# Cryptographic Approaches to Complete Mempool Privacy

This work was done from February-September 2022 as FRP-18

A full report with more mathematical detail is availablehere.

## 1. Mempool Privacy

### MEV & Mempool Privacy

Since the publication of Flash Boys 2.0 in 2019, maximal extractable value (MEV)-related activity on the Ethereum network has grown consistently. Increased MEV-related activity leads to a range of related negative externalities, including higher transaction fees, increased blockspace usage, and network congestion. Such externalities not only result in poor user experience, but form a systemic risk to the network by incentivising miner/validator behaviour that may lead to consensus instability.

MEV extraction takes multiple forms, including frontrunning of arbitrage/liquidation opportunities, censoring attacks, and even potential time-bandit attacks. MEV extractors scan the mempool in search of profit opportunities and then engage in gas bidding wars to seize the opportunity; this led Robinson & Konstantopoulos to dub the mempool a ‘dark forest’

- a treacherous environment in which unsuspecting users are mercilessly exploited for maximal profit[1]

At present, the vast majority of MEV extraction activity is in the form of frontrunning bots or MEV searchers profiting from arbitrage opportunities. Such opportunities are found due to the public nature of the Ethereum mempool. A private mempool, in which transaction details are concealed until they have been committed to a block, would therefore mitigate against a broad range of contemporary value-extraction strategies. However, as discussed in https://eprint.iacr.org/2017/1090.pdf, even a private mempool would still allow for adversarial value extraction in certain situations, such as auction constructs, in which just the knowledge that a concealed bid/transaction has been submitted could lead to altered strategies for other users. In effect, MEV is an unavoidable consequence of intra-user time-dependent information asymmetry.

Despite this caveat, a mempool providing complete privacy for users would constitute a substantial improvement over the status quo. In its current incarnation, Flashbots offers pre-trade privacy, as transactions pre-block inclusion are visible only to relayers and block builders. Complete privacy from all parties is a core design goal and necessary for a trustless system. A range of cryptographic approaches have been proposed to solve this problem, which we outline in §2.

### Properties of a privacy solution

Privacy

A core design requirement for any solution is complete privacy

- transactions are concealed and cannot be viewed by any intermediaries, such as block proposers or builders, until they have been mined on-chain. Complete privacy mitigates against a number of current MEV extraction strategies by concealing the opportunity, including frontrunning such as sandwich attacks. Complete privacy necessarily implies pre-trade privacy and failed trade privacy, which is already provided by Flashbots.

There are two main facets of transaction privacy[2]

- Anonymity

: A transaction cannot be linked to its sender or receiver.

- Confidentiality

: The contents of the transaction is concealed from all intermediaries.

Trustlessness

Ideally, our solution architecture will be trustless - there is no reliance upon a trusted party to set-up or co-ordinate cryptographic protocols. Enforcement of protocol may be through design (e.g. use of commitments within a protocol) or incentivisation mechanisms (e.g. bonded actors). If a trusted setup is required, carefully designed schemes, such as the use of multi-party computation protocols, must be taken during setup to ensure that no parties retain knowledge that could be used to backdoor the system.

Efficiency

A privacy solution should be efficient in terms of:

- Blockspace

- encryption should not greatly increase the blockspace required for each transaction. For example, validity proofs committed on-chain should be as small as possible and we require some mechanism for packing blockspace efficiently.
- Scalability
- The encryption scheme must scale efficiently with both a large number of users and transactions, ideally whilst maintaining lean hardware requirements for encryption and decryption.
- Implementation
- An ideal solution is efficient to implement (e.g. not requiring L1 protocol changes) whilst maintaining compatibility with network incentives and consensus mechanisms. That is, implementation of encryption should not greatly increase the attack surface for generating consensus instability. This includes mitigating against griefing by any party in the protocol and flooding attacks.

Encrypted Ordering

It would be desirable for any privacy solution to be compatible with application of fair-ordering protocols[3]

We therefore require some mechanism for ordering encrypted transactions without

revealing transaction metadata, whilst simultaneously maintaining compatibility with network incentives, such as optimising gas fees.

Metadata Leakage

A completely private mempool would ideally conceal all

data about a transaction, including metadata that may potentially identify an account and therefore engender information asymmetry. A solution requires that transactions with invalid parameters can be invalidated without first being propagated to the entire network and without revealing the parameters themselves.

Optional Encryption

Depending on the encryption scheme used, it may be desirable to allow users to determine whether their transaction is encrypted or not (a feature of other privacy-focused blockchains such as Zcash). For example, users may choose not to encrypt their transaction if the gas fees incurred by encryption outweigh the possible savings generated by concealing the transaction.

Guaranteed Decryption

We require two related guarantees from our encryption scheme:

- Any party delegated with decryption and following the decryption protocol of the scheme is guaranteed to decrypt the transaction (possibly with some time-delay).
- Any user sending an encrypted transaction to the network is guaranteed to have it decrypted eventually

, if it is included within a block.

UX

Ideally, a solution will not impact upon the UX of users interacting with the network - sending an encrypted transaction should not incur significant latency (in the form of block delays to inclusion) or greatly increased gas fees. Non-interactive protocols ensure that synchrony is not required once the transaction has been sent. The inclusion of encryption should be seamless from the end user's perspective.

## Consensus Layer vs Application Layer MEV

In the context of a given block proposal paradigm, we may consider MEV extraction strategies at both the consensus and application layers. Post-merge, at the consensus layer, a range of strategies may be employed that exploit prior knowledge of the block proposer.

Some proposed approaches to MEV minimisation at the consensus layer include:

- Single secret leader elections (SSLE), which mitigate against time-buying attacks and proposer DoS attacks[4]
- Single-slot finality to mitigate against chain reorgs for MEV extraction.

In §2, we we will focus on cryptographic approaches to MEV mitigation at the application layer, either through encrypted transactions or bundles. By concealing transactions, a broad range of currently employed MEV strategies can be

neutralised.

# 2. Cryptographic Approaches to Mempool Privacy

## Multi-party Computation

Multi-party computation (MPC) protocols allow a set of distinct parties to compute joint computation of a function without revealing their private inputs. Each party therefore learns their respective output and nothing else.[5]

A large number of protocols have been designed to implement MPC, satisfying a wide variety of security requirements. All protocols require robustness in the presence of adversaries that control a subset of corrupted parties.

There are a number of models that delineate the range of adversarial behaviour and corrupted parties. For blockchain applications, we must consider malicious adversaries

and an adapative corruption

model, in which previously honest parties may be corrupted during computation. This accurately captures the possibility of an attacker hacking MPC parties and proceeding to arbitrarily deviate from the protocol.

In the case where the number of corrupted parties is unbounded, secure MPC can be achieved, but without fairness or guaranteed output.[6]

These guarantees can be retained in the covert adversaries model, in which a misbehaving adversary is detected with some fixed probability \epsilon

. These protocols can be enhanced with [public verifiability

](https://eprint.iacr.org/2021/366), such that honest parties can prove misbehaviour by adversaries. Such a model could apply in a setting in which MPC parties are bonded, for example.

Mempool privacy using MPC

By applying an MPC protocol, HashCloak have implemented scoring bundles whilst maintaining transaction privacy.

One technical issue with this method of using an MPC protocol for mempool privacy is the requirement to check for transaction or bundle validity. Without this validity check, an attacker could spam the network with invalid transactions as inputs to the protocol. One potential solution is to generate a zero-knowledge proof of bundle validity against the current state, which requires a fully implemented zk-EVM.

## Verifiable Delay Functions

A Verifiable Delay Function (VDF), as defined by Boneh et al.

is a function for which computing an output takes a pre-determined amount of wall-clock time. This output is efficiently verifiable, so verification is asymptotically faster than evaluation. Parallelisation of the computation of a VDF's output will not decrease the time taken to evaluate an output.

Fast verification distinguishes VDFs from other 'slow' functions such astime-lock puzzles, which require time-consuming naive re-evaluation of the function to verify an output. VDFs' parametrised time delay and fast verification allow for a number of useful applications in decentralised systems, such as non-manipulable randomness beacons and computational timestamping.

Mempool privacy using VDFs

To provide mempool privacy, a VDF can be employed as part of a time-delayed commitment scheme in which transactions are committed to, and may only be read by a miner once the desired amount of wall-clock time has elapsed. In this model, the security provided by our choice of delay parameter t

is proportional to the transaction confirmation delay experienced by the end user. For example, if t

is too small, miners can decrypt the transaction whilst it is still in the mempool, thereby allowing MEV opportunities to be uncovered. Conversely, a large value for t

necessarily imposes a significant delay on transaction confirmation and correspondingly, poor UX.

A distinct disadvantage of the VDF approach to mempool privacy is its scalability. As the number of users grows, significant computational resources will need to be expended to compute each individual function and verify outputs. In a Delay Encryption scheme, encryption is based on a unique session identifier

which is changed periodically. This makes Delay Encryption a potentially useful primitive for application in the Ethereum

epoch model.

## Threshold Encryption

One approach that may resolve the issue of deadweight loss caused by encrypted transactions that are not included in a particular epoch is threshold encryption. A threshold encryption scheme consists of a sender and a set of n

receivers, of which at least t

must cooperate to decrypt. In an ideal scheme for blockchain implementation, both the encryption and decryption phases would be non-interactive and the set of receivers may be chosen ad-hoc.

The basic idea is that a transaction is encrypted, with the set of n

receivers chosen upon encryption - the validator committee is a natural choice for the set of receivers. Once an ordered list of transactions has been committed to (e.g. through an on-chain commitment), a subset of t

receivers compute verifiable partial decryptions. These partial decryptions are combined to obtain a complete decryption of the original transaction, which can then be executed. By employing a threshold scheme, the user's issue of non-inclusion by one particular proposer can be mitigated, as any validator within the epoch may decrypt a transaction in cooperation with (t-1)

others.

Mempool privacy using threshold encryption

A threshold encryption scheme integrates well with a validator committee paradigm, as the decryption phase can be consolidated with the consensus mechanism. The cryptographic security of the scheme relies on robust, well-studied cryptographic primitives and scales well, e.g. for use in layer-2 solutions. However, a threshold scheme requires well-designed incentivisation to ensure that a quorum of participants are well-behaved. Incentives mitigate against adversarial behaviour such as off-chain collusion or decryptors griefing by not providing decryption shares. Such mechanisms are feasible in the validator committee paradigm. However, although unlikely with larger committees, off-chain collusion between a quorum of decryptors and a block proposer could allow for pre-finalisation decryption and consequent MEV extraction that would be difficult to both detect and punish, as identifying the contravening entities is non-trivial.

Requiring an on-chain commitment prior to transaction decryption introduces a 1-block execution delay. This negatively impacts UX and introduces price-discovery issues if threshold encryption is not employed on all transactions in the network. However, in principle, applying threshold encryption is opt-in and can be applied only to certain classes of transaction if desired. Such flexibility is advantageous to users and developers. Smart contract implementations of threshold encryption also impose increased gas fees for the user, making optional encryption desirable.

## Zero-Knowledge Proofs

Zero-knowledge (ZK) proofs allow the correctness of a computation to be efficiently verified via a succinct proof without

revealing any information about the inputs of the computation itself. Currently, the majority of ZK development is focused on leveraging succinctness properties to scale blockchains, but an increasing number of privacy focused projects are also deploying ZK proofs.

Using a zero-knowledge proof, a prover

proves some statement to a verifier

. For blockchain applications, we also require that the proof be publically verifiable by any interested party. If we think of a transaction as a function f

with input x

that induces some output state y

, a ZK proof allows us to prove that we know some valid x

such that f(x)=y

for some y

, without having to reveal x

itself.[7]

Although ZK proofs have been utilised by a number of projects to e.g. preserve privacy in the UTXO model, implementing fully private smart contracts is a challenging problem. Approaches such as Zether provide confidentiality at the expense of

restricted contract functionalities. Two main varieties of ZK proofs - SNARKS & STARKS - are currently utilised for blockchains, each with their respective advantages and disadvantages.

**zk-SNARKS**

SNARKs are a family of proof systems that attest to computational integrity (CI). Nitulescu provides a summary of recent developments in SNARK technology. Groth16 benefits from constant proof size and efficient verification at the expense of a trusted setup. More recent SNARK constructions which do not require trusted setups, such as Halo and Aurora, achieve trustlessness at the expense of an increase in proof size.

Mempool privacy using zk-SNARKs

The properties of zk-SNARKs make them potentially useful in achieving the goal of mempool privacy. However, despite their many advantages, there are a number of technical and theoretical issues with implementing SNARKs on Ethereum. Ethereum's modularity means we require a new SRS for each program. General-purpose updateable SNARKs such as Sonic and PLONK allow for arbitrary computation, at the expense of verification efficiency when benchmarked to Groth16. The verifier's task of computing pairing functions is computationally intensive and expensive given Ethereum's gas limitations.

Despite these technical issues, zk-SNARKs may still be useful as part of a modular architecture to achieve complete mempool privacy. Through optimisations such as precompiled contracts that implement pre-defined pairings on certain elliptic curves, it may be feasible to implement zk-SNARKs for dedicated tasks such as solving metadata leakage issues by proving account balance validity for a transaction without first requiring it be gossiped to the network.

More recent SNARK developments that do not require trusted setups, rely on newer cryptographic primitives which have not had the security benefits of years of research. Transparency also comes at the expense of greatly increased proof sizes, meaning higher blockspace requirements. This leads us to zk-STARKs, which retain many of the innovative features of SNARKs, but dispense with the requirement of a trusted setup and utilise well-studied, robust cryptographic assumptions.

**zk-STARKs**

The goal in implementing a STARK is ostensibly the same as for a SNARK: a prover can efficiently generate a proof that attests to the correctness of some computation C

to a verifier, with verification poly-logarithmic in the runtime of C

. STARKs do not require trusted setups.

Mempool privacy using zk-STARKs

A distinguishing feature of STARKs is their minimal cryptographic assumptions, utilising well-studied, robust cryptographic hash functions as opposed to pairings. As stated, STARKs do not require a trusted setup but their scalability and transparency comes at the cost of larger proof sizes compared to the smallest SNARKs (~45kb vs <300B).[8]

It therefore makes sense to use STARKs for proving extremely large computations to amortise the larger blockspace cost of the proof.

A particular technical advantage of STARKs is that by relying on Merkle trees for commitments, arithmetisation can be over any finite domain, allowing for faster proving and verification. The large proof size of STARKs would lead to increased blockspace usage and higher gas fees if used as part of a privacy solution.

## Witness Encryption

Witness Encryption schemes are powerful cryptographic tools that allow an encrypted message to be decrypted by anyone who knows a witness

to some NP relation. This is in contrast to public-key schemes, in which a message is encrypted for a pre-determined decryptor (or set of decryptors). This flexibility makes witness encryption potentially a very useful primitive for blockchain applications. Such a property allows for guaranteed decryption given the satisfying of some defined condition, allowing the construction of modular decryption schemes. For example, witness encryption can theoretically be combine a threshold scheme with delay encryption parameters.

Garg et al.

introduced both the formal definition of witness encryption and a range of applications for the primitive, including building public-key encryption schemes with lightweight key generation algorithms. They also proposed a candidate construction utilising graded encoding systems, although these are impractical to implement for blockchain purposes. in addition, a number of attacks on these systems have been found.

Bartusek et al.

introduce a candidate scheme based on affine determinant programs (ADPs), which are conceptually simpler to work with but still produce impractically large ciphertexts when used to construct a public-key encryption scheme. Witness encryption is therefore an extremely promising primitive which requires further research and development to be efficiently implementable for blockchain uses.

# 3. Summary of Cryptographic Approaches to Mempool Privacy

We summarise the cryptographic approaches discussed in §2, with respect to the design properties outlined in §1. Note that the efficiency of each approach does not take into the account the technical complexity of implementation.

[

Summary

2318×862 69.8 KB

](https://collective.flashbots.net/uploads/default/original/1X/279f5112de1a218c2680a84fe8db4bf80a6b0cb0.png)

## Conclusion

Complete privacy is a critical design goal towards constructing a trustless, transparent and fair MEV extraction ecosystem. Cryptographic approaches can minimise trust assumptions across the range of participants in the MEV ecosystem. As outlined, there are a number of theoretical and practical challenges to implementation of complete privacy for searchers, builders, and end users. Each of the methods outlined in §2 exist along multiple trade-off axes.

In terms of practical implementation, threshold schemes arguably represent the most efficient solution at present. There exist efficient DKG and threshold encryption schemes that allow for practical encryption with only small latency penalties. These benefits are achievable with an honest majority trust assumption, with the downside that collusion amongst decryptors is hard to police. Implementation of such a scheme requires careful consideration of incentives of network participants, although much of this has been completed in prior works.

Although currently impractical, witness encryption represents perhaps the most streamlined solution to achieving mempool privacy. At present, the proposed schemes are not efficient enough for this application and their security relies on novel hardness assumptions that have not been as thoroughly researched as for more established primitives. We therefore propose further security analysis of existing witness encryption schemes and development of more efficient schemes as a direction for further research.

1. Ethereum is a Dark Forest - Paradigm↩
2. Navigating Privacy on Public Blockchains ↩
3. Order-Fairness for Byzantine Consensus ↩
4. Secret non-single leader election - Proof-of-Stake - Ethereum Research↩
5. Secure Multiparty Computation (MPC) ↩
6. https://dl.acm.org/doi/10.1145/12130.12168 ↩
7. https://vitalik.ca/general/2021/01/26/snarks.htm ↩
8. GitHub - matter-labs/awesome-zero-knowledge-proofs: A curated list of awesome things related to learning Zero-Knowledge Proofs (ZKP). ↩