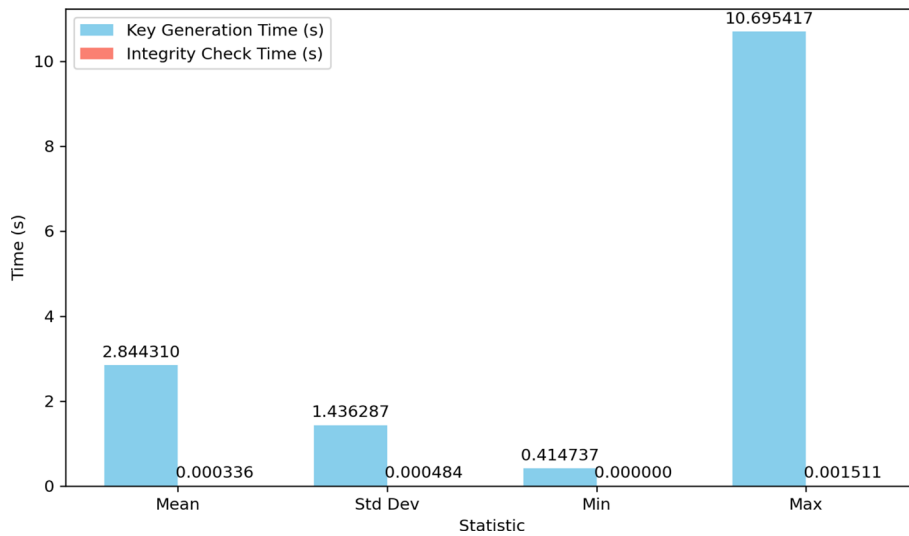


Fig. 5 Key statistics of key generation and integrity check times. This figure provides a detailed analysis of the time required for cryptographic key generation and integrity checks within our system. It illustrates the variance and trends of key generation times using the algorithm $G(n)$, where n is the key length. Additionally, integrity checks are graphed to show their time efficiency, utilizing hash functions $H(x)$ to ensure data integrity. The relationships and performance impacts of these cryptographic operations are discussed, highlighting their importance in maintaining secure and efficient blockchain operations

Table 5 Summary of node trials, probabilities, and statistical tests

Nodes	Trials	Empirical prob	Theoretical prob	Chi-square stat	p-value
200	20,000	0.0098	0.0100	0.05	0.831
300	30,000	0.0076	0.0067	4.23	0.040
400	40,000	0.0050	0.0050	0.02	0.887
500	50,000	0.0043	0.0040	0.98	0.321
600	60,000	0.0032	0.0033	0.18	0.671
700	70,000	0.0023	0.0029	0.80	0.370
800	80,000	0.0022	0.0025	0.20	0.654
900	90,000	0.0021	0.0022	0.05	0.823
1000	100,000	0.0024	0.0020	0.80	0.371

9.2 Our implementation, benchmarking, and results

This subsection describes our implementation of our proposal and presents the benchmarking results to illustrate the system’s effectiveness and efficiency.

- *Encryption and decryption times:* We conducted 1,000 trials to measure operational efficiency. The results show a mean encryption time of 0.002555 s and a decryption time of 0.006267 s. These times indicate high efficiency, making the system suitable for real-time applications, as illustrated in Table 3 and Fig. 4.
- *Key Generation and integrity check times:* Key generation demonstrated a robust performance, with a mean time of 2.844 s across 1,000 trials. Integrity checks were notably efficient, with a near-zero mean time, highlighting the system’s ability to promptly detect and prevent manipulation as illustrated in Table 4 and Fig. 5.

Table 6 Confusion matrix

	Predicted: agreement	Predicted: disagreement
Actual: agreement	TP = 8	FP = 0
Actual: disagreement	FN = 0	TN = 1

- *Overall system performance:* The total time from encryption to integrity checking was measured, with median values offering insight into the system's typical behavior under operational conditions. This demonstrates the system's ability to handle cryptographic operations swiftly, ensuring transaction security even in potential adversarial conditions.

The results confirm the efficiency of our protocol in maintaining efficient cryptographic operations. The rapid execution times for key generation, encryption, and integrity checks, along with their low variability, underscore the protocol's suitability for environments that require robust security measures against sophisticated threats. Future work will focus on further optimizing these operations to reduce execution times and enhance the system's resilience against dynamic adversarial conditions.

9.3 Data analysis and confusion matrix

Tables 5 and 6 summarize the probability of consecutive node assignments with their corresponding theoretical probabilities and statistical test results:

Our analysis demonstrates a strong correlation between empirical and theoretical probabilities across most node sizes, indicating effective randomness in node assignments. This supports the robustness of our assignment mechanism, an important factor in maintaining resilience against MEV and manipulation. However, in the 300-node setup where the p-value was less than 0.05, a deviation was observed. This deviation could indicate a vulnerability or an unexpected behavior, necessitating further investigation to uphold the security integrity of the network.

The confusion matrix provides a clear classification of the nodes based on empirical probabilities, theoretical predictions, and Chi-square test results:

The confusion matrix shows high agreement between empirical probabilities and theoretical predictions for most tested node sizes, as indicated by eight true positives. This consistency underscores the reliability of our theoretical model. The single true negative at Node 300, where a difference was detected, warrants further analysis to identify potential underlying issues, ensuring both the model's reliability and the network's security.

9.4 Risk analysis and statistical confidence

In evaluating the efficacy of our proposed two-tiered architecture for MEV attack mitigation, we compared the frequency of MEV attacks in both the experimental group (using our architecture) and a control group (using traditional methods).

Data

- *Experimental group*: Out of 1,000 trials, MEV attacks occurred 30 times.
- *Control group*: Out of 1,000 trials, MEV attacks occurred 60 times.

Results

- *Relative risk (RR)*: The relative risk was calculated as follows:

$$RR = \frac{\text{Probability of attack in experimental group}}{\text{Probability of attack in control group}} = \frac{30/1000}{60/1000} = 0.50$$

An RR of 0.50 indicates that the architecture reduces the risk of MEV attacks by 50% compared to the control method.

- *Absolute risk reduction (ARR)*

$$\begin{aligned} ARR &= \text{Probability in control group} - \text{Probability in experimental group} \\ &= \frac{60}{1000} - \frac{30}{1000} = 0.03 \text{ or } 3\% \end{aligned}$$

This shows a 3% absolute reduction in the rate of MEV attacks due to the implementation of our architecture.

Statistical confidence

- *95% confidence interval for RR*: The confidence interval for RR, calculated using the standard error of the natural logarithm of RR, was found to be [0.33, 0.77]. This interval suggests that while the RR estimate is robust, the variation due to sample size and inherent variability in attack rates must be considered.

The statistical measures of relative risk and absolute risk reduction provide strong evidence of the effectiveness of our proposed architecture in significantly mitigating MEV attacks. These findings, underscored by the reliable confidence intervals, confirm that the observed risk reductions are statistically significant and not due to random variation. Incorporating these risk analyses into our study not only demonstrates the quantitative benefits of our architectural improvements but also supports the broader adoption of this approach in practical blockchain applications to enhance security against MEV threats.

10 Discussion

10.1 Evaluation of findings and comparison with existing solutions

Our study confirms that a two-tiered architecture utilizing verifiable decryption significantly improves resistance to MEV attacks, outperforming traditional single-tiered systems. The integration of a verifiable layer enhances transparency and accountability in transaction processing, enhancing the integrity of blockchain operations [61].

Compared to existing solutions such as Flashbots [8] and the Eden Network [9], which focus primarily on reordering and auction mechanisms without a verifiable layer, our approach provides a more robust framework. By enabling both the block builders and executors to independently verify each other's actions, our architecture

significantly enhances security against collusion and malicious activities. This architecture is particularly beneficial for financial institutions and blockchain applications that demand high levels of trust and security.

10.2 Challenges and limitations

The primary challenge in our study related to the assumption of absolute compliance and integrity in the verifiable decryption process, which might not hold in real-world settings subject to sophisticated cyber attacks or hardware limitations. This limitation requires careful validation of our architecture under varied and possibly adversarial conditions to ensure robustness and applicability [62].

Operational challenges during our study included scalability issues and significant computational overhead when integrating the two-tiered architecture with existing large blockchain systems. These obstacles underscore the necessity for continued improvement and optimization of the architecture to ensure its practical adoption on a large scale. Issues related to the scalability of the verifiable decryption process and the computational demands of integrating new security features are critical areas that require ongoing attention.

11 Conclusion and future work

MEV attacks compromise the security and decentralization of blockchain networks by exploiting transaction sequencing vulnerabilities. Our study revisits and analyzes previous MEV mitigation strategies, identifying key architectural weaknesses. We advance the current state-of-the-art through our introduction of a verified decryption procedure, ensuring that decryption outcomes are broadcast for public verification, thereby enhancing network transparency and security. We demonstrate that our architecture robustly secures against a range of adversarial behaviors, including actions by corrupted builders and executors, both individually and in collusion.

Our architecture implements ℓ RSA encryptions to secure transactions against MEV attacks. Future research could focus on optimizing cryptographic efficiency by exploring alternatives to RSA, potentially employing more computationally efficient cryptographic constructions. It is possible to reduce communication complexity by aggregating multiple symmetric or asymmetric encryptions into a single operation and similarly aggregating signatures to enhance performance. Another promising research direction involves the development of keyless schemes, leveraging verifiable delay functions (VDFs) to generate verifiable pseudorandom outputs, thus eliminating key management vulnerabilities and reducing dependency on traditional cryptographic keys.

Abbreviations

ECDSA	Elliptic curve digital signature algorithm
MPC	Multi-party computation
PRNG	Pseudorandom number generator
MEV	Miner/maximal extractable value
zkRollups	Zero-knowledge rollups
PBS	Proposer-builder separation
VER	Value extraction by reordering
VED	Value extraction by denial
PoA	Proof-of-authority
EdDSA	Edwards-curve digital signature algorithm
VRF	Verifiable random function

CB	Corrupted builder
CE	Corrupted executor
CBE	Corrupted builder and executor
PGAs	Priority gas auctions
DAGs	Directed acyclic graphs
SGX	Intel's software guard extensions
zkEVM	Zero-knowledge ethereum virtual machine
DeFi	Decentralized finance
P2P	Peer-to-peer
crList	Censorship resistance scenarios
RSA	Rivest–Shamir–Adleman
BLS	Boneh–Lynn–Shacham
AURA	Authority round
BABE	Blind assignment for blockchain extension
BFT	Byzantine fault-tolerant
EPID	Enhanced privacy ID
TPS	Transactions per second
UC	Universal composability
VANETs	Vehicular ad hoc networks
CPA-Secure	Chosen plaintext attack security
IND-CCA Secure	Indistinguishability under chosen ciphertext attack
DDoS	Distributed denial of service
MLPC	Multi-layer classifier
GBT	Gradient-boosted trees
LR	Logistic regression
RF	Random forest
SVM	Support vector machine
BBSF	Blockchain-based secure forecasting
ICN	Information-centric networking
V2V	Vehicle-to-vehicle
CH	Cluster heads
NS-2	Network simulator-2

Acknowledgements

The authors would like to thank the anonymous reviewers for their valuable comments.

Author contributions

MIA, MSK, and SK contributed to the initiation of the research and MIA was a major contributor in writing the manuscript. AAB contributed to the missing parts and analyzed the security and revision of the manuscript. All authors read and approved the final manuscript.

Funding

The authors would like to thank the support of De Montfort University, Leicester, UK and Batman University, Turkiye, for paying the Article Processing Charges (APC) of this publication.

Availability of data and materials

Data sharing is not applicable to this article as no datasets were generated or analyzed during the current study.

Declarations

Ethics approval and consent to participate

Not applicable.

Consent for publication

Not applicable.

Competing interest

While the work was being carried out and a first draft of this paper was written, the authors had no conflict of interest.

Received: 1 September 2023 Accepted: 4 August 2024

Published online: 13 August 2024

References

1. S. Nakamoto, Bitcoin: A peer-to-peer electronic cash system. *Decentral. Bus. Rev.* 21260 (2008)
2. G. Wood et al., Ethereum: a secure decentralised generalised transaction ledger. *Ethereum Project Yellow Pap.* **151** (2014), 1–32 (2014)
3. K. Kulkarni, T. Diamandis, T. Chitra Towards a theory of maximal extractable value i: Constant function market makers. *arXiv preprint arXiv:2207.11835* (2022)
4. Optimistic Rollups. <https://ethereum.org/en/developers/docs/scaling/optimistic-rollups/>. Accessed on 16 May 2024

5. zkROLLUPS. <https://ethereum.org/en/developers/docs/scaling/zk-rollups/>. Accessed on 16 May 2024
6. ZKSync: zkEVM. <https://docs.zksync.io/zk-stack/components/zkEVM/overview.html#zkevm>. Accessed on 16 May 2024
7. A. Judmayer, N. Stifter, P. Schindler, E. Weippl, Estimating (miner) extractable value is hard, let's go shopping! In: International Conference on Financial Cryptography and Data Security, pp. 74–92 (2022). Springer
8. Flashbots Auction. <https://docs.flashbots.net/flashbots-auction/overview>. Accessed on 16 May 2024
9. Edennetwork: Multichain Infrastructure for Maximal Value. <https://www.edennetwork.io/>. Accessed on 16 May 2024
10. V. Buterin, Proto-Danksharding FAQ. https://notes.ethereum.org/@vbuterin/proto_danksharding_faq#What-is-Danks-harding. Accessed on 16 May 2024
11. P. Daian, S. Goldfeder, T. Kell, Y. Li, X. Zhao, I. Bentov, L. Breidenbach, A. Juels, Flash boys 2.0: Frontrunning in decentralized exchanges, miner extractable value, and consensus instability. In: 2020 IEEE Symposium on Security and Privacy (SP), pp. 910–927 (2020). IEEE
12. M. Team, Introducing Themis Protocol v1. A solution to MEV for Application-Specific Blockchains. <https://blog.mangata.finance/blog/2021-10-10-themis-protocol/> (2021). Accessed on 16 May 2024
13. R. Miller, MEV-SGX: A Sealed Bid MEV Auction Design. <https://ethresear.ch/t/mev-sgx-a-sealed-bid-mev-auction-design/9677> (2021). Accessed on 16 May 2024
14. Ethereum Foundation: Danksharding. <https://ethereum.org/en/roadmap/danksharding/>. Accessed on 16 May 2024
15. Q. Wang, R. Li, Q. Wang, S. Chen, Y. Xiang, Exploring unfairness on proof of authority: Order manipulation attacks and remedies. In: Proceedings of the 2022 ACM on Asia Conference on Computer and Communications Security, pp. 123–137 (2022)
16. Ethereum Foundation: Clique PoA protocol & Rinkeby PoA testnet. <https://github.com/ethereum/EIPs/issues/225>. Accessed on 16 May 2024
17. Ethereum Foundation: go-ethereum: Official Go implementation of the Ethereum protocol. <https://geth.ethereum.org/>. Accessed on 16 May 2024
18. S. De Angelis, L. Aniello, R. Baldoni, F. Lombardi, A. Margheri, V. Sassone, Pbft vs proof-of-authority: Applying the cap theorem to permissioned blockchain. *Int. J. Emerg. Sci. Eng. (IJESE)* (2018)
19. P. Ekparinya, V. Gramoli, G. Jourjon, The attack of the clones against proof-of-authority. arXiv preprint [arXiv:1902.10244](https://arxiv.org/abs/1902.10244) (2019)
20. H.K. Alper, BABE- Web3 Foundation. <https://research.web3.foundation/Polkadot/protocols/block-production/Babe>. Accessed on 16 May 2024
21. P. Rogaway, T. Shrimpton, Cryptographic hash-function basics: Definitions, implications, and separations for preimage resistance, second-preimage resistance, and collision resistance. In: International Workshop on Fast Software Encryption, pp. 371–388. Springer (2004)
22. H. Delfs, H. Knebl, Symmetric-Key Encryption, pp. 11–31. Springer, Berlin (2007). https://doi.org/10.1007/3-540-49244-5_2
23. C. Paar, J. Pelzl, Understanding Cryptography: A Textbook for Students and Practitioners, Springer, Berlin (2009)
24. A. Shetty, K. Shravya, K. Krithika, A review on asymmetric cryptography-rsa and elgamal algorithm. *Int. J. Innov. Res. Comput. Commun. Eng.* **2**(5), 98–105 (2014)
25. Cardano Foundation: Cardano. <https://cardano.org/>. Accessed on 16 May 2024
26. Ethereum Foundation: Ethereum Roadmap. <https://ethereum.org/en/upgrades/>. Accessed on 16 May 2024
27. A. Abidi, B. Bouallegue, F. Kahri, Implementation of elliptic curve digital signature algorithm (ecdsa). In: 2014 Global Summit on Computer & Information Technology (GSCIT), pp. 1–6 (2014). IEEE
28. G. Neven, N.P. Smart, B. Warinschi, Hash function requirements for Schnorr signatures. *J. Math. Cryptol.* **3**(1), 69–87 (2009)
29. D. Blackman, S. Vigna, Scrambled linear pseudorandom number generators. *ACM Trans. Math. Softw. (TOMS)* **47**(4), 1–32 (2021)
30. D. Malkhi, P. Szalachowski, Maximal extractable value (mev) protection on a dag. arXiv preprint [arXiv:2208.00940](https://arxiv.org/abs/2208.00940) (2023)
31. P. Ferraro, C. King, R. Shorten, On the stability of unverified transactions in a dag-based distributed ledger. *IEEE Trans. Autom. Control* **65**(9), 3772–3783 (2020). <https://doi.org/10.1109/TAC.2019.2950873>
32. V. Costan, S. Devadas, Intel SGX Explained. *Cryptology ePrint Archive*, Paper 2016/086. <https://eprint.iacr.org/2016/086> (2016)
33. A.C.-C. Yao, How to generate and exchange secrets. In: 27th Annual Symposium on Foundations of Computer Science (sfcs 1986), pp. 162–167 (1986). <https://doi.org/10.1109/SFCS.1986.25>
34. A. Ozdemir, D. Boneh, Experimenting with Collaborative zk-SNARKs: Zero-Knowledge Proofs for Distributed Secrets. *Cryptology ePrint Archive*, Paper 2021/1530. <https://eprint.iacr.org/2021/1530> (2021)
35. D. Evans, V. Kolesnikov, M. Rosulek, A pragmatic introduction to secure multi-party computation. *Found. Trends® Privacy Secur.* **2**(2-3), 70–246 (2018). <https://doi.org/10.1561/33000000019>
36. H.E. Edition, Crlist. <https://notes.ethereum.org>. Accessed on 16 May 2024
37. K. Rashid, Y. Saeed, A. Ali, F. Jamil, R. Alkanhel, A. Muthanna, An adaptive real-time malicious node detection framework using machine learning in vehicular ad-hoc networks (vanets). *Sensors* **23**(5), 2594 (2023)
38. H. Sohail, M. Hassan, M. Elmagzoub, A. Rajab, K. Rajab, A. Ahmed, A. Shaikh, A. Ali, H. Jamil, Bbsf: Blockchain-based secure weather forecasting information through routing protocol in vanet. *Sensors* **23**(11), 5259 (2023)
39. A. Ali, M.M. Iqbal, S. Jabbar, M.N. Asghar, U. Raza, F. Al-Turjman, Vablock: A blockchain-based secure communication in v2v network using icn network support technology. *Microprocess. Microsyst.* **93**, 104569 (2022)
40. Sharding. <https://github.com/ethereum/wiki/wiki/Sharding-FAQs/c54cf1b520b0bd07468bee6950cda9a2c4ab4982>. Accessed on 16 May 2024
41. Iota. <https://www.iota.org/foundation/research-papers>. Accessed on 16 May 2024
42. S. Johnson, V. Scarlata, C. Rozas, E. Brickell, F. Mckeen et al., Intel software guard extensions: Epid provisioning and attestation services. *White Pap.* **1**(1–10), 119 (2016)

43. R.L. Rivest, A. Shamir, L. Adleman, A method for obtaining digital signatures and public-key cryptosystems. *Commun. ACM* **21**(2), 120–126 (1978)
44. A.C. Yao, Protocols for secure computations. in: *23rd Annual Symposium on Foundations of Computer Science (sfcs 1982)*, pp. 160–164 (1982). IEEE
45. K. Qin, L. Zhou, A. Gervais, Quantifying blockchain extractable value: How dark is the forest? in: *2022 IEEE Symposium on Security and Privacy (SP)*, pp. 198–214 (2022). <https://doi.org/10.1109/SP46214.2022.9833734>
46. M. Bartoletti, J.H.-Y. Chiang, A. Lluch Lafuente, Maximizing extractable value from automated market makers. in: *Financial Cryptography and Data Security*, pp. 3–19. Springer, Cham (2022)
47. J. Piet, J. Fairuze, N. Weaver, Extracting godl [sic] from the salt mines: Ethereum miners extracting value. arXiv preprint [arXiv:2203.15930](https://arxiv.org/abs/2203.15930) (2023)
48. P. Züst, Analyzing and Preventing Sandwich Attacks in Ethereum. Bachelor's thesis, ETH Zurich (2021)
49. Polkadot: Polkadot Network. <https://polkadot.network/>. (Accessed on 16 May 2024)
50. H. Morita, J.C. Schuldt, T. Matsuda, G. Hanaoka, T. Iwata, On the security of the schnorr signature scheme and dsa against related-key attacks. in: *ICISC 2015*, pp. 20–35 (2015). Springer
51. J. Arndt, Generating random permutations. PhD thesis, Australian National University (2010). <https://www.jjj.de/pub/arndt-rand-perm-thesis.pdf>
52. Mangata: Mangata Node. <https://github.com/mangata-finance/mangata-node>. Accessed on 16 May 2024
53. Substrate: Consensus. <https://docs.substrate.io/fundamentals/consensus>. Accessed on 16 May 2024
54. I.J. Scott, M. de Castro Neto, F.L. Pinheiro, Bringing trust and transparency to the opaque world of waste management with blockchain: A polkadot parathread application. *Comput. Ind. Eng.* **182** (2023). <https://doi.org/10.1016/j.cie.2023.109347>
55. B. Kaliski, PKCS #7: Cryptographic Message Syntax Version 1.5. RFC Editor (1998). <https://doi.org/10.17487/RFC2315>. <https://www.rfc-editor.org/info/rfc2315>
56. D. Pointcheval, I. OAEP: Optimal Asymmetric Encryption Padding, pp. 443–445. Springer, Boston, MA (2005)
57. S. Singh, A.S.M.S. Hosen, B. Yoon, Blockchain security attacks, challenges, and solutions for the future distributed IoT network. *IEEE Access* **9**, 13938–13959 (2021)
58. A. Bhudia, A. Cartwright, E. Cartwright, J. Hernandez-Castro, D. Hurley-Smith, Extortion of a staking pool in a proof-of-stake consensus mechanism. in: *2022 IEEE International Conference on Omni-layer Intelligent Systems (COINS)*, pp. 1–6 (2022). <https://doi.org/10.1109/COINS54846.2022.9854946>
59. G. Fanti, L. Kogan, P. Viswanath, Economics of proof-of-stake payment systems. In: Working Paper, pp. 1–6 (2021)
60. W. Li, S. Andreina, J.-M. Bohli, G. Karame, Securing proof-of-stake blockchain protocols. in: *Data Privacy Management, Cryptocurrencies and Blockchain Technology: ESORICS 2017 International Workshops, DPM 2017 and CBT 2017*, Oslo, Norway, September 14–15, 2017, Proceedings, pp. 297–315 (2017). Springer
61. K. Venkatesan, S.B. Rahayu, Blockchain security enhancement: an approach towards hybrid consensus algorithms and machine learning techniques. *Sci. Rep.* **14**(1), 1149 (2024)
62. J. Akinsola, M. Adeagbo, S. Akinseinde, F. Onipede, A. Yusuf, Applications of blockchain technology in cyber attacks prevention. in: *Sustainable and Advanced Applications of Blockchain in Smart Computational Technologies*, pp. 129–159 (2022)

Publisher's Note

Springer Nature remains neutral with regard to jurisdictional claims in published maps and institutional affiliations.