A Novel Redesign of the Blockchain

Solving the Challenges of Miner Extractable Value, Front-running Attacks, and Transaction Throughput through Separation of Concerns and Duties.

Adit Patel Theresa Garcia

September 3, 2021

Abstract

We propose a novel blockchain to solve the primary issues inherent to extant blockchains. Our proposal allows transactions at scale, arbitrarily complex contract execution with self-optimizing timeouts, and drastically limits conflicts of interest by segregating duties among three specialized nodes, each responsible for a mutually exclusive, collectively exhaustive portion of the transaction lifecycle. We achieve our goals by utilizing hashed transaction information both in the mempool and during mining operations by a network of specialized "mining nodes". These hashed ledger entries in a mined block are then appended and executed by a separate stand-alone network of "clearing nodes" that resolve the block. Finally, a network of "commit nodes" orchestrates the whole process and is responsible for committing any new additions to the blockchain. Each of these networks is fully decentralized and has been constructed to ensure there are no exploitable conflicts of interest. We achieve this while allowing network participants the ability to specialize into a role best suited for their system architecture and hardware.

Contents

1 Current Landscape and Challenges

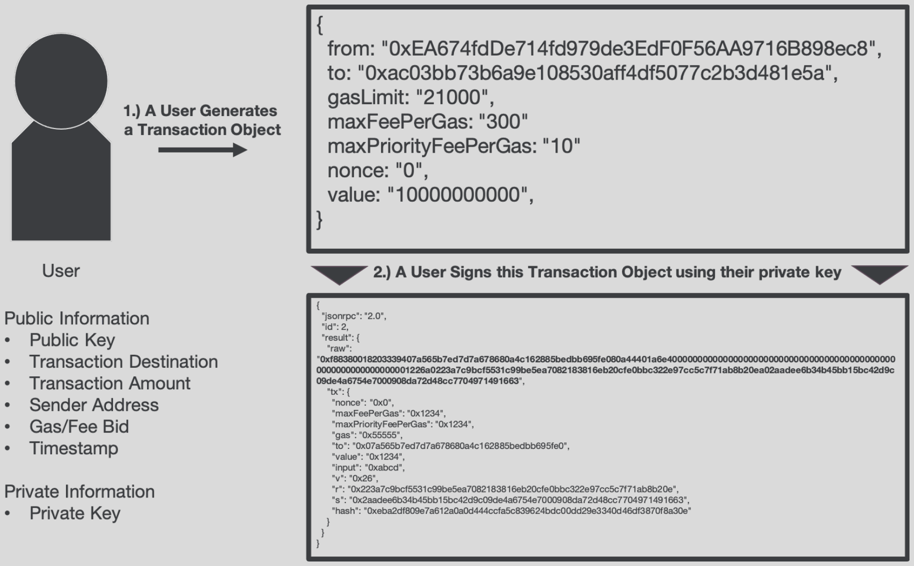
The advent of the blockchain has created entirely new frameworks and possibilities for interacting across networks in a trust-less, transparent, and decentralized manner. It's continued development and the exploration of its use-cases has led to the creation of Decentralized Exchanges (DEXs). These novel exchanges allow users to transact directly with one another, removing the need for a central authority or market maker. However, the transparent nature of the mempool enables malicious actors to execute attacks such as front-running trades on these DEXs. Front-running is an attack where a user gains access to privileged knowledge, such as the intention to enter a trade, and executes a similar trade before the victim, causing them to pay inflated prices which the attacker profits from. Front-running and its effects are becoming more apparent on DEXs and are costing users hundreds of millions of dollars per month.

Today, this has fully developed into sharp fights on the blockchain where arbitrage bots, front-running bots, and miners all compete to claim a piece of the potential profits from these attacks. Many solutions have been raised, but as they cannot change the fundamental nature of the blockchain, none have been 100% effective.

1.1 A Brief Summary of the Current Blockchain Mining Process

The fundamental logic and operations of a blockchain haven't significantly changed since originally being implemented by Satoshi Nakamoto. Here is a visual recap of this process.

A user wishing to transfer money, make a trade, or buy an NFT creates a new transaction object.


1.) A User Generates a Transaction Object
2.) A User Signs this Transaction Object using their private key
User
Public Information
• Public Key
• Transaction Destination
• Transaction Amount
• Sender Address
• Gas/Fee Bid
• Timestamp
Private Information
• Private Key

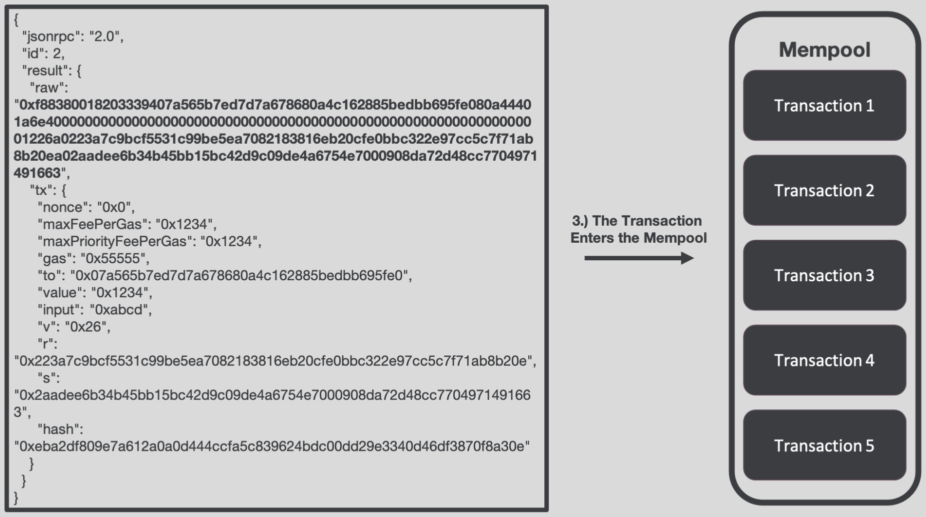
```
{
 from: "0xEA674fdDe714fd979de3EdF0F56AA9716B898ec8",
 to: "0xac03bb73b6a9e108530aff4df5077c2b3d481e5a",
 gasLimit: "21000",
 maxFeePerGas: "300"
 maxPriorityFeePerGas: "10"
 nonce: "0",
 value: "10000000000",
}
```

```
{
  "jsonrpc": "2.0",
  "id": 2,
  "result": {
    "raw":
"0xf88380018203339407a565b7ed7d7a678680a4c162885bedbb695fe080a44401a6e4000000000000000000000000000000000000000
000000000000000000001226a0223a7c9bcf5531c99be5ea7082183816eb20cfe0bbc322e97cc5c7f71ab8b20ea02aadee6b34b45bb15bc42d9c
09de4a6754e7000908da72d48cc7704971491663",
    "tx": {
      "nonce": "0x0",
      "maxFeePerGas": "0x1234",
      "maxPriorityFeePerGas": "0x1234",
      "gas": "0x55555",
      "to": "0x07a565b7ed7d7a678680a4c162885bedbb695fe0",
      "value": "0x1234",
      "input": "0xabcd",
      "v": "0x26",
      "r": "0x223a7c9bcf5531c99be5ea7082183816eb20cfe0bbc322e97cc5c7f71ab8b20e",
      "s": "0x2aadee6b34b45bb15bc42d9c09de4a6754e7000908da72d48cc7704971491663",
      "hash": "0xeba2df809e7a612a0a0d444ccfa5c839624bdc00dd29e3340d46df3870f8a30e"
    }
  }
}
```

Figure 1: A user generates and signs a transaction object.

There are a few notable elements to this transaction object. First, it is signed by the users private key and can only be decoded by their public key, thus proving their identity. Second, once decoded, all of the details are stored in plain-text, meaning it is public for anyone to see. This detailed information includes 2 important fields; the amount and the recipient address.

After initialization and signing, this transaction object is transmitted to the mempool.



```
{
  "jsonrpc": "2.0",
  "id": 2,
  "result": {
    "raw":
"0xf88380018203339407a565b7ed7d7a678680a4c162885bedbb695fe080a4440
1a6e40000000000000000000000000000000000000000000000000000000000
01226a0223a7c9bcf5531c99be5ea7082183816eb20cfe0bbc322e97cc5c7f71ab
8b20ea02aadee6b34b45bb15bc42d9c09de4a6754e7000908da72d48cc7704971
491663",
    "tx": {
      "nonce": "0x0",
      "maxFeePerGas": "0x1234",
      "maxPriorityFeePerGas": "0x1234",
      "gas": "0x55555",
      "to": "0x07a565b7ed7d7a678680a4c162885bedbb695fe0",
      "value": "0x1234",
      "input": "0xabcd",
      "v": "0x26",
      "r":
"0x223a7c9bcf5531c99be5ea7082183816eb20cfe0bbc322e97cc5c7f71ab8b20e",
      "s":
"0x2aadee6b34b45bb15bc42d9c09de4a6754e7000908da72d48cc770497149166
3",
      "hash":
"0xeba2df809e7a612a0a0d444ccfa5c839624bdc00dd29e3340d46df3870f8a30e"
    }
  }
}
```

Figure 2: The unconfirmed transaction is transmitted to the mempool where everyone can view the actions of the the user.

Here in the mempool, all transactions are public including their details. A malicious actor monitoring the mempool can immediately take action once they find an attractive target, e.g. a user initiating a large transaction with a decentralized exchanges' smart contract.

Miners now select the transactions they wish to include in a block from the mempool. Generally, an honest miner should choose the highest fees offered first, but they are not obligated to. They are free to add whichever transaction from the mempool in whatever order they wish to a block. The value of the power to choose which transactions to include and in which order is termed miner extractable value (MEV). By abusing this power freely, miners and bots have the ability to generate hundreds of millions of dollars of profit with virtually no risk or limitations.

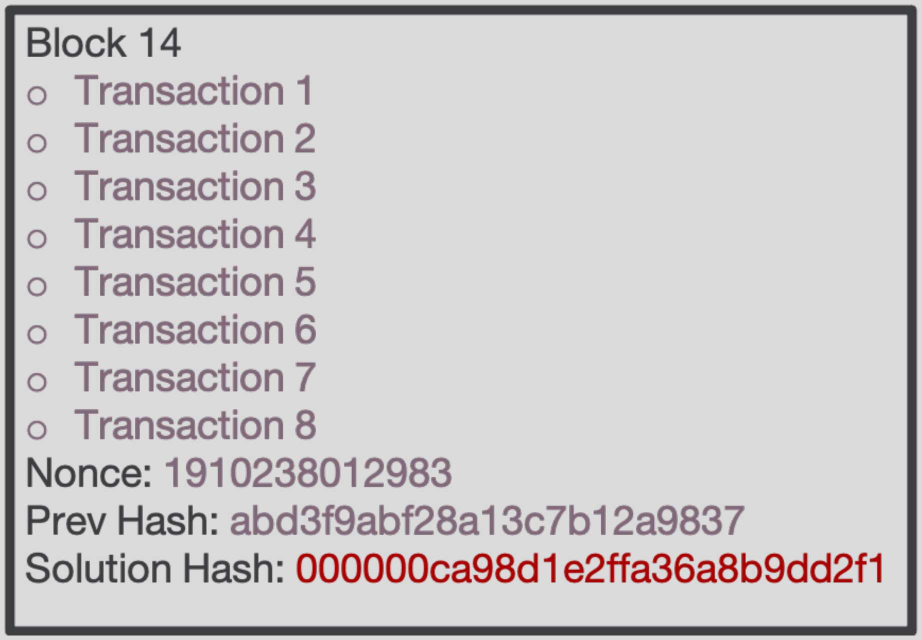


Figure 3: Blocks have hashes that tie all of the blocks together in a chain and can be verified. That's because the hash of any block is generated from a combination of the transactions in the block, the previous hash, and a mined nonce.

Transactions get added to the blockchain through this structure, a "block". A block contains many transactions. In a proof-of-work blockchain, these transactions are appended together with the hash from the previous block and a nonce to generate a new hash. It is the responsibility of the miners to find a hash that satisfies the current difficulty function. The difficulty comes from the number of 0s that must form the beginning digits of the hash.

Note: Extant blockchains refer to what we are calling the solution hash as the block or block header hash. We refer to it as the solution hash as we'll be forming our block header hashes in a different way in our proposal.

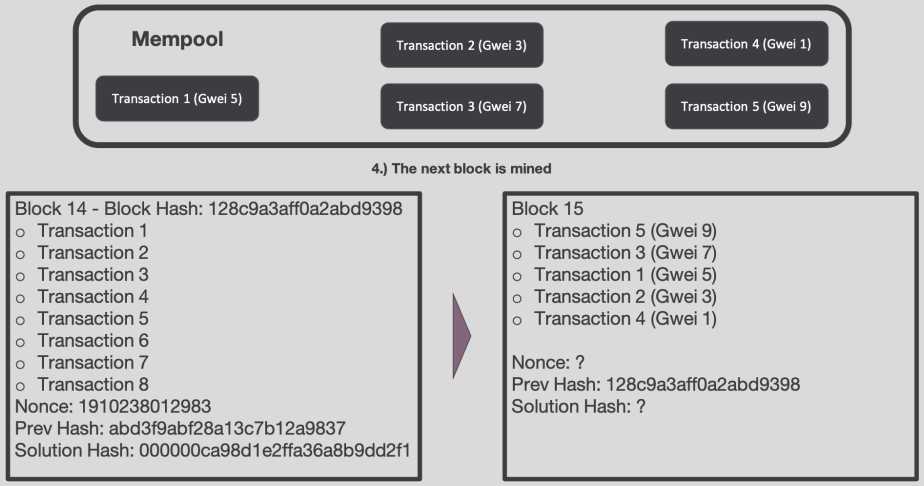


Figure 4: To form the next block, the miners select transactions to include from the mempool based on their bidded gas, and attempt to find a nonce that satisfies the difficulty.

Sticking with proof-of-work for the time being, miners guess nonces until the a hash that satisfies the difficulty function is found. At that point, the node broadcasts out the block to the network which verifies it and adds it to the chain.

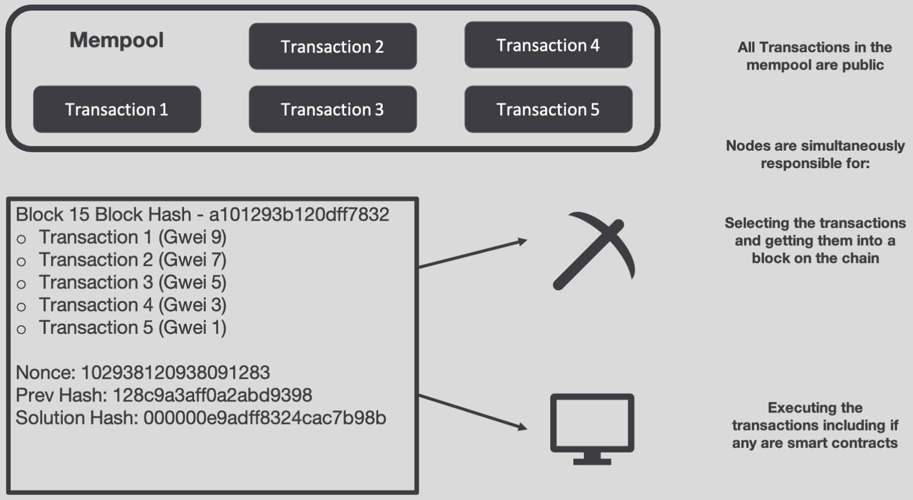

Mempool
Transaction 2
Transaction 4
Transaction 1
Transaction 3
Transaction 5
All Transactions in the mempool are public
Nodes are simultaneously responsible for:
Block 15 Block Hash - a101293b120dff7832
Transaction 1 (Gwei 9)
Transaction 2 (Gwei 7)
Transaction 3 (Gwei 5)
Transaction 4 (Gwei 3)
Transaction 5 (Gwei 1)
Nonce: 102938120938091283
Prev Hash: 128c9a3aff0a2abd9398
Solution Hash: 000000e9adff8324cac7b98b
Selecting the transactions and getting them into a block on the chain
Executing the transactions including if any are smart contracts

Figure 5: Once a Nonce is found, the miner broadcasts the new block, earning the amount of gas bid multiplied by the amount of it has used. Important Note: The miner also simultaneously executed all of the transactions, including smart contracts. In today's blockchain, mining and executing are combined functions both resolved immediately at the time of mining.

We wish to draw attention to something very specific about this process. Specifically the concept of "clearing". Clearing refers to the process of settling transactions, trades and funds. In finance, it is an exceedingly important process, so much so that the process is handled by a specialized intermediary, termed a clearing house. These institutions are widely regarded as removing any conflicts of interest and by performing clearing as a separate function, making traditional financial markets more efficient.

When it comes to extant blockchains, all clearing is handled by the node at the time it mines a new block. If there is a smart contract as part of the block, no matter how complicated it is, it is executed by the node when it finishes mining the block. Fundamentally, in the modern blockchain, the process of "clearing", "committing" and "mining" are seen as a single function by most blockchain implementations. This establishes a low upper bound on contract complexity, block size, and resolution time.

1.2 Current Challenges

Extant blockchains face several challenges in the modern era. The three we believe are the most pressing are the challenges of miner extractable value, front-running attacks, and transaction throughput.

From Figure 4 above, you can see that miners have an incredibly broad latitude when it comes to selecting and ordering the transactions in a block. By changing the included transactions or their execution order, individual transactions in that block can gain or lose significant amounts of value. Miner extractable value, refers to the economic value of this power. Entire tokens have been created to help miners directly auction this power to attackers or victims, and there are proposals in the Ethereum community to make MEV auctions or MEVAs an integral part of the network itself. It's important to note that there is extensive discourse on the subject and some believe that the cure of auctions is worse than the disease. In either case, it is clear that MEV is a major challenge for all extant blockchains and is a major limiting factor to efficient transactions on decentralized exchanges.

Speaking of which, one of the most prominent uses of a complex smart contract is the decentralized exchange, or "DEX". A decentralized exchange maintains liquidity pools with automated market makers that are accessed through smart contracts. The logistics of their operations are beyond the scope of this paper. Essentially they are an ownerless exchange where users can exchange one cryptocurrency for another. Our primary concern with them is that similar to centralized or traditional exchanges, trades that are executed first receive better pricing.

As an example, if a malicious agent identified that a bank was going to make a large purchase of Apple stock, it would be possible to use that information to generate a profit. All the malicious agent would have to do is purchase Apple stock before the bank does, even if by microseconds. Their purchase would cause the price to move up for when the bank buys. Our malicious agent can sell immediately after the bank concludes its trade at an inflated price, pocketing some of the banks money as a risk-free profit.

The challenge for decentralized exchanges in particular, is that these transactions, while in the mempool, are unencrypted and may take minutes to execute. Malicious bots can and do scan the mempool transactions for trading smart contracts. If they identify a transaction heading for a decentralized exchange, they can front run and back run the transaction in the same block by offering fees slight above and slightly below the targeted transaction or even pay a miner to build them such a block by splitting the profit. When the transaction is incorporated into a block, the smart contract receives the malicious front-run trade first. Immediately after, the victim's order then executes at a highly inflated price. The malicious actor then follows up with a back-run trade, pocketing a risk-free profit from the victim by forcing them to transact at an artificially high price.

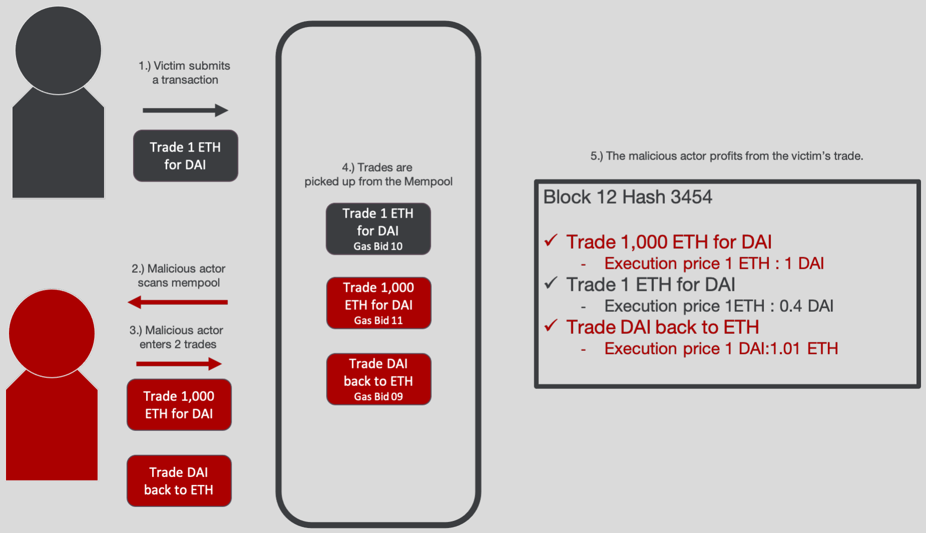

1.) Victim submits a transaction
Trade 1 ETH for DAI
2.) Malicious actor scans mempool
3.) Malicious actor enters 2 trades
Trade 1,000 ETH for DAI
Trade DAI back to ETH
4.) Trades are picked up from the Mempool
Trade 1 ETH for DAI Gas Bid 10
Trade 1,000 ETH for DAI Gas Bid 11
Trade DAI back to ETH Gas Bid 09
5.) The malicious actor profits from the victim's trade.
Block 12 Hash 3454
✓ Trade 1,000 ETH for DAI
- Execution price 1 ETH : 1 DAI
✓ Trade 1 ETH for DAI
- Execution price 1ETH : 0.4 DAI
✓ Trade DAI back to ETH
- Execution price 1 DAI:1.01 ETH

Figure 6: An Illustration of the logistics of a "Sandwich attack".

Today, there is a balance created by the high fixed fees assessed by the blockchains. In an assessment of profitability, [the lower bound of transaction size for a profitable attack was 15 ETH]. However as technology advances and transaction fees fall, the scope of viable targets will dramatically increase.

However, most blockchains have fundamental limits to transaction throughputs. In the extant implementation, a single node, that is optimized solely for mining, as that is how they win the right to mine a block, is responsible for mining, clearing, and committing a block. This places a low upper bound on how large and complex an individual block can be, as it still needs to be executed by the mining node in a reasonable time. If a smart contract took 10 seconds to execute, it would place undo strain on the mining node. In an effort to rectify this, the clearing of an entire ethereum block is limited to an execution time of 0.5-1.5 seconds on an average consumer CPU. This bounds the amount of transactions a given block can contain. Increasing the block size would allow more transactions, but would also make it difficult for a mining node to clear all the transactions within a reasonable time.

2 Proposed Solution

When confronted with similar challenges, traditional financial institutions developed an extensive infrastructure network and introduced separation of concerns. A broker, exchange, clearinghouse and market maker all own clear discrete parts of the transactions lifecycle, but none owns all of it. Our proposal is an implementation of the full, traditional financial infrastructure in a decentralized way to efficiently clear and settle transactions. We explain the philosophy to implementing this infrastructure and separation of concerns in a following subsection.

In order to avoid conflicts of interest, we also incorporate a decentralized version of "Dark Pools". In traditional finance, these are liquidity pools that obfuscate the nature of the transaction enabling large financial orders to be completed without a malicious actor front running the trade. However, blockchains are, by their nature, public, immutable ledgers. The transaction cannot itself be encrypted as both parties need to verify each other's credentials as well as the integrity of the block chain.

In order to combat these malicious attacks and allow for efficient decentralized exchanges, we propose a novel blockchain implementation that hashes the transaction until it has been included into the blockchain along with clearing and committing nodes that resolve the block. We have attempted to remain as true to the original vision of Satoshi Nakamoto as possible and remain committed to the transparent, immutable P2P ledger that they envisioned.

In summary, our proposed solution has two fundamental differences from the current implementation of cryptocurrencies. First, transactions are hashed by SHA256 before entering into the mempool, obfuscating the details of the trade. Second, we introduce the doctrines of segregation of duty and separation of concerns by creating three networks of specialized nodes, each responsible for a separate portion of the lifecycle of a transaction with careful checks and balances between them. We propose distinct ownership over what we believe are the three fundamental stages of a transaction; mining (origination), clearing (execution), and committing (closing).

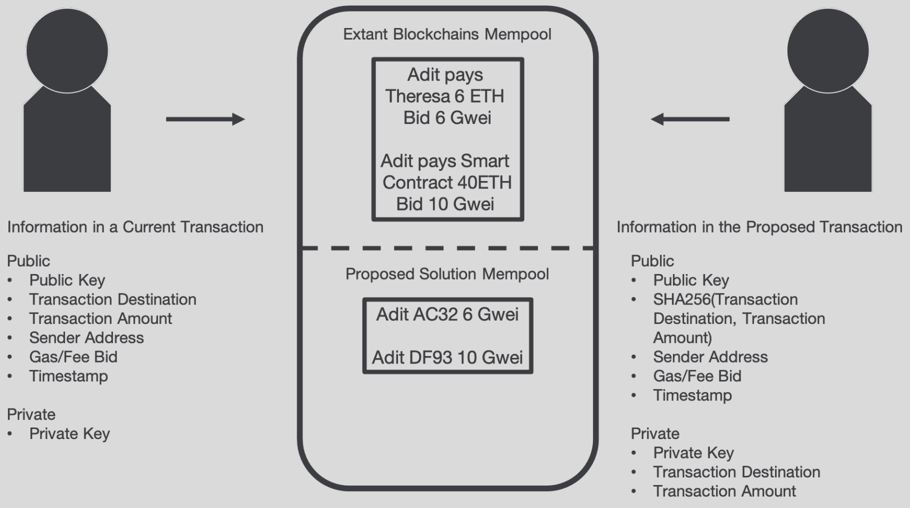

Extant Blockchains Mempool
Adit pays Theresa 6 ETH Bid 6 Gwei
Adit pays Smart Contract 40ETH Bid 10 Gwei
Proposed Solution Mempool
Adit AC32 6 Gwei
Adit DF93 10 Gwei
Information in a Current Transaction
Public
Public Key
Transaction Destination
Transaction Amount
Sender Address
Gas/Fee Bid
Timestamp
Private
Private Key
Information in the Proposed Transaction
Public
Public Key
SHA256(Transaction Destination, Transaction Amount)
Sender Address
Gas/Fee Bid
Timestamp
Private
Private Key
Transaction Destination
Transaction Amount

Figure 7: Our proposed changes to the mempool

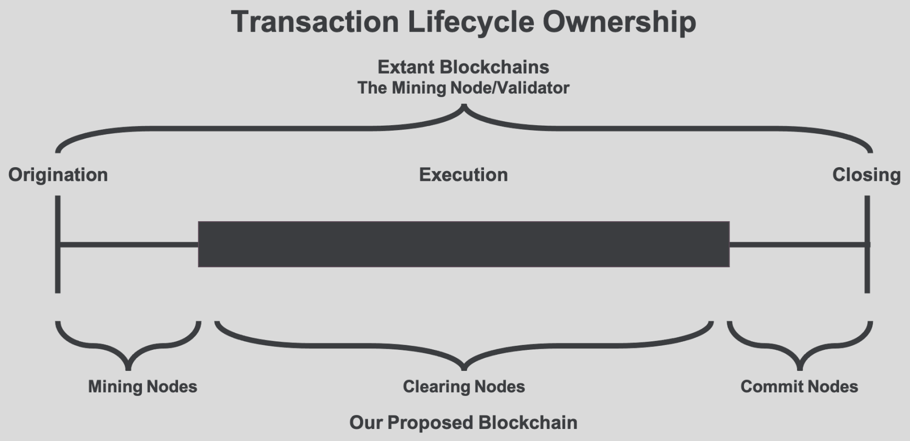
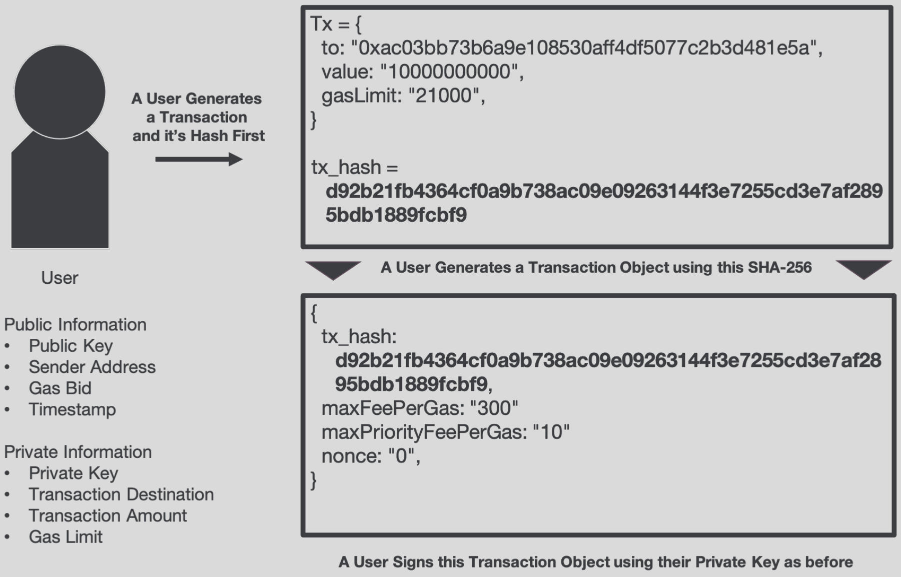


Figure 8: Ownership over the Transaction Lifecycle.

2.1 The Altered Mining Process

Let's begin with the altered mining process. For this illustrative example we are going to use the Ethereum blockchain as a generic reference. That is, a proof-of-work block chain with gas fees. It is purely illustrative and our proposal is agnostic of how new blocks are mined or fees are assessed. The logistics for an end user and a miner are very similar to existing blockchain implementations and our proposal can be adapted to either a proof-of-stake or a proof-of-work blockchain.

Now let's start with a user who wishes to initiate a transaction. As before, they will need to construct a transaction object containing their address, public key, and a fee bid.


A User Generates a Transaction and it's Hash First
User
Public Information
• Public Key
• Sender Address
• Gas Bid
• Timestamp
Private Information
• Private Key
• Transaction Destination
• Transaction Amount
• Gas Limit
A User Generates a Transaction Object using this SHA-256
A User Signs this Transaction Object using their Private Key as before

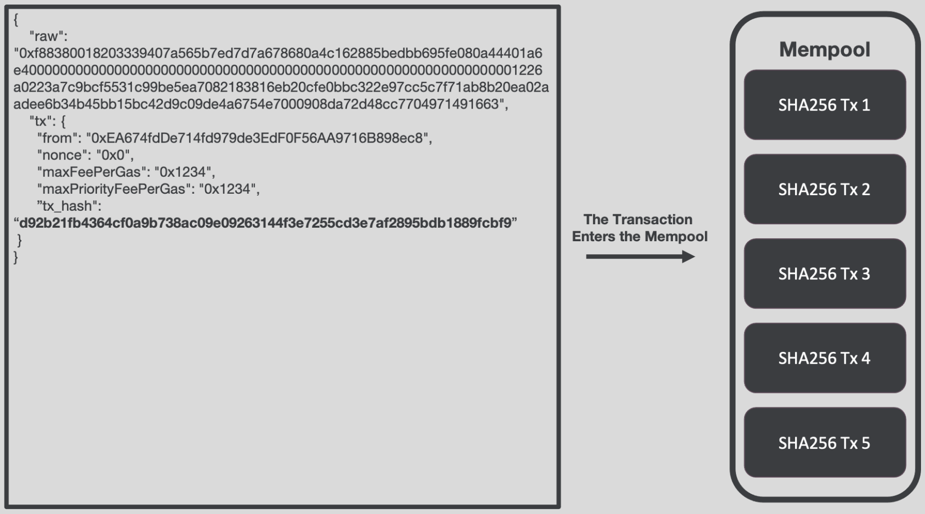
```
Tx = {
  to: "0xac03bb73b6a9e108530aff4df5077c2b3d481e5a",
  value: "10000000000",
  gasLimit: "21000",
}

tx_hash =
  d92b21fb4364cf0a9b738ac09e09263144f3e7255cd3e7af289
  5bdb1889fcbf9
```

```
{
  tx_hash:
    d92b21fb4364cf0a9b738ac09e09263144f3e7255cd3e7af28
    95bdb1889fcbf9,
  maxFeePerGas: "300"
  maxPriorityFeePerGas: "10"
  nonce: "0",
}
```

Figure 9: A user generates a transaction with the typical parameters, however this transaction object is then hashed using the SHA256 algorithm. The user keeps the raw details of the transaction secret and encrypts the transaction object containing the SHA256 hashed transaction details with their private key.
Note: For the sake of brevity and clarity in this and all other figures, we've simplified operational details. As an example, the user would need to append a one-time nonce before the transaction is hashed.

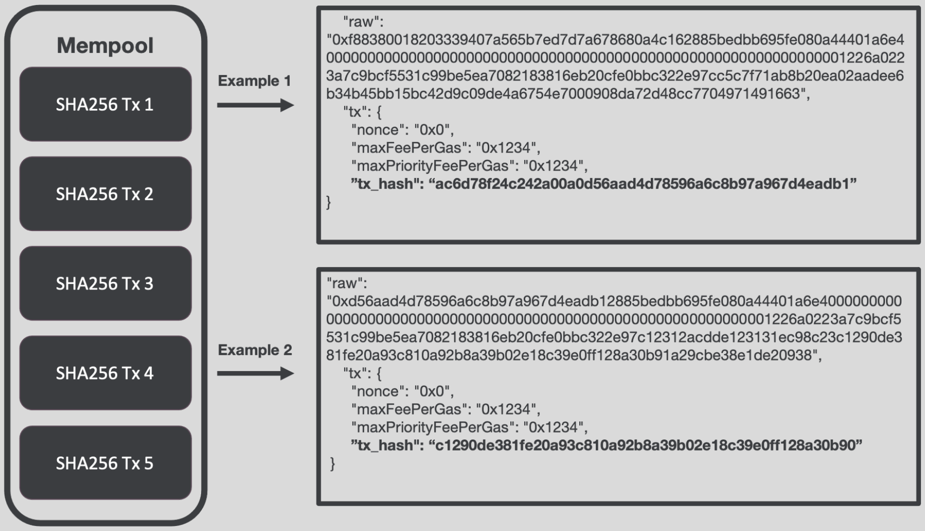
Traditionally, this transaction object includes the plain-text details of their transaction. In our proposal, users instead use SHA256 to hash any sensitive transaction details together. In this example, it's the three details; the transaction amount, the transaction destination, and the gas limit. This hash is what is appended to the transaction object. The plain-text details are kept secret and reserved by the user.



```
{
   "raw":
"0xf88380018203339407a565b7ed7d7a678680a4c162885bedbb695fe080a44401a6
e400000000000000000000000000000000000000000000000000000000000000001226
a0223a7c9bcf5531c99be5ea7082183816eb20cfe0bbc322e97cc5c7f71ab8b20ea02a
adee6b34b45bb15bc42d9c09de4a6754e7000908da72d48cc7704971491663",
   "tx": {
     "from": "0xEA674fdDe714fd979de3EdF0F56AA9716B898ec8",
     "nonce": "0x0",
     "maxFeePerGas": "0x1234",
     "maxPriorityFeePerGas": "0x1234",
     ”tx_hash":
“d92b21fb4364cf0a9b738ac09e09263144f3e7255cd3e7af2895bdb1889fcbf9”
 }
}
```

Figure 10: The unconfirmed transaction is transmitted to the mempool.

The user signs this transaction object with their private key. This transaction object still has in plain-text the gas bid, user address, and public key. With this information, a mining node can still verify the identity of the user by confirming that they do indeed hold the private/public key pairing and still prioritize which transactions to include according to bids. The transactions the miner adds in this block MUST be ordered by gas price followed by the transaction hash in descending alphabetical order. If a user wishes to be assured of an early placement in the block, they can change their transaction nonce to mine a low transaction hash.



```
   "raw":
"0xf88380018203339407a565b7ed7d7a678680a4c162885bedbb695fe080a44401a6e4
0000000000000000000000000000000000000000000000000000000000000001226a022
3a7c9bcf5531c99be5ea7082183816eb20cfe0bbc322e97cc5c7f71ab8b20ea02aadee6
b34b45bb15bc42d9c09de4a6754e7000908da72d48cc7704971491663",
   "tx": {
     "nonce": "0x0",
     "maxFeePerGas": "0x1234",
     "maxPriorityFeePerGas": "0x1234",
     ”tx_hash": “ac6d78f24c242a00a0d56aad4d78596a6c8b97a967d4eadb1”
}
```

```
"raw":
"0xd56aad4d78596a6c8b97a967d4eadb12885bedbb695fe080a44401a6e4000000000
000000000000000000000000000000000000000000000000001226a0223a7c9bcf5
531c99be5ea7082183816eb20cfe0bbc322e97c12312acdde123131ec98c23c1290de3
81fe20a93c810a92b8a39b02e18c39e0ff128a30b91a29cbe38e1de20938",
   "tx": {
     "nonce": "0x0",
     "maxFeePerGas": "0x1234",
     "maxPriorityFeePerGas": "0x1234",
     ”tx_hash": “c1290de381fe20a93c810a92b8a39b02e18c39e0ff128a30b90”
 }
```

Figure 11: Unconfirmed transactions in the mempool.

Now, the mining node, along with any agent accessing the mempool, isn't able to read the transaction payload as it is simply the SHA256 hash of the transaction details. This forces miners to be agnostic when selecting which transactions to mine into the next block.

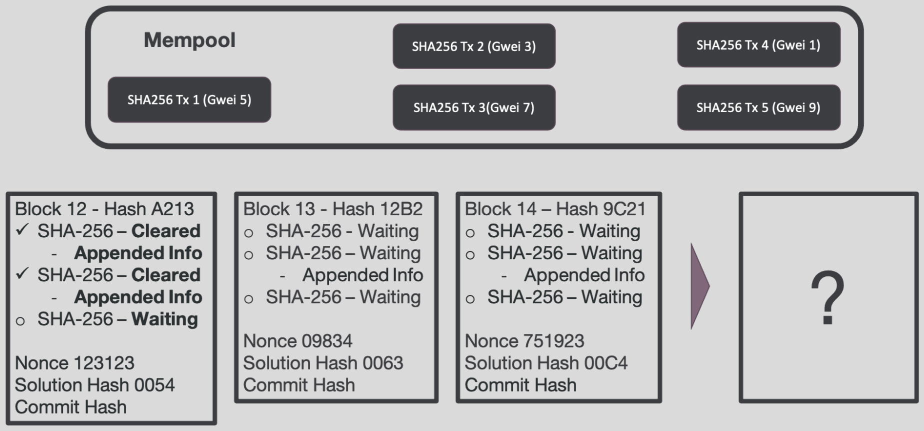


Figure 12: Miners only have access to the transaction hash.

Mining nodes now begin the process of mining blocks like usual. Here we have a 14 block chain and the miners have just found the solution hash to block 15. They are preparing to broadcast the block to the commit nodes for committing. Before we go over that process, let's cover the differences between a traditional block and one from our blockchain.

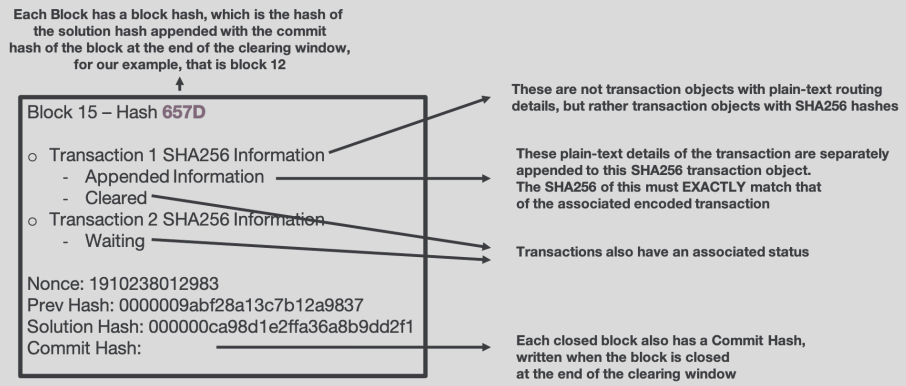

Each Block has a block hash, which is the hash of the solution hash appended with the commit hash of the block at the end of the clearing window, for our example, that is block 12
Block 15 – Hash 657D
○ Transaction 1 SHA256 Information
- Appended Information
- Cleared
○ Transaction 2 SHA256 Information
- Waiting
Nonce: 1910238012983
Prev Hash: 0000009abf28a13c7b12a9837
Solution Hash: 000000ca98d1e2ffa36a8b9dd2f1
Commit Hash:
These are not transaction objects with plain-text routing details, but rather transaction objects with SHA256 hashes
These plain-text details of the transaction are separately appended to this SHA256 transaction object. The SHA256 of this must EXACTLY match that of the associated encoded transaction
Transactions also have an associated status
Each closed block also has a Commit Hash, written when the block is closed at the end of the clearing window

Figure 13: A block from our proposed blockchain.

There are a few key differences between our block and a traditional one. First, each ledger entry is just the SHA256 hash of a transaction, and does not currently have any plain-text transaction details. Second, each ledger entry has two associated fields; a status flag and optionally appended plain-text transaction details. The hash of this plain-text transaction detail must match the hash of the ledger entry in a valid block. We'll cover statuses in a moment, but the block also has one other field, a commit hash. We cover the commit process in the next section, but for now know that this commit hash exists and can be validated for all committed blocks. The block also has a traditional block hash as you would find in any other blockchain however instead of just being the solution hash, it is added to the block by the commit node network as part of the commit process. This is why we refer to it as the solution hash as it no longer serves as our block hash. Note: this process is very similar to extant implementations as you can still generate the full Merkle tree of the transactions from the transactions hashes.

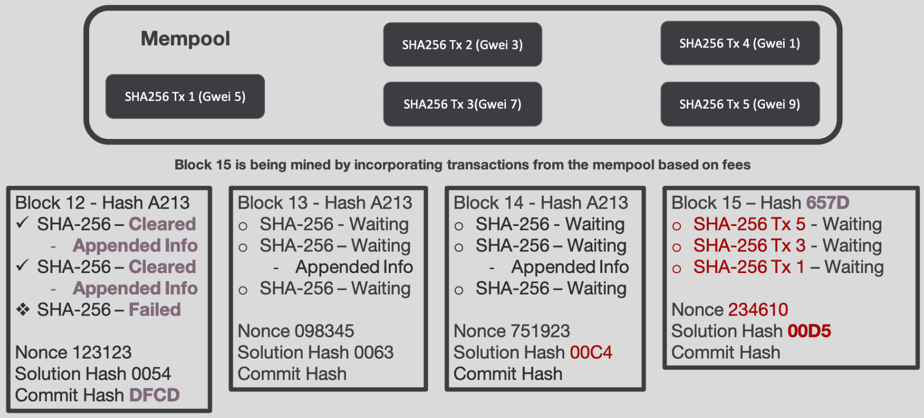


Figure 14: Committing block 15 closes block 12. Here, in red, are transactions and hashes assigned by the mining node. In purple are the transactions finalized and hashes assigned by the commit node.

After a mining node mines this block, whether through proof-of-work or proof-of-stake, it initiates the commit process by transmitting this new block to the commit nodes' network. We'll detail the process of committing that is performed by a commit node in a following section. For now, the relevant part of that process is that these commit nodes verify the block, write its block hash, and add it to the blockchain.

The newly added block at this point resembles a standard block from any other chain except each transaction entry is a SHA256 hash with a status of "waiting" instead of plain-text transaction data and the block currently has a null commit hash.

However, these blocks are currently useless. No one can read them so while they are immutably on the chain, the actual transaction hasn't been cleared or committed yet.

2.2 The Clearing Window

You might be wondering how we can now get the plain-text details to execute the transaction.

After this newly created block is added to the blockchain and broadcast to the network by the commit nodes, the wallet who initiated the transaction has a window of time. We term this the "clearing window" and it is the amount of time, measured in blocks, that a wallet has to confirm the actual details of the transaction. If you are following the illustrative example, because the mining of block 15 committed block 12, it is a 3 block window.

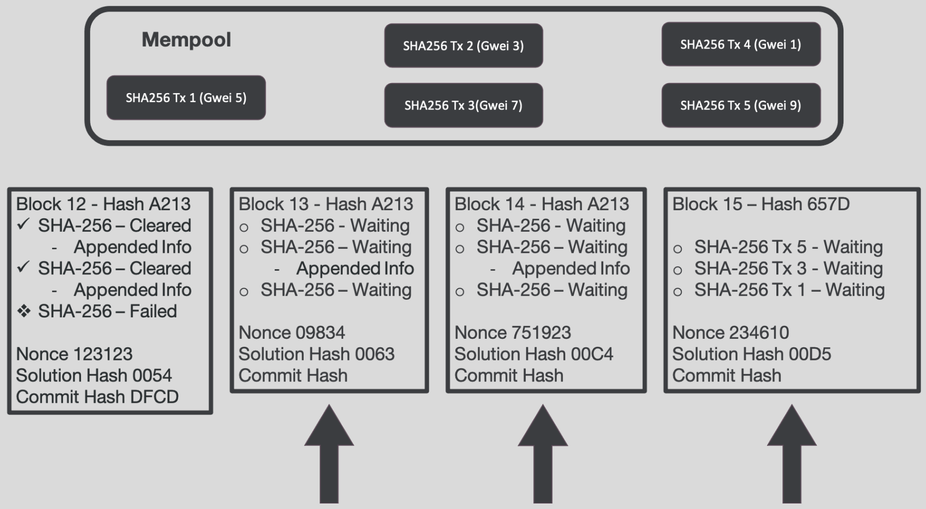


Figure 15: Up until block 15 was mined, users could have appended their plain-text transaction details to blocks 12, 13 and 14. After block 15 is mined, users can now append their transactions to block 13, 14 and 15.

Confirming the details is simple, the user submits the plain-text transaction details and nonce. The network calculates the SHA256 hash of the included transaction information. If the network confirms this value matches a SHA256 hash in an open block, the raw transaction data is appended to that block under the hash. If the user does not broadcast the plain-text transaction details in time, or does not have the funds to perform the transaction, the transaction fails when the the block is committed. Note here that the user has already paid their transaction fee to the miner when they were incorporated into a block. In addition, there may be additional punitive measures taken if a user abuses this functionality and consistently fails to append their details.

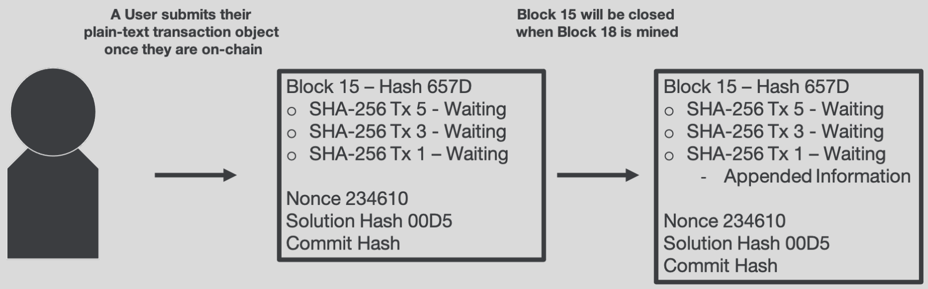


Figure 16: Once on-chain, the user has a limited window to submit plain-text details. These details are automatically checked and appended to the matching SHA256 and this block will be closed when block 18 gets mined.

This incentivizes users to ensure they submit their plain-text transaction details. We envision that for the bulk of users and smart contracts, this process will be entirely automated either through an exchange or their smart wallet.

Revealing plain-text transaction details at this point is safe as the transaction is now confirmed, on-chain, and is in no danger of being front-run. Even though the transaction plain-text details have been appended, the block, with all its transactions and smart contracts, still needs to be settled.

2.3 The New Clearing Function

Enter the clearing nodes. The job of these nodes is to clear, in the financial sense, the blocks. They are responsible for listening for plain-text transactions, verifying funds, executing any smart contracts, and clearing individual transactions.

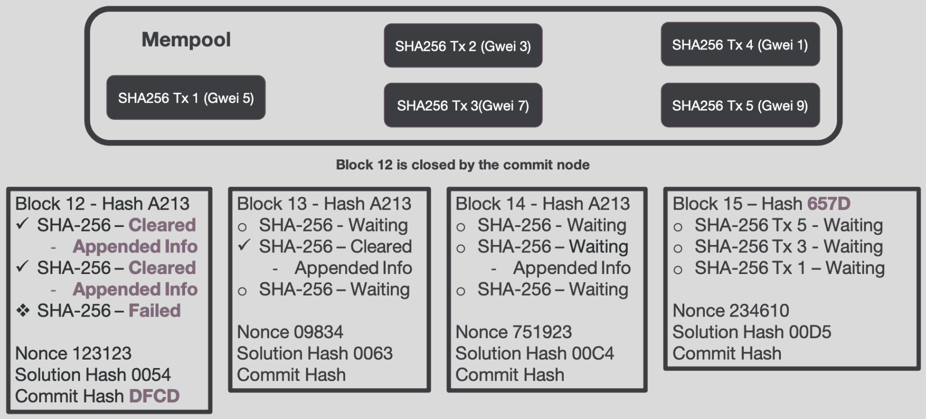


Figure 17: It is the clearing nodes responsibility to clear the transactions efficiently and in time. This means it's responsible for settling transactions and executing revealed smart contracts which can be of arbitrary complexity. Here, they are working on blocks 13, 14 and 15.

The clearing nodes know to begin their work when they receive and verify a new block that was broadcast by the commit nodes. The clearing nodes can immediately get to work clearing and settling all the transactions they can find for which users have submitted plain-text data. They can't officially commit these transactions or smart contracts to the blockchain yet as that is only done when the block closes and is the responsibility of the commit node network. The process of committing is detailed in the next section.

The clearing nodes are incentivized to make sure they complete the execution for appended info as they receive transaction fees based on the computational effort of the smart contracts and number of cleared transactions. They are also competing for speed and efficiency as the most appended block that is first transmitted to the commit node wins. That is why they are also the nodes responsible for listening on the network for appended information from users.

A clearing node doesn't have to be a single system as we will explain in the "Separation of Duties and Specialization" sub section. However, they must be able to quickly and efficiently clear the transactions in a block including executing any smart contracts, which requires them to maintain an up-to date full copy of the global database. If the node does not have a full copy of the complete blockchain, they can retrieve a copy of the current committed global database from the commit node network.

In order to have an unambiguous, decentralized timing system for users to submit their information, clearing nodes are permitted to work on a block until the commit nodes commit that block. At this point, any transaction that was in the process of being cleared and any transactions in that block waiting for matching plain-text data are failed and the clearing nodes move on to the next block that needs to be cleared.

2.4 Committing

Committing is the process by which a newly mined block gets added to the chain and the block at the end of the clearing window is executed and closed. In a decentralized system, there is no single clock or transaction timestamp that can be referenced or trusted for network consensus so we use the moment a new block is mined and needs to be added to the block chain as the timing event.

We said before that, when a mining node mines a new block it submits it to the commit node network, let's go into detail about what that process involves. The commit node network has 2 jobs. First, it maintains a copy of and consensus on the current blockchain including its global database. Second, it's responsible for handling the committing process which is the only way changes can be made to that blockchain.

Let's describe in detail their functioning. At all times, a commit node is listening to the entire network. From the clearing nodes, it is listening for updates to any blocks that are currently in the clearing window. Anytime a clearing node manages to append extra information to a block and clears an additional transaction it signs this new block and transmits it to the commit node. A commit node retains this cleared block until it receives another one.

At that point, the only way its retained cleared block is replaced is if a newly submitted block has more transactions appended. This ensures that the block retained by the commit node is from the first clearing node to clear a block as much as possible given the plain-data submitted by users. If a commit node replaces its block, it then broadcasts this new block to the network of its fellow commit nodes. It can also broadcast it to a clearing node, if the clearing node queries the commit node. This is to best ensure that the clearing nodes have in memory, the most appended block from the first clearing node to fully clear it.

There is one other way it can replace its clearing block. With the complexity of smart contracts, a clearing node can possibly lie about the output of the smart contract. While it is their responsibility to clear the node including settling all smart contracts, a clearing node can easily submit the correct appended info, but then claim that a smart contract resolves as payment to an address it controls instead. If any commit node receives a block from a clearing node where there is a conflict in how smart contracts are ultimately settled, it is now the responsibility of the commit node network to verify the smart contract. The commit node network now needs to clear that transaction and punish the malicious clearing node. In a proof of stake, this can be by garnishing some of the clearing nodes tokens which becomes a reward distributed across the commit node network. In order to prove which commit node acted maliciously, it can demonstrate that the malicious block was signed by the clearing nodes private key.

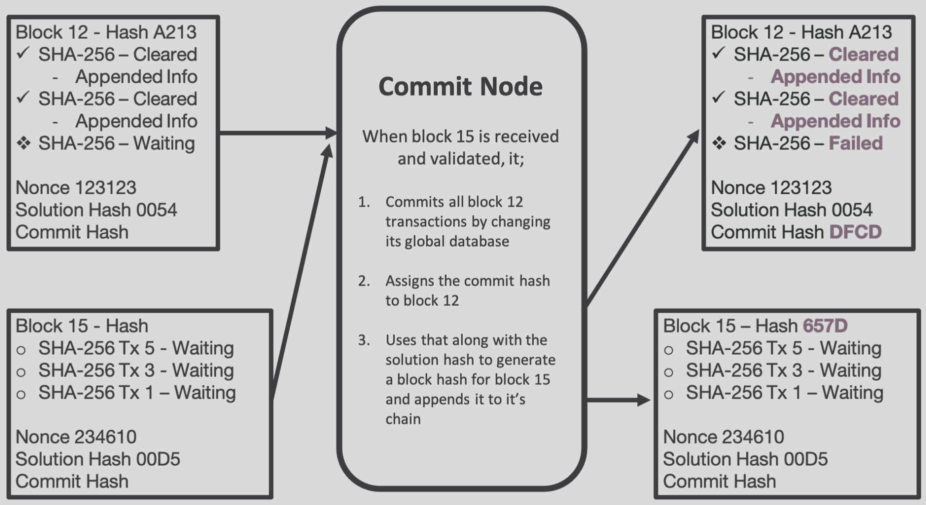


Figure 18: The commit process involves committing block 12's transactions, assigning a block hash to the new block 15 and appending block 15 to the blockchain.

Commit nodes are also listening to the mining nodes. When a mining node mines a new block, it is submitted to the commit nodes. The commit nodes at this point must add the new block the blockchain and close the block at the end of the clearing window. It does this in the following way, it first closes the block at the end of the clearing window by "failing" all transactions that are still waiting for plain-text data. These users have now lost their transaction fees to the network as the penalty for not submitting plain-text data in time. In addition, any users who do not have enough balance to pay for a transaction or any smart-contracts that were timed out by the clearing nodes are also failed. (We go into more detail about execution time-out in the next subsection).

The smart contracts are executed and the commit node's block and global database are updated to match the clearing node's. The block is now effectively closed and in order to make it immutable, the commit note writes the commit hash which is a SHA256 hash of the statuses and appended information. This commit hash is then appended with the solution hash of the new block, and the hash of this becomes the block hash for the new block which is now on the chain. This new completed chain is broadcast to the full network by the commit node. This alerts the mining network to begin mining the next node, and the clearing nodes to move the clearing window up one block.

2.5 Segregation of Duties and Specialization

Our proposal is built on the concepts of segregation of duty, and separation of concerns. Segregation of duty is an accounting philosophy that no one agent should have sole control over the lifespan of a transaction. In a traditional block chain, the "winning" mining node has the power to read all the transactions in full detail, select the transactions it wishes to block together, arrange them for execution in the order it wishes, and is responsible for clearing, contract execution and fund settling.

We believe that much power over the full life cycle of a transaction is too much power, especially as miners are now selling that power for hundreds of millions of dollars in profit a month. Separating the responsibility of clearing from committing from mining allows us to allocate ownership of different life cycles of the transaction to systems that have the specialized resources.

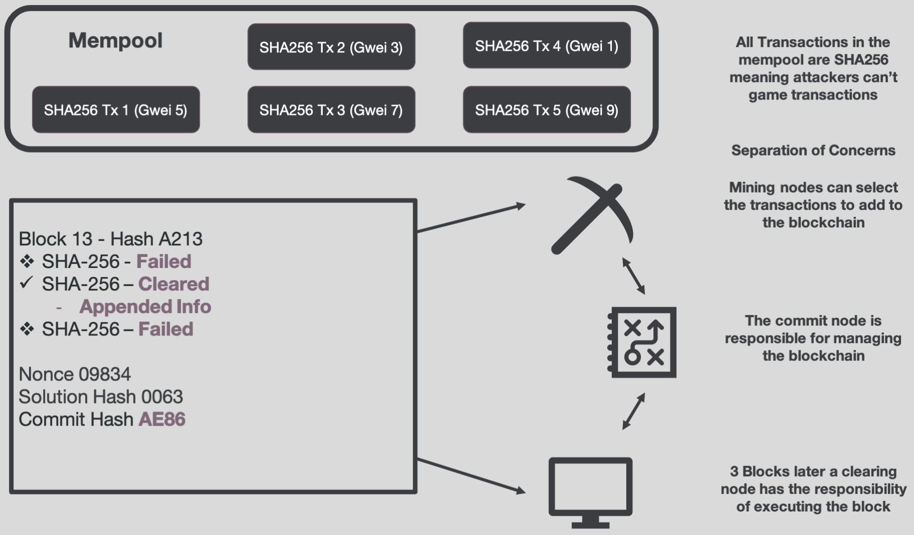

Mempool
SHA256 Tx 1 (Gwei 5)
SHA256 Tx 2 (Gwei 3)
SHA256 Tx 3 (Gwei 7)
SHA256 Tx 4 (Gwei 1)
SHA256 Tx 5 (Gwei 9)
All Transactions in the mempool are SHA256 meaning attackers can't game transactions
Separation of Concerns
Mining nodes can select the transactions to add to the blockchain
The commit node is responsible for managing the blockchain
3 Blocks later a clearing node has the responsibility of executing the block
Block 13 - Hash A213
❖ SHA-256 - Failed
✓ SHA-256 – Cleared
- Appended Info
❖ SHA-256 – Failed
Nonce 09834
Solution Hash 0063
Commit Hash AE86

Figure 19: In our proposal, we've fundamentally divorced clearing functions from mining functions from committing.

Extant implementations of blockchains have required a single node be capable of mining, clearing, and network operations. Furthermore, as mining has gotten more profitable, systems have been forced to focus on hash rates to secure the sole rights to oversee the full lifecycle of the transaction. This has made it inaccessible for average users, concentrating the power in the hands of large mining conglomerations with specialized hashing machines. In addition, because a network with smart contracts like Ethereum maintains a global database for the clearing and execution of those smart contracts, it also requires significant amounts of memory, high network bandwidth, and the ability to quickly and efficiently perform database updates, putting it even further outside of the reach of the average consumer.

In our proposal this is no longer the case. We aim to provide an opportunity for systems to specialize into a specific role best suited for their individual architecture. Ultimately, we envision systems networking together to divide duties to create highly efficient clearing nodes. As an example, a clearing node can be set up where tasks are delegated. Incoming blocks are split by a main system at the transaction level and then distributed to sister systems. One sister system can hold all of the databases referencing NFTs and specialize in their clearing, another can specialize into dealing with atomic-swaps, while another machine maintains a full copy of the global database for cross database smart contracts.

By eliminating the requirement that a single system needs to do it all, we overcome the traditional limitation to block size, block resolution speed, and storage requirements. These have all been limited in an effort to reduce the system requirements needed to oversee the full lifecycle of a transaction. Our proposal enables a high cadence of large blocks with complex smart contracts thanks to separation of duties and specialization.

3 Summary of Advantages, Limitations, and Competitor Solutions

We've designed our blockchain implementation to best address the challenges of extent blockchains. In our research, these have been the challenges of Miner Extractable Value (MEV), front-running attacks, and transaction throughput. In addition, we've also taken advantage of our specialized nodes to drive competition to micro-optimize our overall network. Let's breakdown the advantages and limitations as well as compare our proposal to the other potential solutions.

3.1 Proposal Advantages

By hashing transactions in the mempool, front-running has become massively more difficult. It now involves attempting to predict the behavior of transactions with imperfect information. Just like in traditional financial markets, it's still theoretically possible to build complex models that attempt to use historic information to model future transactions, however now it involves significant and genuine risk as the attacker no longer has perfect information.

Our solution to decoding these hashed transactions is to separate clearing and mining functions. Thanks to that specialization, our proposal can support significantly higher block sizes as the mining nodes don't need the infrastructure to clear or commit and the clearing nodes don't need the infrastructure to maintain an energy efficient hash rate. More importantly, by separating the clearing functionality, we are no longer constrained to an arbitrary gas limit of 1 second of CPU time. Clearing nodes are likely be pools of systems, but smaller pools should have reduced latency times which will allow them to compete for blocks that have less complex execution requirements. Meanwhile if a block has dozens of extremely complex smart contracts, a powerful clearing node comprised of multiple systems will outperform the low latency smaller pools.

This specialization also allows us to engage in something no other extant blockchain can, block by block micro-optimization of execution criteria. By forcing the clearing nodes themselves to compete with each other, we no longer need to centralize execution criteria such as timeouts. If, for example, a clearing node sets its timeout window too long, it won't be able to complete the block if there is an infinite loop in the transactions, thus ceding it to a competing clearing node. On the other hand, a timeout window set too short might timeout transactions that could otherwise be completed, this would also cede the block to a competing clearing node. These execution criteria can be left to the individual clearing nodes to optimize in order to generate the most fee revenue by best clearing a block. This makes our proposal significantly more user friendly than asking a user to estimate the gas limit for a random smart contract.

3.2 Proposal Limitations

Unfortunately, it is unavoidable that our blockchain in its current implementation will have a human-scale latency in transaction resolutions thanks to the clearing window. While it will certainly be shorter than the windows for many blockchain implementations such as Bitcoin which has a 10 minute block period, there is a strict lower bound as users will need human time scales to submit their transactions details. As a design choice for this implementation, we do not implement an automated appending system such as a timelock encryption or verifiable delay system. As of writing, effective, decentralized implementations of time-lock encryption are still in the proof of concept stage. We plan to revisit their implementation when they are better understood. Such a system can be appended as an update to the commit nodes in the future and would enable us to drastically cut the clearing window by automating the appending of plain-text transaction details.

3.3 Competitive Solutions

Due to the severity of these issues, dozens of teams are working on their own competitive solutions. The primary difference between our proposal and the current competitive set is that we are fundamentally rebuilding the blockchain. To best inform a reader on the space below we have summarized the major projects tackling these issues at the time of writing.

The first is to allow the free bartering of order placement. This can be done by the network itself where miners auction out on the side, the placement of orders to augment their transaction fee revenue. This is already occurring and is likely to accelerate even further if it is incorporated into the fundamental structure of mining. This is called Miner Extractable Value Auction, or MEVA. There is fierce debate as some claim this sort of side dealing will always exist while others claim that this is essentially a bribe from front-running bots and will end up hurting the average trader.

EDEN is a token that allows miners to monetize placement. Essentially, if a miner places and orders transactions as per the rules of the EDEN network, they receive EDEN tokens. Applications, bots, or traders can then essentially pay EDEN tokens to secure a higher placement in blocks or rent out block space. This is a private, tokenized version of MEVA.

In addition, there are several privacy tokens such as Monero, that make the entire chain completely opaque to all outside observers. While this does limit front-running transactions, it comes with significant regulatory and moral challenges that are beyond the scope of this paper.

There are a wide variety of solutions that take advantage of secure regions of the latest generation of CPUs, these are known under the brand names Intel's

SGX or AMD's SEV. Whether on blockchain or not, these solutions propose that all transaction blocking and routing be handled in the secure enclave of the CPU. The limitations on this are that it is not truly decentralized as it gives mining power to users with these specialized CPUs, and more specifically centralizes control at Intel or AMD. If these companies were to implement backdoors into the secure enclave or a workaround was found, this would completely compromise the security of the process instantly.

The reason all of these solutions have limitations and intelligent, distinguished teams all working on the same issue haven't solved it yet is that the challenge is fundamental. In the extant blockchains, the mining node or validator has too much power. It has full transparency and complete control over the full lifecycle of the transaction. We believe the only effective solution is to implement a blockchain that has ownership distributed among multiple node networks with checks and balances to ensure there are no conflicts of interest.

4 Contact

Feel free to reach out and contact us if you have any questions or would like to get involved.

Stardust Wealth

cryptodevelopment@stardustfunds.com