**Abstract:**

With the growing interest in cryptocurrency and its fundamental algorithm, studies of blockchain and cryptocurrency have been rising in the last few years, in this paper we study the security threshold for Ethereum, which is now the second cryptocurrency[[1]](#footnote-1).  
In this model we used the PoW protocol which miners as operators though it was recently changed to PoS protocol.

We analyze the selfish mining strategies in Ethereum, and we study the threshold size by implementing an MDP model following the previous works that have studied Ethereum and Bitcoin. Our addition in this paper that we consider the whale transaction that the miner receives an additional reward called Fees for the transaction included in the block.

We show that (TODO add results)

KEYWORS:

Ethereum, Proof of Work, Proof of Stake, Blockchain, Cryptocurrency, Markov Decision Process, Selfish mining, Miners, Tokens, Whale Transaction, Attacker Miner, Threshold Size, WeRLman, Nephew and Uncle rewards.

**Introduction :**

Ethereum is a decentralized blockchain platform that establishes a peer-to-peer network that securely executes and verifies application code, which is called smart contract.  
It was announced in 2014 and began operating in 2015, and now it’s the second largest cryptocurrency by market capitalization and today’s biggest decentralized platform that runs smart contracts which allows participant to transact with each other without a trusted central authority.

when we first started working on this project and for more than seven years, the chain was secured by POW protocol, which is based on decentralized blockchain protocols.  
PoW distribute reward to the miners according to their share of the total contribution. Thus, in this work we mainly study miners. Though we mention that recently Vitalik Buterin[[2]](#footnote-2), founder of the network had advocated for a transition, therefore on 15 September 2022, Ethereum adopted new algorithm, the proof-of-Stake consensus mechanism. This validation method requires stakers to verify new transactions.

Blockchains are maintained by miners , who receive rewards (virtual tokens) for creating blocks containing user-generated transactions, each block rewards its creator with newly minted tokens (virtual tokens) and with transaction fees paid by the users, in addition to the nephew and uncle rewards, special reward on Ethereum.

In ideal world, each miner receives rewards based on how much computational power he controls , however considering the selfish mining behavior we present an attack with which colluding miners obtain a revenue larger than their fair share, such attack can have significant consequences for Ethereum, attacker miners will prefer to join the selfish miners, and the colluding group will increase in size until it becomes a majority, at this point the Ethereum system ceases to be a decentralized currency.   
Therefore, we must ensure that all miners are smaller than a threshold size. To find the threshold size, we identify the optimal strategy for miners of different sizes.

To our knowledge, previous analysis on Ethereum doesn’t consider whale transactions (blocks with exceptional rewards), we model this behavior which implies a state space that is larger , therefore we use the WeRLman framework to analyze such models.

Following WeRLman paper that study Bitcoin considering whale transactions, in this paper we study Ethereum considering whale transactions, we study the security threshold of Ethereum’s mining strategies from the selfish mining perspective considering whale transactions, we try to tighten the bound on the threshold by introducing MDP model that searches against selfish mining pools and analyze its revenue to get an upper bound on the security threshold.

In this work we propose (TODO add how we solved the MDP)

WeRLman uses deep Reinforcement Learning, inspired by the AlphaGo Zero algorithm   
we use WeRLman to analyze the incentives of an attacker miner in various settings and upper bound the security threshold of Ethereum blockchain, that considering whale transactions. (not sure about WeRLman)

We show that the Ethereum protocol considering whale transaction, as prescribed and implemented is (TODO add results)

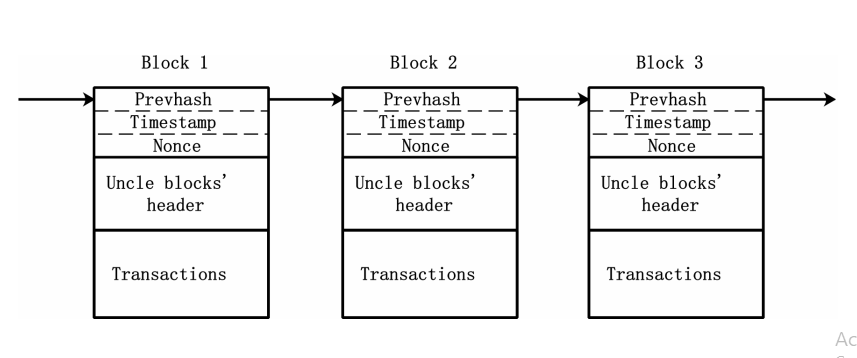
(TODO Compare to previous results )

**Blockchain: (TODO need to update to our words)**

The blockchain records the transactions in units of blocks, each block in the Ethereum blockchain contains three components, as we see in figure 1:  
A block header, includes a Keccak 256-bit hash value of the previous block, a time stamp and a nonce.

A set of transactions and some references links to certain previous blocks called uncle and nephew blocks.

Such a chain structure has several desirable features, it is tampered free, any changes of a block will lead to subsequent changes of all later blocks in the chain , it also prevents double spending, that lead to all clients having the same copy of the blockchain.



*Figure 1- an illustration of the blockchain structure in Ethereum.[[3]](#footnote-3)*

Communication of newly computed blocks is modeled to be faster than block creation, and we assume that no blocks are generating while others being transmitted, also we assume that new blocks are created according to a Poison process with rate λ (independent of the block creation process).

Any miner may add a valid block to the chain by simply publishing it over an overly network to all other miners, if two miners create two blocks with the same preceding block, the chain is forked into two branches, which will form a tree, and miners may add new valid blocks to either branch, in the same time they have to maintain a globally-agreed totally ordered set of transactions.

תמונה שמכילה תרשים

התיאור נוצר באופן אוטומטי

*Figure 2 – an illustration of a forked blockchain [[4]](#footnote-4)*

To resolve forks, the protocol prescribes miners to adopt and mine on the longest chain they know of, or the first one they hear of if there ate branches of equal length. This causes forked branches to be pruned and transaction in pruned blocks are ignored and may resubmitted by clients.

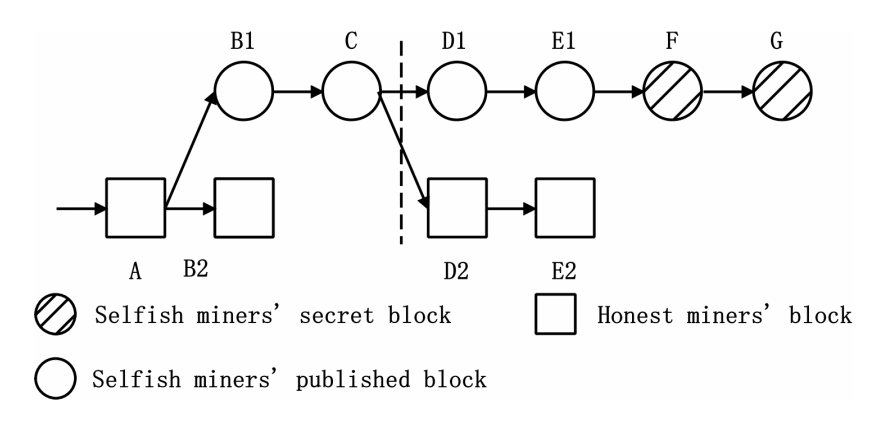
Blocks generate static rewards only if they are eventually part of the longest chain, if the block is uncle or nephew blocks at receives special rewards, also miners can receive transaction fee as a reward for the verifying and executing all the transactions.

**Selfish mining:**

Due to the nature of mining process, the interval between mining events exhibits high variance from the point of view of a single miner.   
Consequently, miners typically organize themselves into mining pools. All the members of a bool work together to mine each block, and later they share their revenue when one of them successfully mines a block, each miner who joins the pool we refer to him as an attacker miner, other miners are referred by honest miners.

Each honest miner observes a tree of blocks, it chooses a main chain from the tree and mines new blocks on its main chain, once a new block is produced the miner broadcast the block to everyone in the system, also it includes as many reference links as possible to unreferenced uncle blocks in the tree.

The attacker miner can withhold its newly mined blocks and publish them strategically to maximize its own revenue. The basic idea behind selfish mining is to increase the selfish pool’s share of static rewards, at the same time to gain as many uncle and nephew rewards as possible. The pool keeps its newly discovered blocks private, creating a fork on purpose. The pool then continues to mine on the private branch, while the honest miners still mine on public branches which are often shorter than the private branch.



*Figure 3 – an example to illustrate private branch length and public branch length[[5]](#footnote-5)*

Ethereum allows strategies that create blocks destined to become uncles, to deter miners from doing this, in the case the block references one or two uncles, the block and its uncles count only as two blocks.  
In our model we assume that any uncle counts as 1 block regardless of whether 2 uncles were referenced by a single block (in line with other previous work)

When the public approaches the pool’s private chain in length, and the pool choose to broadcast their private chain, this leads honest miners to waste resources on mining cryptographic puzzles that end up serving no purpose.  
Both honest miners and attackers waste some resources, but it is known that the honest miners lose more. Leads to the pool’s rewards exceed its share of the network’s mining power, this way makes other honest miner will join the pool.

Therefore, we must ensure that all miners are smaller than a threshold size.

Above a certain threshold size the revenue of the pool rises super linearly with pool size, above its revenue with the honest strategy.

Once a pool reaches the threshold, honest miners will prefer to join the pool to reap the higher revenues compared to other pools.  
At the end this pool might be the only creator, leading to that the decentralized nature of the currency to collapse and the single pool manager will control the system.

**Ethereum new Protocol:**

Since the Ethereum first operation in 2015, the chain was secured by Proof-of-Work (PoW) protocol. When we started this project Ethereum was still operating the chain in PoW, which uses miners as operators, thus we mainly studied miners’ work.

Recently, Vitalik Buterin, founder of the network had advocated for a transition to Proof-of-Stake (PoS) which at its early stage, it lives in parallel with the exiting PoW chain.  
Validators drive the entire PoS chain called Beacon chain of Ethereum 2.0, in order to be a validator one needs to deposit a certain amount of Ether as a “stake” by sending a transaction to the deposit contract, the validation method requires validators to verify new transactions. Instead of searching randomly for the nonce, validators place their Ethereum holdings into a smart contract as collateral. The deposit contract holds the history of deposits and locks all the deposits in the Ethereum chain.  
If validators fail to fulfil their validation responsibilities accidently or maliciously, they can be punished by losing their staked coins.   
Validators for any given block are chosen via a pseudo-random algorithm known as RANDAO.

**Mining Model:**

In this paper, we consider a system of n miners, theth miner has fraction of total hash power, then we have .  
Let S denote the set of selfish miners and H denote the set of honest miners, we consider that all the actions available to the attacker at any given point of time, and assume that the attacker controls a fraction α of the computational power, then the honest network has 1-α of the computational power ( the blocks are being generated by the attacker with probability of α and by the honest with probability of 1-α )

A Bernoulli process can be approximated by a Poisson process if the duration of a trail is very short, and the success probability is very low. Both conditions are held in the Ethereum model, therefore we can model the mining process of the th miner as a Poisson process with rate λ.

Hence, the attacker miner generates blocks at rate α\*λ , and honest miner generate blocks at rate (1-α)\*λ.

Let γ be the rushing factor of the system, if an honest miner publishes a blocks *b*, the attacker sees this block and can publish his own block *b*’, that’s been secret. The rushing factor γ denotes the fraction of mining power that will receive *b’* block before *b*. in our model since ties are broken uniformly at random rate, we assume , therefore in Ethereum the attacker miner generates blocks at rate α, while the honest miner generates blocks at rate (1-α).

We describe the ARR-MDP model (TODO).

Following the previous papers, we add assumptions that we used while solving our model[[6]](#footnote-6) :

1. There is a constant , for any policy 𝞹 it holds that :
2. There is a constant , for any policy 𝞹 it holds that : , the reward for the difficulty contribution symbolize numbers of blocks added at every step, and bounded by the length of the longest possible fork in the network, unlike Bitcoin model, in Ethereum the negative bounded are allowed.
3. for any policy 𝞹, the average difficulty contribution in ARR-MDP is lower bounded by some constant . Formally:
4. there is a state in ARR-MDP that is positive recurrent for any policy 𝞹. Denote this state as the initial state or .  
   This assumption holds for Bitcoin and Ethereum, if we assume that the miner doesn’t have more than 50% of the mining power, which is reasonable since otherwise the blockchain is already compromised.  
   Thus the honest miner always catch up with the attacker, since the miner cannot keep waiting forever, the miner must sync with the most of the network at some point and get back to the initial state.

**Mining Rewards in Ethereum:**

Ethereum introduces additional rewards to compensate a miner of a block that ended up out of the public chain, so the uncle and nephew rewards are unique for Ethereum.

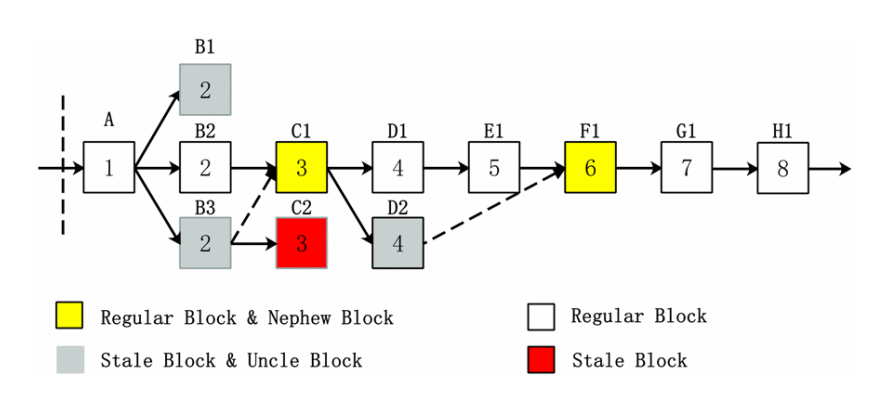
There are three types of block rewards in Ethereum, static blocks reward, uncle block reward and nephew reward, we have also additional reward for the miner called transaction fee.  
The static rewards are used in both Ethereum and Bitcoin, each block can bring its miner a static reward.

An uncle block is a block that is a direct child of a block in the public chain, an uncle block receives certain reward if it is referenced by some future block called a nephew block, through referenced links. The value of uncle rewards depends on the distance between the uncle and the nephew blocks.

In Ethereum if the distance is 1 the uncle reward is of the static block reward, if the distance is 2 then the uncle reward is , and so on.

Once the distance is greater than 6, the uncle reward will be 0.  
The nephew reward is always of the static block reward.

In addition, miners can also receive transaction fee as a reward for the verifying and executing all the transactions. Previous works in Ethereum ignored transaction fee as it is dwarf with other rewards, on our model we analysis Ethereum with transaction fees.



*Figure 4 – different block types in Ethereum [[7]](#footnote-7)*

**Markov decision processes** : (TODO need to change to our words)

The goal in a Markov decision process is to find a good "policy" for the decision maker: a function Π  � that specifies the action �(�)Π(S) that the decision maker will choose when in state �S. Once a Markov decision process is combined with a policy in this way, this fixes the action for each state and the resulting combination behaves like a [Markov chain](https://en.wikipedia.org/wiki/Markov_chain)

Our method is based on the theory of Markov Decision Processes, which we now review. An MDP describes a controlled stochastic process in discrete time, where at time *t* an agent observes an environment at state , takes an action , and subsequently the environment transitions stochastically to a new state , while the agent is awarded some reward . The transitions between states are Markovian, and denoted by :

The agent’s goal is to choose actions that maximize its long-term rewards, defined by the MDP objective function. Three objectives that have been extensively investigated in the literature are the discounted reward, the average reward, and the stochastic shortest path. For all these cases, an optimal decision-making policy can be presented as a Markov policy.

Furthermore, several algorithms for finding an optimal policy are known. An MDP with a specific policy 𝞹 induces a Markov chain over the states visited by the policy. In the sequel, the notation denotes an expectation over states in the Markov chain induced by the policy 𝞹.

**Method:** (not sure)

We describe our method for computing security bounds for blockchain protocols based on the MDP model. We begin with describing our MDP model for Ethereum , then we present the results of PTO and compare them to the approximately optimal results of SquirRL.

Finally, we describe our derivation of the security threshold for Ethereum and compare with previous works that have been done.

**Definitions:**

We define the MDP as 4 tuples:

  : is a [set](https://en.wikipedia.org/wiki/Set_(mathematics)) of states called the state space.

   :is a set of actions, called the *action space* (alternatively, �� is the set of actions available from state � )

:  is the probability that action �a in state   �at time �t will lead to state �′ at time �+1t+1.

is the immediate reward (or expected immediate reward) received after transitioning from state S �to state �′S’, due to action �a.

**Varying Fees Model:**

We present the varying model of a Nakamoto-based blockchain with varying fees. We model the process of mining from the perspective of the attacker miner as an MDP.

The attacker mines on a single public chain, and all other miners mine honestly on a single chain.

We normalize the static block reword to be 1, the baseline includes the subsidy and the fees commonly available in the mempool (the set of available transactions that were not included yet). Occasionally, there appear whale transactions, which offer an additional fee of F.

We assume each block can give a reward of either 1 or 1+F. in practice F would be the aggregation of multiple fees.

To represent the available whale transactions in the network we introduce an integer value pool into the state space all new whale transactions arrive at the pool by incrementing the value of the pool and can then be included in a block.

Honest miner always includes a transaction when available. However, the attacker miner can choose whether to include a transaction or not when there is one available.

**Comparing Ethereum to Bitcoin model with fees: ( TODO need to add more)**

The basic mechanism of the two protocols is similar, in both systems miners generate blocks that form a graph. As in Bitcoin the relative mining power is α, and the maximum fork length will play an important part.

As in Bitcoin, Ethereum’s blockchain is the longest chain of blocks. However, ties are broken uniformly at random rate. So, the miner’s rushing level does not play a role in Ethereum, so we assume a constant probability .

Ethereum presents the concept of uncle blocks as we described earlier, that’s to compensate a miner of a block that ended up out of the public chain. Ethereum also introduces additional rewards, like uncle and nephew rewards.

**ARR-MDP model for Ethereum:**

We now present the ARR-MDP model for Ethereum , we will present first the objective function of the MDP, then we present the state space and actions, later we describe how we confine the state space to be finite.

1. Objective function: In general, the MDP can be modeled such that at time step t the miner gains a reward of and the difficulty adjustment mechanism advances by , while denote the number of blocks that the miner added to the longest chain that includes the transaction fees as well as the subsidy rewards, and denotes the total number of blocks added to the longest chain.  
   The attacker’s incentive is to maximize average revenue per unit of time, .  
   The difficulty adjustment mechanism ensures that grows at a constant rate. Thus, dividing the accumulated reward by the sum yields the miner’s reward per unit of time.
2. S : state space

: list of blocks that have been built by the attacker since the latest fork, each item in the list denotes whether the block holds a whale transaction.

it’s different from the version with no fees that so we can keep track of all blocks with whale transactions.

: list that keeps the blocks that mined by the honest miners in the public chain.

it’s different from the version with no fees that we hold the blocks to know if there is a fee or not.

fork: the current case of the system , obtains 2 values :

means that the previous state was , this implies if .  
match is feasible   
 or means the fork label plays no effective role.

represents the case where the honest network is already split , due to a previous match action.

: it is 0 if the first block of the attacker miner is still a secret.  
else: r denote the length of the public chain since the last fork , which is .  
if the miner chose to reveal his first block, then the honest miner points to this block and the attacker miner gets the uncle reward with distance r , which specifies the uncle’s distance, This is because the miner’s first block since the last fork cannot be referenced before (it was a secret).

: denote whether there is a revealed block by the attacker miner since the last fork, which was not referenced as uncle before.  
it serves only to denote whether there is a potential uncle block pending from before the last fork.  
it doesn’t capture its uncle distance since we count its reward and difficulty contribution immediately after its fork is resolved.  
we know for sure that this block will be referenced, as the attacker miner references his own blocks and the rest of the network references all potential uncles

: a binary victor of length 6 denotes whether there are blocks by honest miners since the last fork that can be included as uncles, we need the block in order to denote if there is a fee or not  
every vector registers only the last 6 possible uncles, since further uncles are not allowed to be included   
for every block in the list, each entry denotes whether there is possible block to be included as an uncle with distance , after the last fork between the attacker miner’s chain and the public chain.

: the element pool is the number of available transactions since the last fork, we assume the block creation process is a Poisson process, our model is event-based, thus we assume that transactions can only arrive after each block creation event with some probability.  
the difficulty adjustment mechanism ensures that the rate of block creation in the main blockchain is constant.

1. A: action space

: always feasible, represent the attacker’s acceptance of the first of the honest network’s chain, the blocks in the attacker’s current chain are now can be used as uncles.

: represent the publication of the first of the attacker’s blocks,  
 feasible if

: represents the case where the most recent block was built by the honest network, and the attacker now publishes a conflicting block of the same height. Not always feasible depends on the state space  
the match actions give only two cases for the current state:  
Relevant – match can be performed if   
Active – match was already performed and the network is split, so match cannot be performed.

: always feasible, implies that the attacker does not publish new blocks, but keep working on its branch until a new block is built.  
after this action a new block is mined and added to the attacker chain or the public chain,  
 the binary parameter *f* specifies whether the attacker miner attempts to include a whale transaction to the block he is trying to mine, that block will hold a transaction is there is one available.

: If > 0 and the miner may reveal the first block of his private chain to be included as an uncle , unless it was already revealed   
revealing any additional blocks will not achieve anything as an uncle has to be direct child of a block in the main chain

1. Transitions and Rewards:

state x action =  *the list of the* attacker *miner shifted with blocks, those blocks are now uncles*  *blocks added to the public chain and the list of the honest miner shifted with blocks*all referenced uncles are removed from and the remaining possible uncles in are shifted back by .  
we have two cases:if r > 0 and the miner’s first block was referenced (we know this by scanning the list and check for every node if there is block that have link to this node), then: and r   
if r > 0 and the miner’s first block was not referenced then : to remember that this uncle still has to be referenced with the current distance r , and then set   
then we add the reward for the uncle block to the attacker with distance r , and we add the whale transaction to F

also, the honest miner accepts additional reward, we scan the list that we accepted and calculate the number of the transactions on the list and multiply with F  
F : discounted with the sum of whale transactions in the  *blocks*The difficulty contribution is the number of blocks plus the number of uncle blocks referenced.

pool: discounted with the sum of whale transactions in the  *blocks.*  
If r > 0 , we reward the attacker miner with a relevant uncle reward, assuming it will be referenced in the first block of the next fork.  
If was 1 and this means when we added the reward before referencing, we calculated depending on the current r , while in face we should calculate based on r+1 , so we correct this by fining the attacker miner by

state x action =check if there is a transaction as we create the block, if there is, we always add them, if there is not so we create an empty block.   
we emphasize that the attacker chain and the honest chain are not related, which means if we decided to add a transaction to the attacker’s chain, the honest miner can also create a block that contains the same transaction.

with probability of α , we add the new transaction to the attacker miner  
with probability of (1-α) , we add the new transaction to the honest miner.  
we must check if the new block that was created have been added to the longest chain, then we update the pool size.  
fork remains .  
we increase the difficulty contribution with 1.

state x action =

If the next mined block is by the attacker or by the honest miner who extends the public chain, no blocks become accepted by everyone. So, the fork is not resolved in this case and the uncle information stays the same.

If the next mined block is by an honest miner who extends the miner’s chain, then the fork is resolved and the public chain advanced by blocks. In this case:  
all the possible uncles in are shifted back by .  
if was 1 we give the attacker miner a nephew reward and then set   
we add 1 to the difficulty contribution.  
we delete the transactions that have been solved in the that have been accepted  
the attacker accepts additional reward , we scan the attacker’s chain that we accepted and calculate the number of the whale transactions and multiply with F, which demonstrate the fees.

state x action =

feasible only if

and remains unchanged.  
 remains relevant  
 is now equal to .

In this action no rewards added, therefore the difficulty contribution remains the same.

state x action =feasible only if

and remains unchanged.  
 is now Active  
 : if then remains the same , else if is smaller than 6 , which meant, then .  
In this action, we don’t add reward.  
the difficulty contribution is now increasing with the new value of ( if )

state x action =feasible only if   
 : *the list of the* attacker *miner shifted with* blocks, those blocks are part of the public chain.  
 : the list of the honest miner is now empty.  
we scan the list of the attacker in order to know the number of accepted transactions.  
 is now relevant.  
.   
   
if was 1 before, we add nephew reward to the rational miner and then   
 discounted with the sum of whale transactions in the  *blocks.*the difficulty contribution increased with .  
then we add the reward for the blocks to the attacker.  
the attacker accepts additional reward from the whale transaction, we scan the first nodes of that we accepted and calculate the number of the whale transactions and multiply with F, which demonstrate the fees.

**Simplified varying model:**we introduce a simplified model based on the varying model which allows the attacker a subset of the possible actions. This allows the constricted model to be represented more efficiently and greatly decreases the size of the state space. We can then use PTO to solve this model exactly.

The state space of the simplified model is represented as 11-tuple of the form:

: list of flags representing whether each block in the attacker miner’s chain holds a transaction.  
: list of flags representing whether each block in the public miner’s chain holds a transaction.  
: represent the length of the attacker miner’s chain.

: represent the length of the honest miner’s chain.

: denotes the number of transactions in the attacker’s chain.

: denotes the number of transactions in the honest miner’s chain.

: the current case of the model: relevant \ active

: 0 if the first block of the attacker is still secret, else .

: binary vector of length 6, denotes whether there are blocks by the honest miner that can be included as uncles.

: denotes whether there is a revealed block by the attacker miner.

: the number of available transactions since the last fork

**Model assumptions:**

we bound the state space of our model , because it is easier to work with finite state space   
we bound the maximum fork length by enforcing that: all actions that would lead to either or to be longer than the maximum value cannot be used , instead the miner would have to reveal or adopt some blocks to reduce the length of the chains

we bound the maximum possible pool size, whenever a whale transaction arrives and causes the number of available transactions to exceed the limit , the new transaction is discarded, and pool is left unchanged.

the max pool size is the same as the max fork size.

**References:**

1. Ethereum: State of Knowledge and Research Perspectives
2. Ittay Eyal and Emin Gun Sirer, Majority is not Enough: Bitcoin Mining is Vulnerable (2013)
3. Chen Feng and Jianyu Niu, Selfish Mining in Ethereum (2019)
4. Roi Bar-Zur, Itay Eyal, Aviv Tamar, Efficient MDP Analysis for Selfish-Mining in Blockchains (2020)
5. Ayelet Sapirshtein, Yonatan Sompolinsky and Aviv Zohar, Optimal Selfish Mining Strategies in Bitcoin (2015)
6. Roi Bar-Zur, Ameer Abu-Hanna, Itay Eyal, Aviv Tamar, WeRLman: To Tackle Whale (Transactions), Go Deep (RL)
7. Elie Kapengut and Bruce Mizrach , An Event Study of the Ethereum Transition to Proof-of-Stake.
8. Daejun Park, Yi Zhang and Grigore Rosu , End-to-End Formal Verification of Ethereum 2.0 Deposit Smart Contract

1. According to Cryptocurrency Prices by Market Cap [↑](#footnote-ref-1)
2. According to Elie Kapengut and Bruce Mizrach study [↑](#footnote-ref-2)
3. Taken from Selfish Mining in Ethereum paper, by Jianyu Niu and Chen Feng [↑](#footnote-ref-3)
4. Taken from Selfish Mining in Ethereum paper, by Jianyu Niu and Chen Feng [↑](#footnote-ref-4)
5. Taken from Selfish Mining in Ethereum paper, by Jianyu Niu and Chen Feng [↑](#footnote-ref-5)
6. Model assumptions 4.2 , in Efficient MDP analysis for selfish mining in blockchains [↑](#footnote-ref-6)
7. Taken from Selfish Mining in Ethereum paper, by Jianyu Niu and Chen Feng [↑](#footnote-ref-7)