Protocol Security Review

Dusk

v1.0

September 5, 2024

Table of Contents

Table of Contents **2**
License **3**
Disclaimer **3**
Introduction **5**
Purpose of This Report 5
Documentation Submitted for the Review 5
Methodology 6
Functionality Overview 6
Summary of Significant Findings **7**
Consensus Protocol **8**
Protocol Overview 8
Maturity 9
Detailed Findings **9**
1. Future block generators can be predicted by the current block generator 9
2. Not allowing contracts to perform direct calls may be too restrictive 10
3. Subsidized transactions may be abused by block producers 11
Economic Model **12**
Protocol Overview 12
Maturity 12
Detailed Findings 13
4. Intentional out-of-gas offers the possibility to pay for transactions only if they are successful. 13
5. Strategic implications of the unspent gas kickback might still incentivize ineffective contract designs 13
6. Manipulation of gas prices 14
7. Automatic gas price estimation may be difficult and gameable depending on the implementation 15
8. Total supply limit leads to a shrinking security budget and deflationary economic incentives 16
9. High initial supply concentration of the ERC20 token and related risks of proof-of-stake and proof-of-work systems 17
10. Slashing and validator incentives are undocumented 18
11. Attestor incentives are undocumented 18
Literature review on fee modeling for blockchains and the suboptimality of first-price auctions 18

License

Disclaimer

Oak Security

https://oaksecurity.io/
info@oaksecurity.io

Introduction

Purpose of This Report

Oak Security has been engaged by DUSK NETWORK BV to perform a security review of Dusk network.

The objectives of the review are as follows:

1. Generate a risk model for the Dusk network
2. Identify security concerns related to the consensus protocol
3. Identify security concerns related to the economic model
4. Generate a risk model for the network
5. Provide initial input for parameterization of the economic model

This report represents a summary of the findings.

As with any security review, there is a limit to which vulnerabilities can be found. The authors of this report do not guarantee complete coverage (see disclaimer).

Documentation Submitted for the Review

The review has been performed on the following targets:

Repository	https://github.com/dusk-network/wiki
Commits	`386831ea9e4b68573d6bfe47656a5b109dd74243`
Scope	All relevant documentation regarding the consensus protocol and economic model.

Repository	https://github.com/dusk-network/dusk-protocol
Commits	`e37345c883df1cb5134d4000f3cff0abf2798d23`
Scope	All relevant documentation regarding the consensus protocol and economic model.

Methodology

The protocol security review has been performed in the following steps:

1. Gaining an understanding of the protocol intended purpose by:
    a. Screening publicly available information
    b. A kick-off call with the client
    c. Reading the available documentation
    d. Reviewing the literature mentioned by the protocol and related to the protocol
2. Analyzing threats and vulnerabilities, the use of best practice guidelines, related but not limited to:
    a. Consensus manipulation
    b. Centralization issues
    c. DoS Attacks
    d. Economic incentives
    e. Game theoretical considerations
3. Report preparation

Functionality Overview

The scope of the review focuses on the consensus protocol and Dusk's economic model. Dusk is a Turing-complete blockchain protocol that aims to host privacy-preserving decentralized and regulated financial applications. The novelties include a fully custom-built consensus protocol, a smart contract fee mechanism at the base layer of the blockchain that allows abstracting gas payments for users, and an incentivization mechanism for smart contract developers to write efficient and gas-saving contracts.

Summary of Significant Findings

Overall the protocol is well documented and contains the typical specifications expected. One exception is the lack of documentation on the incentivisation, reward and slashing specifications for validators and attestors.

Furthermore, three issues are highlighted:

1. The consensus protocol works such that future block generators can be predicted by the current block generator, which could increase the risk of MEV and DoS attacks.
2. Contracts might run out of gas strategically to avoid the payment of fees.
3. The incentives for gas-saving contracts might still incentivize ineffective contract designs.

Consensus Protocol

Protocol Overview

Dusk uses the Succinct Attestation (SA) protocol, a permission-less, committee-based Proof-of-Stake consensus protocol. The protocol is run by all stakers, known as provisioners, who are responsible for generating, validating, and ratifying new blocks. Participation occurs in rounds, with provisioners being pseudo-randomly selected for each phase of the consensus using the Deterministic Sortition algorithm. Each round consists of iterations, with three steps: Proposal, Validation, and Ratification. If a candidate block is produced in the Proposal step and a supermajority of votes is reached in favor of the block, the candidate is added to the blockchain.

The Deterministic Sortition algorithm is used in the SA protocol to extract provisioners in a non-interactive way based on their stakes. The algorithm takes a list of provisioners and a number of voting credits to assign to the extracted provisioners. It is executed in a pseudo-random way, based on the round, iteration, and step. To prevent pre-calculating extractions for future rounds, a block-unique Seed value is used, which is computed by the block generator by signing the Seed of the previous block. The extraction process is based on a pseudo-random Score value generated by hashing the parameters `(round, iteration, step, seed)` along with the index of the provisioner to extract. Each provisioner can be assigned one or more credits, depending on its stake. The extraction algorithm follows a weighted distribution that favors provisioners with higher stakes: the higher the stake, the higher the probability of being assigned a credit. The algorithm ensures that eligible provisioners participate in committees with a frequency and power proportional to their stakes.

In the Proposal step, the previously described sortition algorithm is used by provisioner nodes to extract the block generator. This node creates, signs, and broadcasts a block, while the other nodes wait to receive the candidate block.

During the Validation step, a committee of provisioners verifies the candidate block's validity and casts their votes. The committee is formed by pseudo-randomly selecting provisioners using the Deterministic Sortition algorithm. The main purpose of the Validation step is to agree on whether a candidate block has been produced and if it is a valid new tip of the blockchain. The output of this step is used as the input for the Ratification step.

In the Ratification step, another committee of provisioners agrees upon the result of the Validation step. The main purpose of this step is to ensure that provisioners are "aligned" with respect to the Validation result. If a quorum is reached for any result, a Quorum message is generated with the aggregated signatures of both the Validation and the Ratification steps. The output, together with the Validation output, is used to determine the outcome of the iteration.

The finality of the SA protocol is achieved through the concept of Consensus State, which defines whether a block can or cannot be replaced by another one from the network. Blocks in the local chain can be in three states: Accepted, Attested, and Final. An Accepted block can be replaced by a lower-iteration one, while an Attested block cannot be replaced by a lower-iteration block with the same parent but one of its predecessors is Accepted and could be replaced. A Final block is definitive and cannot be replaced in any case. The protocol also specifies global parameters such as Minimum Stake, Epoch Duration, Committee Credits, Maximum Iterations, Rolling Finality Blocks, and Maximum Step Timeout to govern the consensus process.

Maturity

The SA protocol is a well-designed consensus protocol that combines different aspects of existing protocols. However, while the protocol seems to plan on penalizing misbehavior, the specific details of its slashing implementation were not provided in the available information. Slashing is a crucial component as it introduces financial incentives for correct behavior and disincentivizes misbehavior among provisioners. Without a clear description of the slashing mechanism, it is not possible to assess the potential financial incentives for misbehavior within the protocol. Therefore, the effectiveness of the protocol in ensuring correct behavior cannot be fully evaluated.

Detailed Findings

1. Future block generators can be predicted by the current block generator

The sortition algorithm sets the seed to the signature of the block generator on the seed of the previous block. Therefore, a block generator is able to predict the next block generator. When they will be the generator of the next block (or someone they collude with and can query the signature from), they can also predict the block generator of the block after that one, etc… While this is not a problem for the consensus mechanism itself, it can lead to problems in the application layer and for instance enable multi-block MEV. If an entity controls a large stake, the problem is exacerbated because the probability that the entity produces consecutive blocks increases significantly. Moreover, an attacker could use this information to impact the availability of the next node (e.g., via a DDoS attack) if this is profitable for them.

Recommendation

We recommend making application developers aware of these risks. Applications that are sensitive to multi-block MEV (such as TWAP oracles) should be developed with care and it should be analyzed what the costs and benefits of manipulation are.

Team Response

The team agreed on the following action points:

1. Prevent predictability of following seed when building the current block: this will be achieved by postponing the new seed computation to the Validation step. The new seed will be generated by one of the Validation committee members, named the 'seeder', which is chosen by means of Deterministic Sortition. As in the current protocol, the new seed will be the signature of the appointed seeder on the previous seed. The new seed must be received and agreed upon by a supermajority of the Ratification committee. As a consequence, a Valid Quorum on the candidate block is only obtained when a supermajority of provisioners agreed on both the candidate and the new seed.

This solution prevents the block generators from knowing the next seed, and hence the next generator, when producing the new block, making MEV attacks unfeasible (except when the generator is colluded with the seeder).

The major drawback of this solution is to require two unique provisioners to participate in each iteration in order to have a valid block: the generator and the seeder. This can potentially affect finality. However, it is worth noting that provisioners are always required to be online and active to avoid being slashed due to missed block generation. Thus, it is not expected to have many missing seeders. Nevertheless, slashing can be considered for seeders as it is for generators.

The use of threshold signatures (to generate the new seed) has been considered as well. However, this solution has been discarded due to the need of a setup phase to run at each iteration, which would negatively impact performances.

2. While the above solution is not implemented, developers and users will be warned about the potential implications of the predictability of the next block generator.

3. Future investigation:

- Strengthen network-level anonymity to avoid targeted attacks against block producers

- Reduce the probability of a provisioner being extracted multiple consecutive times

2. Not allowing contracts to perform direct calls may be too restrictive

Functions of contracts can be tagged as `C2C` (contract-to-contract) or `direct`. Those that are tagged with `C2C` can only be called by other contracts (which is used for providing inter-contract services), whereas the ones that are tagged as `direct` can only be called by a user through a transaction. The latter restriction may be too restrictive, as it can render some use cases impossible. For instance, users may have contracts that act on their behalf (e.g., multisig wallets that are implemented as a contract). In such situations, they want to be able to call all functions that a user can call directly through the contract. On other blockchains, such as Ethereum, there have also been attempts to deny direct calls by contracts (on the application level), but they were mostly discarded because of the previously mentioned restrictions considerations.

Recommendation

We recommend reconsidering whether `direct` calls should only be allowed by users and not by other contracts.

Team Response

The distinction between direct and ICC calls has been removed due to inefficiencies in the ICC execution. Therefore, there is no limitation in place that prevents users from delegating call to contracts.

3. Subsidized transactions may be abused by block producers

When contracts pay for gas and charge a fee, it can happen that the fee is insufficient for the execution and transactions need to be subsidized by contracts. This behavior needs to be explicitly enabled by disabling the `insufficient fee safety check`.

While it is prudent that the behavior is not enabled by default, it should be noted that enabling it is very risky and can potentially lead to situations where block producers can make a risk-free profit. If the fee is lower than the reward they get for the execution, they have a strong incentive to spam the contract with transactions until it runs out of funds.

Recommendation

We recommend documenting these risks explicitly and making all application developers aware of it. Moreover, instead of a binary check, more fine grained control could be helpful for developers. For instance, they may want to be able to control how much they subsidize per transaction or per block at most.

Team Response

The team is aware of the issue but considers it a calculated risk that some contract owners are willing to assume under certain circumstances. In particular, this feature gives contract developers the opportunity to make deals off-chain and make contract services available for free to their customers. As such, we expect subsidies to be only available to specific users included in some sort of access list. This fine-grained filtering is left to the developers.

Nevertheless, we agree to properly document the risk of subsidized transactions and warn developers about the inherent risks.

We also agreed to allow developers to set a cap on gas subsidies.

Economic Model

Protocol Overview

Dusk's Economic Model introduces an innovative approach to managing costs within blockchain transactions. Unlike traditional methods, it allows users to pay smart contracts directly, choose their preferred payment method, and shift the gas cost burden from users to contracts. This aligns more with conventional business practices where service providers cover infrastructure costs. To ensure accurate transactions, the Transfer Contract in Dusk's system settles gas and fees simultaneously, enabling smart contracts to define their charging strategies.

The model outlines three different scenarios for handling gas and fees, including traditional user-paid gas, user-paid gas with an additional contract fee, and a novel approach where the contract pays for gas and sets a fee. In scenarios with insufficient gas, the system locks gas by making it unspendable for a set duration to protect the network and contracts.

The protocol ensures that contracts cannot charge lower fees than their gas expenditure, with the exception of so-called "unsafe fee safety". In the latter case, the contract subsidizes or completely pays the user's transaction. This option is introduced for use cases such as (off-chain) subscriptions or a transaction that should be billed as a percentage of the payload.

All consumed gas is always transferred to the block generator for any transactions with gas limits. Additionally, unspent gas is largely (70%) paid back to the user, and the remaining gas (30%) is paid back to the contract.

Maturity

The economic model innovates on the structure of gas payments. Instead of only users paying for gas, contracts can pay for gas. This is not a fundamentally new approach, as for example, the introduction of EIP-2771 on Ethereum allowed for similar use cases – but on the contract level. Nevertheless, such approaches are still novel and have not been tested on a large scale. At the same time and as discussed below, the literature on fee mechanism design is very young and there is no academic consensus on the standards of blockchain fee assessments. The idea to allow contracts to claim a proportion of the unspent gas is experimental but very promising, as it incentivizes contracts to save gas. Nevertheless, it also comes with second-round effects – which are not modeled formally and might need to be tested in the field.
The mechanism that governs the gas price is standard. It is a first-price bid auction, and the introduction of a base fee & tip model comparable to EIP 1559 is discussed and intentionally discarded. The block rewards structure (incentives for attesters vs. block generators), as well

as slashing are not explicitly discussed. A token emission schedule for the first 18-36 years (until the supply limit is reached) is provided.

Detailed Findings

4. Intentional out-of-gas offers the possibility to pay for transactions only if they are successful.

Within the context of Dusk's Economic Model, particularly under Scenario 3 where the contract pays for gas and applies a fee, there exists a notable consideration if the contract runs out of gas. In such cases, the system design refunds the gas, creating a unique incentive mechanism. This could lead to a situation where contracts might be programmed to intentionally run out of gas, ensuring that no fees are paid. For instance, a contract could execute a transaction like an arbitrage operation, with a built-in logic to deplete its gas if the operation is not profitable.

Consequently, the transaction would fail due to insufficient gas, triggering the refund mechanism. This strategy effectively turns the gas cost mechanism into a contingent fee, where the user only incurs a cost if the transaction is profitable. This creates a potential exploit in the economic model, where contracts can be designed to circumvent fee payment under certain conditions. It might lead to unintended consequences for the network's economic stability and resource utilization. For example, this issue could be exploited for denial-of-service attacks, where attackers spam the network with failing transactions for extended periods of time.

Recommendation

We recommend not allowing free or protocol-subsidized transactions in any case, as they might lead to network congestion, spam attacks, or free-riding behavior.

Team Response

The Dusk team proposes to solve this by introducing a mitigation measure, which locks the contract when running out of gas. In particular, the contract is locked until an ‘unlock’ transaction is sent by the owner, filling up the contract’s funds. This measure effectively penalizes a contract running out of gas, making it disadvantageous to do this on purpose.

5. Strategic implications of the unspent gas kickback might still incentivize ineffective contract designs

The contract receives an income of the form $0.3(gas_{limit} - gas_{consumed})$ i.e. 30% of the unspent gas. This idea is promising as the income is inversely related to the consumed gas by the contract. Nevertheless, there are still incentives to write complicated or inefficient contracts to earn excess fees. While developers now are incentivized to minimize the average

$gas_{consumed}$, they can realize additional returns by luring users into setting gas_{limit} to very high values. This can be done by making the gas consumed very volatile or more specifically extremely high in rare instances that are not under the control of the user. As a consequence, users have a precautionary motive to set the gas_{limit} to a high value resulting in an average higher payout for the contract writer.

Recommendation

To prevent such behavior a more resilient incentive scheme could be calibrated. This could take the form $0.3(gas_{limit} - \alpha\, gas_{consumed} - \beta\, \mu_{gas_{consumed}} - \gamma\, \sigma_{gas_{consumed}})$ where $\mu_{gas_{consumed}}$ is a mean, median or percentile of the gas consumed by the contract, $\sigma_{gas_{consumed}}$ is the volatility of the gas consumed by the contract and α,β and γ as weighting parameters. This would disincentivize high average gas consumption, higher percentiles, and high volatility, but would also require keeping track of the contracts spending behavior. Also other reward functions $f(gas_{limit}, gas_{consumed}, \kappa)$, where κ denotes a vector of arbitrary parameters that do not require past values as input parameters might be worth considering e.g. functions that discount large gaps between the limit and the consumed value exponentially such that extreme mismatches do not pay off much more than smaller ones.
As competition among contract developers already disincentivizes ineffective contract designs, it might be worth considering mild measures first before implementing very complex functions.

Team Response

The Dusk team proposes to solve this by removing all intra-contract calls. Consequently, all related incentives were removed, including distributing a portion of unspent gas.

6. Manipulation of gas prices

The fact that all used gas is paid to the block generator enables side payments between users and block producers. This can be used to create fake transactions to manipulate gas prices or make black markets for transactions profitable.

Users can collude with block producers or block producers can write and execute contracts themselves such that all fees or a relevant share of the fee is reimbursed to the user or the block producer, respectively. Such transactions might be used to prioritize a certain user group or simply to inflate gas prices of the current or even future blocks. This issue is linked to the issue Future block generators can be predicted by the current block generator as block generators might collude to increase gas prices for each other.

To enrich the context of this issue we provide a literature review on fee modeling for blockchains and the suboptimality of first-price auctions.

Recommendation

We recommend not distributing all gas consumed to block generators. A simple solution is burning a fraction of the consumed gas, and a non-deflationary redistribution mechanism might be a possible solution as well.

Team Response

The team proposes to solve this by distributing block rewards, including transaction fees, among block generators and block validators. This shall mitigate the gas manipulation strategy described. Additionally, the introduction of the Seeder, described in the reply to Future block generators can be predicted by the current block generator shall mitigate the predictability of future generators.

7. Automatic gas price estimation may be difficult and gameable depending on the implementation

Developers can specify contracts that pay for gas and charge fees. Such contracts indicate a `gas_price` and likely utilize host data such as overall gas spent in the last block, epoch median, or congestion metrics to do so.

It may be difficult for a contract to estimate a good gas price with these metrics. Depending on the exact algorithm and metrics that the contract uses, it could happen that it does not adapt fast enough to changes in the network congestion or is too sensitive to short changes/bursts. Both of these scenarios can lead to transactions that do not get executed for a long time (in which case it is not clear if the contract can overwrite them with a transaction that has a higher priority) or where it pays too much.

Another potential problem is that the algorithms of the contract will presumably be known by all network participants. Depending on the algorithm, this may be exploitable by participants who have an incentive for higher gas prices (i.e., the block generators).

To enrich the context of this issue, we provide a literature review on fee modeling for blockchains and the suboptimality of first-price auctions.

Recommendation

We recommend assisting application developers with setting these values, for instance by letting them specify/query three different gas price levels (e.g. fast, medium, slow) and allowing them to optionally implement their own logic if they really want to. The algorithm for calculating these provided gas price levels should be developed with care and it should be analyzed to determine whether it works under all network conditions and with potentially malicious participants that may have an incentive to game it.

Team Response

The team proposes to solve this by providing a gas estimation algorithm that can be used by all contracts has been developed by Dusk. The gas estimation will be provided as a host function that is available for all contracts. This algorithm can be easily updated in case any gameability risk is discovered. At the same time, contracts can decide to implement their own *gas estimation function if willing to do so. The logic of the gas estimation will be kept secret as a part of a compiled binary. This makes reverse engineering costly.*

8. Total supply limit leads to a shrinking security budget and deflationary economic incentives

Dusk has a capped supply limit. Cryptocurrencies with a capped supply introduce a unique economic model in the digital currency landscape. The idea behind a supply limit is to create scarcity, potentially increasing the value of the currency over time, similar to precious metals like gold. This scarcity can help in preventing inflation which is often seen in fiat currencies without such limits.

However, this model brings with it certain challenges, particularly regarding the security budget of the network. The security budget in the context of cryptocurrencies generally refers to the resources allocated to incentivize block generators, attestors, and other stakeholders who maintain the integrity and security of the blockchain. These incentives typically come from two sources: block rewards (newly minted coins) and transaction fees.

Diminishing Block Rewards: With the capped supply of Dusk, block rewards – the primary incentive for miners or validators – are designed to decrease over time. As the available supply diminishes and reaches its cap, the block rewards eventually reduce to zero. This decrease in block rewards means that the security budget will shrink over time if it is not compensated by an increase in transaction fees or a sharp value increase in the coin.

Reliance on Transaction Fees: As block rewards diminish, these networks must increasingly rely on transaction fees to compensate block generators. The sustainability of this model depends on the volume and value of transactions on the network. If the transaction fees are insufficient to adequately incentivize the network's security providers, it could lead to security vulnerabilities, such as reduced hash power or even susceptibility to 51% attacks.

Economic Implications: The shift from block reward to transaction fee incentives necessitates a high volume of transactions or sufficiently high fees per transaction to maintain the security budget. This shift could impact the network's usage: higher transaction fees might deter users, while too low fees might not sufficiently incentivize miners or validators.

Long-Term Security and Adoption: For capped-supply cryptocurrencies, the long-term security and viability hinge on finding a balance between incentivizing block generators and maintaining user-friendly transaction costs. This balance is crucial for widespread adoption and network resilience.

Recommendation

We recommend reconsidering the total supply limit.

Team Response

The team reassessed the total supply limit and its emission schedule extensively, including mathematical models to assess all risks related to the economic model.

9. High initial supply concentration of the ERC20 token and related risks of proof-of-stake and proof-of-work systems

The documented initial token distribution highlights that 50% of the supply was sold in a private sale, and an additional 12,8% is held by the team and advisors. This might concentrate a large share of the stakeable supply among holders that could collide because they have a business relationship.

In a proof-of-stake mechanism, the concentration of a significant portion of the stake in the hands of a small group of token holders can pose several issues and risks to the network's integrity, security, and decentralization. While a 51% majority can generally undermine more or less all blockchain security assumptions, even a smaller majority of 30% can undermine the integrity of the entire protocol, for instance, if attackers combine eclipse attacks and block generation attacks as described in the seminal paper Stubborn Mining: Generalizing Selfish Mining and Combining with an Eclipse Attack such that non-malicious or non-colluding block generators are no longer incentivized by block rewards. While the mentioned paper focuses on proof-of-work systems with stochastic finality, the concepts can be transferred to proof-of-stake systems with finality gadgets or even proof-of-stake systems with perfect theoretical randomness assumptions. Particularly, an approach based on the predictability of future block producers might be worth considering. For example, see Gaming the Blockchain System: Analyzing Optimal Selfish Mining and Reward Schemes in the Proof-of-Stake Protocol Algorand.

Due to the initial token distribution and the fact that Future block generators can be predicted by the current block generator, we see a risk that a coalition of block generators might combine block generation and eclipse attacks to outperform non-colluding block generators, e.g., by invalidating their blocks and thereby undermine the incentives of the protocol.

Nevertheless, we must qualify our analysis with the remark that young protocols typically start with a very concentrated token or staking distribution.

Recommendation

We recommend fostering decentralization and reconsidering the predictability of block generators. In addition, we recommend research that determines resilient thresholds for combined attack scenarios.

Team Response

The team highlighted, that while the initial token distribution indeed concentrated a large portion of the supply into few owners, this portion has been redistributed in the following years. In particular, the Dusk token has been liquid since 2019 and is currently shared among nearly 20.000 holders, with no evident concentration of wealth on any singular entity. In this respect, note that all initial buyers sold their tokens and that even the Dusk organization *does not own more than 1/10 of the current supply. Information on token distribution can be easily checked on Etherscan.*

10. Slashing and validator incentives are undocumented

The incentives for truthful validation and slashing are undocumented.

Recommendation

We recommend documenting validators' incentives and slashing.

Team Response

The team added documentation for rewards and slashing here: Incentives and Penalties.

11. Attestor incentives are undocumented

The incentives for attestors and ratifications are undocumented.

Recommendation

We recommend documenting incentives for attestors and ratifications.

Team Response

The team added documentation for rewards and slashing here: Incentives and Penalties.

Literature review on fee modeling for blockchains and the suboptimality of first-price auctions

As an informational addendum, we would like to provide a brief literature review of fee mechanism design for blockchains. We do this mainly to make our reasoning on the fee mechanism transparent and highlight that researchers are still debating the frameworks to assess fee design mechanisms. The literature review is particularly relevant for the issues Manipulation of gas prices and Automatic gas price estimation may be difficult and gameable depending on the implementation.

Based on theoretical considerations, fees should be set to be incentive-compatible for users and miners – in layman's terms, this means users should be incentivized to bid their true willingness to pay and miners should be incentivized by the fee and not by potential side payments/bribes from users. In addition, as a precautionary measure, miners should be assumed to be myopic in the sense that they maximize short-term profits, and incentives should take this into account.

Following these considerations, first-price auctions are suboptimal as they result in strategic bidding.
Mechanisms for stable fees are discussed in the literature but are rarely implemented, with EIP-1559 being an exception. Based on the theory, EIP-1559 should have drastically reduced

fee payments and network congestion. However, the empirical evidence for a reduction is rather mild in magnitude as highlighted in Empirical Analysis of EIP-1559: Transaction Fees, Waiting Times, and Consensus Security.

In practice, enforcing side payments can be challenging. Off-chain contracts may be difficult to enforce, but they could be costly when settled by on-chain contracts. Additionally, if the potential profits from Maximal Extractable Value (MEV) opportunities are large enough, and the reputation of the actors is at risk if they collude, these theoretical opportunities of a side payments cartel might not have been exploited to an extent that is significantly visible in the literature on gas prices or for market participants.