DECOR+LAMI: A Scalable Blockchain Protocol

Philippe Camacho*

Sergio Demian Lerner†

September 9, 2016

Abstract

Since Satoshi Nakamoto's breakthrough, Bitcoin has evolved from a single piece of open source software to a vibrant ecosystem composed by developers, entrepreneurs, investors and academics. Despite this outstanding success, several challenges have to be overcome. In particular the scalability of blockchain protocols is still an open problem, and recent attacks such as selfish mining are an evidence that new tools are needed in order to guarantee the security of the consensus between miners. In this work we introduce DECOR+LAMI. DECOR (DEterministic COnflict Resolution) is a framework which purpose is to provide the right incentives in order to align miners' behavior to the expected properties of the protocol. On the other side, LAMI is a set of implementation optimizations for improving block propagation between miners that complements the DECOR protocol in order to improve scalability and security for blockchains.

1 Introduction

Since Satoshi Nakamoto's breakthrough [13], Bitcoin, has evolved from a single piece of open source software to a vibrant ecosystem composed by developers, entrepreneurs, investors and academics. While the original goal of Bitcoin was to provide a mechanism for handling money without the need of a trusted third party, the underlying technology, also called the Blockchain, has opened the path for additional exciting applications including anonymous online payments [2], decentralized naming systems and public key infrastructure [1], and recently smart contracts [10, 3]. Despite this outstanding success, several challenges have to be overcome. Scalability has become a priority in the development roadmap of Bitcoin [15, 17] and has yielded new proposals [8, 5, 12] regarding the consensus mechanism originally proposed by Nakamoto. On the other hand, the security of the blockchain technology is still a matter of intense study [7, 8, 14]. In particular new kind of attacks like selfish mining [6] have shown that new tools and ideas are needed in order to guarantee that economic incentives are aligned with the decentralized nature of blockchain systems.

In this work we introduce DECOR+LAMI. DECOR (DEterministic COnflict Resolution) is a framework which purpose is provide the right incentives in order to align the miners' behavior to the expected properties of the protocol such as fairness or bounded monetary supply for example. The idea of DECOR is twofold: First it provides a way for miners to choose the same block in the case of a conflict (several blocks at the same height). Second, a configurable

*philippe.camacho@gmail.com

†sergio@rootstock.io

incentive policy is computed by all the miners in order to assign rewards to the blocks. For example if we want to incentivize the collaboration of miners we will give an extra reward to blocks that reference uncles. While the full analysis of such incentive policies will be the object of future research, we provide evidences that DECOR can be used to improve resistance against selfish mining or guarantee a fixed money supply for inclusive blockchain protocols[12]. LAMI is a set of implementation optimizations for improving block propagation between miners. LAMI is fundamental complement of DECOR as it allows to consider that all miners have almost the same view of the published blocks that are to be included in the blockchain at some point of time.

Contributions. In this work we make the following contributions:

- We introduce a new conceptual framework, DECOR, for designing blockchain protocols. This framework can be instantiated in different ways depending on the properties that need to be achieved.
- We propose DECOR to be used for achieving consensus between miners when the block interval is small, thus providing alternative to existing solutions like GHOST[9].
- We show that inclusive blockchain protocols are not resistant to selfish mining by default, and that DECOR can be used to make this kind of attack less effective.
- We propose LAMI, a set of implementation optimizations for block propagation that combined with DECOR enables to reduce latency and increase throughput.

2 Preliminaries

2.1 Notations and Definitions

If X is a bit string of length n then $X[i]$ is the bit starting in position $i+1$. The XOR operator between bit arrays is written $\oplus$. Let X and Y be two bit strings then $X||Y$ is the concatenation of X and Y. In this paper $\mathtt{H}$ will denote a collision-resistant hash function. We use k to denote the length of $\mathtt{H}$'s output, that is $\forall x \in \{0,1\}^* : |\mathtt{H}(X)| = k$. For example if $\mathtt{H}$ is SHA-1 then $k = 160$.

The block interval is the average time between needed by the miners in order to extend the chain by one block. In the case of Bitcoin the block interval is about 10 minutes. In the rest of the paper $0 \leq \alpha \leq 1$ and $\beta = 1 - \alpha$ will represent the fraction of the computational power of the adversary and honest participant respectively.

2.2 Selfish mining

The problem of selfish mining [6] introduced by Eyal and Sirer is a surprising attack where the adversary leverages his ability to delay the publication of mined blocks in order to decrease the relative revenue of honest participants.

In Figure 2.2 we can see the state diagram of a selfish mining strategy where the adversary mines in secret until (1) he manages to have a chain that is longer than the public one by at least two blocks or (2) both public and private chains are of length one. We recall here the description of the states:

- **0**: there is only one public chain, the selfish mining attack did not start yet.
- **1**: the adversary manages to compute one block. The block is kept private.
- **2**: the adversary's private chain is longer than the public one by two blocks.
- **3,...,n**: the adversary's private chain is longer than the public one by $3, ..., n$ blocks respectively.

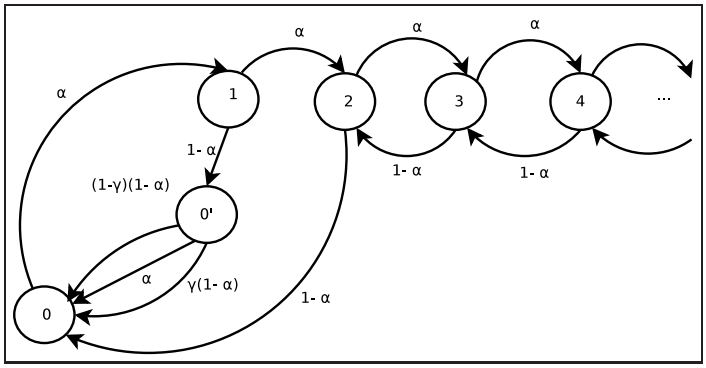

Figure 1: Markov decision process for selfish mining attack [6].

Let $X_1, X_2, ..., X_N$ be list of conflicting blocks
Let Y be the mined block
Let $X_i := \mathtt{S}(X_1, ..., X_N)$
Mine on top of X_i
$[r_1, r_2, ..., r_i, ..., r_N, y] := \mathtt{R}(X_1, X_2, ..., X_i, ..., X_N, Y)$

Listing 1: DECOR+ algorithm.

- **0'**: this is the crucial state where the adversary and the honest participant's chains are both of length 1. The adversary then decides to publish his block so that this block can be included in the main chain. An important parameter for this outcome is $\gamma \in [0, 1]$, the proportion of honest miners that will mine on top of the adversary's block.

The intuition why the selfish mining attack works despite the mining power of the adversary $\alpha < 50\%$ is because the adversary can keep his risk to a minimum by releasing his block at "the right time" (see state $0'$ described above). In Section A we introduce and analyze a similar attack for inclusive blockchain protocols.

3 DECOR+ (DEterministic COnflict Resolution)

Currently in Bitcoin when two blocks have been solved at equal height there is no clear way for miners to decide which block should be selected to extend the chain. It has been shown that this decision can affect the probability of success of selfish mining attacks [6] or compromise the consensus between participants when the block interval becomes small[7].

In this paper we propose a high level strategy, DECOR that enables the miners agreeing on the next block to mine in case of a conflict. While the implications of such strategy are complex to analyze and require further research, we are convinced that it will lead to better performance and security for blockchain mining protocols. In the following section we will introduce the abstract version of the DECOR strategy and then propose some instantiations in order to illustrate the potential of the concept.

3.1 Strategy

DECOR is a strategy which purpose is twofold:

- Choose *deterministically* a block to mine on top in case of conflict using the selection function $\mathtt{S}$.
- Provide the right incentives to the miners by assigning a specific reward to each block.

The general DECOR strategy is described in Listing 1. Let $X_1, X_2, ..., X_N$ The reward function $\mathtt{R}$ assigns specific rewards $r_1, r_2, ..., r_i, ..., r_N, r_y$ to blocks $X_1, X_2, ..., X_N, Y$ respectively.

For example in the case of Bitcoin we have that:

$$\mathtt{R}(X_1, ..., X_N, Y) := [0, ..., 0, \mathtt{r} + Fee(Y)]$$

Where $\mathtt{r}$ is the base reward for a block (e.g.: $12.5BTC$ at the time) and $Fee(X_j)$ is the transactions fee of block Y.

Note that the reward function $\mathtt{R}$ can be arbitrary and in particular it is possible to assign the transactions fee of block X_i to block X_j or simply increment the reward of the node X_i that has been selected by the function $\mathtt{S}$.

The main idea behind DECOR is to strengthen the consensus mechanism by enabling the honest miners to take a unified decision in case a conflict occurs. Thus it is important to highlight that DECOR is not a replacement for the longest-chain rule or could possibly be combined with other consensus mechanism like GHOST[8]. Also DECOR implicitly considers inclusive blockchain strategies yet it it can be instantiated without considering the inclusion of uncles in the chain (see example above).

In the following section we describe the expected properties of the functions $\mathtt{S}$ and $\mathtt{R}$ respectively.

3.1.1 Selection Function S

The selection function $\mathtt{S}$ is the core element of the DECOR strategy. Intuitively the function $\mathtt{S}$ should enable fast and accurate consensus between miners when dealing with conflicting blocks and also avoid this consensus mechanism to be manipulated by and adversary. So we propose that the selection function must be:

- **Deterministic:** All the miners must agree on the same block if we consider that they all share the same list of conflicting blocks.
- **Non-forward settable:** intuitively this means that no miner should be able to compute his block in such a way that the probability to have this block selected is higher than $1/N$ where N is the number of conflicting blocks. Here we assume that the miner does not know the set of conflicting blocks before it starts to create/mine his own.

The first property is straightforward to define and implement. We explore here the second property more in details.

Definition 1. ***Non-forward settable selection function*** $\mathtt{S}$***:*** *Let $\mathcal{A}$ be an adversary and $\mathtt{S}$ a selection function. We say that $\mathtt{S}$ is non-forward settable if and only for any $X_1, ..., X_{N-1}$ blocks* not known *by the adversary we have:*

$$Pr[Z \leftarrow \mathcal{A}(); Z = \mathtt{S}(X_1, ..., X_{N-1}, Z)] < \frac{1}{N}$$

Impeding an adversary to compute block that has higher probability to be selected than others is fundamental in order to avoid having the adversary control the consensus mechanism for its own advantage.

Let us illustrate what can happen in the case the selection function does not have this property. Let $\mathtt{S}$ be defined as follows: For any $X_1, ..., X_N$, $S(X_1, ..., X_N) = max_{j \in [1..N]}\{Fee(X_j)||\mathtt{H}(X_j)\}$ where $Fee(X)$ represents the transactions fee of the block. So

here the idea is to pick the block that has the highest fee and in case other blocks share the same fee use the hash to select only one block.

In this case we can observe that an adversary can increase its chance of having its block selected simply by creating a block that contains transactions (possibly generated by this same adversary) with very high fees.

Having an adversary being able to have its block selected can increase its chances to perform a selfish mining attack for example. In section 3.2.4 we show that by picking a *non-forward settable* function S we can limit the success of an adversary for such attacks.

We now instantiate a selection function and show it is *non-forward settable*:

Proposition 1. *Let* S *be a selection function defined as follows: If for any* $X_1, ..., X_N$*, we have that* $S(X_1, ..., X_N) = X_j$ *where* $j = \sum_{i=1}^{k} (\oplus_{i=1}^{N} \mathtt{H}(X_i))[i] \mod N$ *then* S *is non-forward settable (and also deterministic).*

Proof. (Sketch) The fact that S is deterministic is straightforward. The idea of S is to first compute the hash of each block and then combine all these hash values with an XOR function. If we assume that H is a random oracle[4], then the bit array $B = \oplus_{i=1}^{N} \mathtt{H}(X_i)$ belongs to a uniformly random distribution. Then the index j is also taken from a random distribution and thus if Z is a block computed by any adversary, we have that $Pr[\mathtt{S}(X_1, ..., X_{N-1}, Z) = Z] = \frac{1}{N}$. □

It is important to note the fact that while S described above is deterministic, by relying on the random oracle we can consider that the S function operates similarly to a function that would pick a block at random.

3.1.2 Reward function R

The reward function R enables to design incentive mechanisms for miners in order to obtain specific behaviors. The definition of the reward function R considers the conflicting blocks as input and also can update the rewards for these blocks. Note that, while it is not mandatory, the function R could rely on the function S in order to assign a specific reward to the selected block. Moreover, the reward function can take into account more factors than the mining difficulty and transaction fees. For example in order to incentivize the inclusion of uncles while keeping the monetary supply bounded, one can think have rewarding the miners who include all the uncles using a publisher fee.

3.2 Applications

In this Section we illustrate how DECOR can be used to improve the efficiency and security of blockchain protocols.

3.2.1 An alternative to GHOST

In order to increase the throughput of a blockchain protocol one can decrease the block interval so that the number of blocks and thus transactions that are processed in a fixed time window can increase. However setting arbitrary small confirmation time can have a negative impact on security[7]. Thus new proposals like GHOST [16] provide alternative ways of reaching consensus to the original "*extend the longest chain rule*" in Bitcoin. While DECOR and GHOST may be used together, they share a common purpose: Indeed the selection function S enables honest miners to agree on the block that needs to be picked in order to extend the chain and thus decreases the probability that large forks appear in the chain whether by accident or malicious behavior.

$\mathtt{R}(X_1, ..., X_N, Y)$:

Let i be the inclusion reward

res $= [\frac{\mathtt{r}}{N+1}, ..., \frac{\mathtt{r}}{N+1}, \frac{\mathtt{r}}{N+1} + \mathtt{i}N]$

return res

Listing 2: Incentivizing block inclusion

$\mathtt{R}(X_1, ..., X_N, X_{N+1}, X_{N+2}, ..., X_{N+l}, Y)$:

Let $X_{i_1}, X_{i_2}, ..., X_{i_{N+l}}$ be such that

$\mathtt{H}(X_{i_1}) < \mathtt{H}(X_{i_2}) < \cdots < \mathtt{H}(X_{i_{N+l}})$

$[r_{i_1}, ..., r_{i_N}, r_{i_{N+1}}, ..., r_{i_{N+l}}, R_Y] := [\frac{\mathtt{r}}{N+1}, ..., \frac{\mathtt{r}}{N+1}, 0, ..., 0, \frac{\mathtt{r}}{N+1}]$

return $[r_{i_1}, ..., r_{i_N}, r_{i_{N+1}}, ..., r_{i_{N+l}}, R_Y]$

Listing 3: Bounding the number of uncles

3.2.2 Incentivizing the inclusion of blocks

By default there is no reason why miners should reference uncles when mining their block. At the same time there are several advantages for incentivizing miners to include uncles:

- As uncles also contain transactions, the global throughput (number of transactions processed by second) of the blockchain will increase.
- Not including uncles may be a way to facilitate double spending attacks or selfish mining attacks.
- Uncles inclusion favors a more collaborative approach to the issuance of new coins, facilitating the decentralization of the mining process.

Using the reward function R one can incentivize the inclusion of uncles by assigning an extra reward for each uncle block that is included by the miner (see Listing 2).

3.2.3 Avoiding unbounded increase of uncles

While the inclusion of uncles has many benefits, it may create conflicts with other expected properties of the blockchain. In particular having a fixed monetary supply is not compatible with rewarding an unbounded number of uncles as noted by Sergio D. Lerner[11]. In the case of DECOR the reward function can be used to incentivize miners to keep the number of uncles below a specific threshold N. An option is for example to order the uncles by their hash and only reward the first N uncles. The effect of this measure is that miners will not want to take the risk to mine uncles if they already observe that N or close to N blocks have already been mined for a block interval.

3.2.4 Improving resistance to selfish mining

As mentioned in [12] inclusive blockchain protocols are not resistant to selfish mining by default. Even assuming all the miners have the right incentive to include all blocks (see Section 3.2.2), selfish mining attacks still are possible. The full analysis of such an attack is available in Appendix A. Intuitively the problem is that while the honest miner will suffer less damage in case of a selfish mining attack, the attacker will be able to be successful while taking less risk. In Figure 2 we can see that there exist a selfish mining strategy that becomes profitable when $\alpha > 21\%$ for $\gamma = 50\%$.

$\mathtt{R}(X_1, ..., X_N, Y)$:

$X = \mathtt{S}(X_1, ...X_N)$

Let Z be the block selected by the miner

Let Y be the mined block

Let $\mathtt{p} \in [0, 1]$ be the punishment fee

if $X! = Z$:

$res = [\frac{\mathtt{r}}{N+1}, ..., \frac{\mathtt{r}}{N+1}, \frac{\mathtt{r}(1-\mathtt{p})}{N+1}]$

else :

$res = [\frac{\mathtt{r}}{N+1}, ..., \frac{\mathtt{r}}{N+1}, \frac{\mathtt{r}}{N+1}]$

return res

Listing 4: DECOR algorithm with punishment fee.

In the following we show that this attack can be done less effective by using a specific reward function (see Listing 4).

USING THE PUNISHMENT FEE TO INCREASE THE DIFFICULTY OF PERFORMING A SELFISH MINING ATTACK. We describe here a way to increase the threshold α to make a selfish mining attack successful. The idea is to introduce a punishment fee that will be applied on the blocks mined by any miner who does not follow the selection function. The instantiation of the DECOR strategy with punishment fee relates the selection function $\mathtt{S}$ and the reward function $\mathtt{R}$ so that miners who do not follow $\mathtt{S}$ will see the reward of their block diminished by some factor $1 - \mathtt{p}$ where $\mathtt{p} \in [0, 1]$ is an arbitrary punishment fee.

In the case of the attack described in Section A.1 we can observe that the selfish miner cannot follow the selection function $\mathtt{S}$ otherwise he will not be able to inflict any damage to honest miners. So with probability 1 the punishment fee will be applied to the first block of the private chain of the adversary. We can also observe that it is fundamental to use a non-forward settable selection function as otherwise the selfish miner can artificially increase the value of γ.

The consequence for the selfish miner is that the first block of his private chain will be affected by the punishment fee which will make his attack less profitable. Concretely (see Table 2) the rewards obtained in transitions $0' \to 0$, $0' \to 1$ and $2 \to 0$ will be $1 + 0.5\mathtt{p}$, $0.5\mathtt{p}$ and $1 + 0.5\mathtt{p}$ respectively (instead of 1.5, 0.5 and 1.5 respectively).

In Figure 3 we can see the effect of a punishment fee of $\mathtt{p} = 50\%$: profitable selfish mining attacks require now $\alpha > 36\%$ for $\gamma = 0.5$ instead of $\alpha > 20\%$ when not using punishment fees.

OTHER TYPES OF ATTACKS. The attack described in Section A.1 is such that the selfish miner waits for a fork in the chain and starts mining on top of a block B that will be not selected to extend the chain considered by honest miners. The advantage for the attacker is that block B can be mined by an honest miner so that the attacker does not need to invest computational power at an early stage.

Another reasonable strategy for the attacker would be to compute B himself trying to leverage the punishment fee as follows: By publishing the new block B later the selfish miner is able to have his block chosen by the selection function $\mathtt{S}$ so that blocks mined by honest miners have their reward decreased due to the punishment fee. We give here the intuition why this strategy may be also costly for the selfish miner: Indeed if the selection function is non-forward settable then the selfish miner faces the following dilemma:

1. Wait for the honest miners to publish their blocks and then compute the block so that it will be chosen by the selection function.

2. Try to compute the block while not knowing the other blocks and taking the risk to have his block not chosen by the selection function.

The first alternative forces the selfish miner to delay his start in the computational race against the honest miners. Note that the challenge faced by the selfish miner is to be able to mine several blocks in a row while having less computational power than the honest majority. Clearly in this case delaying the computation of his first block increases the risk to loose the race. The second alternative also presents some issues for the selfish miner. Given that the selection function is non-forward settable, the selfish miner must take a high risk of having his block not chosen by the selection function and thus having the reward of the second block of his private chain decreased, making the attack less practical. Finally we want to highlight that there might exist other selfish-mining attacks that are more tolerant to the use of punishment fee. Analyzing the impact of punishment fees on general / optimal mining attacks will be the object of future research.

3.2.5 Achieving all security objectives through R and S

The purpose of DECOR is to provide tools to achieve several security objectives providing the right incentives to miners. Defining security objectives and providing a solution that satisfy all of them is a difficult task. In the previous sections we provided examples to illustrate how specific reward functions can help to achieve a given security objective. In Table 1, we summarize the list of security objectives mentioned in the previous sections and how a specific instantiation of DECOR(see Listing 5) can make these security objectives compatible.

$\mathtt{S}(X_1, ..., X_N)$:
    Let $j = \sum_{i=1}^{k} (\oplus_{i=1}^{N} \mathtt{H}(X_i))[i] \mod N$
    return X_j

$\mathtt{R}(X_1, ..., X_N, X_{N+1}, X_{N+2}, ..., X_{N+l}, Y)$:
    Let $X_{i_1}, X_{i_2}, ..., X_{i_{N+l}}$ be such that
        $\mathtt{H}(X_{i_1}) < \mathtt{H}(X_{i_2}) < \cdots < \mathtt{H}(X_{i_{N+l}})$
    $LetX := S(X_1, ...X_N)$
    Let Z be the block selected by the miner
    Let Y be the mined block
    Let $\mathtt{p} \in [0, 1]$ be the punishment fee
    if $X! = Z$:
        $r_Y = \frac{\mathtt{r}(1-\mathtt{p})}{N+1}$
    else :
        $r_Y = \frac{\mathtt{r}}{N+1}$
    Let i be the inclusion reward
    $[r_{i_1}, ..., r_{i_N}, r_{i_{N+1}}, ..., r_{i_{N+l}}, R_Y] := [\frac{\mathtt{r}}{N+1}, ..., \frac{\mathtt{r}}{N+1}, 0, ..., 0, r_Y + \mathtt{i}N]$
    return $[r_{i_1}, ..., r_{i_N}, r_{i_{N+1}}, ..., r_{i_{N+l}}, R_Y]$

Listing 5: Compatibilizing all security objectives

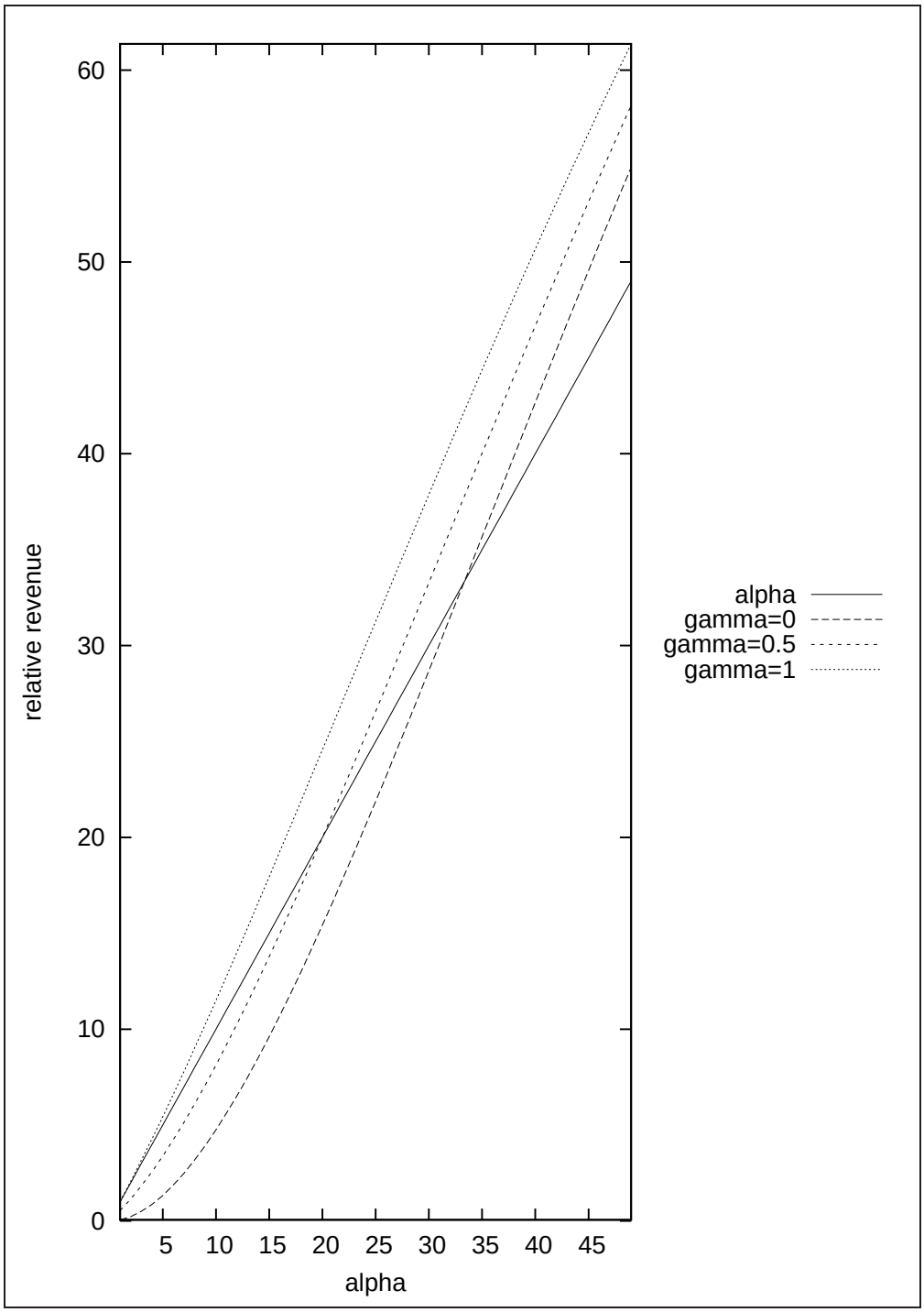

Figure 2: **Relative revenue:** the plain curve represents α. The other curves represent the relative revenues $R = \frac{r_a}{r_a + r_h}$ for different values of γ.

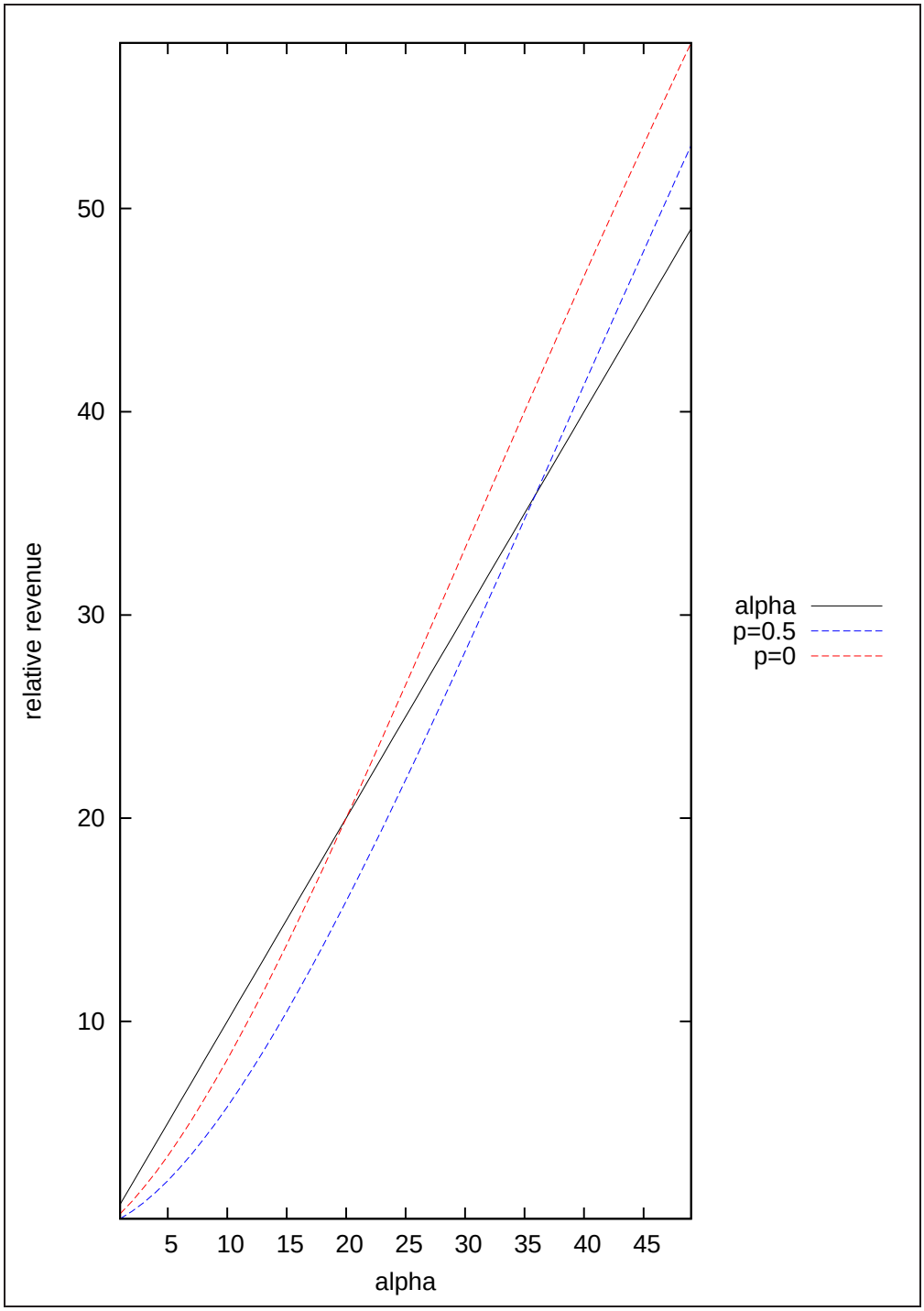

Figure 3: **Relative revenue with punishment fee:** the plain curve represents α. The other curves represent the relative revenues $R = \frac{r_a}{r_a + r_h}$ for different values of the punishment fee p and and $\gamma = 0.5$.

Security objective	Selection Function S	Reward Function R
Resistance to selfish mining	Must be non-forward settable	Use the punishment fee
Bounded monetary supply	The selected block must be rewarded	Only reward the first N uncles
Include uncles	-	Give an extra reward for including uncles

Table 1: Security objectives

We can make the following observations:

- The security objectives are not necessarily independent. For example incentivizing the inclusion of uncles indeed contributes to making selfish mining attacks less practical (see Section 3.2.4).
- A special care needs to be taken when picking constants such as the punishment fee p, or the inclusion reward i, as they may lead to the wrong incentives. For example if the inclusion reward i is such that $\frac{\mathtt{r}(1-\mathtt{p})}{N+1} << \mathtt{i}N$ then selfish-mining attacks may remain practical for low values of α.
- The selection function S and the reward function R are related to each other. For example when bounding the number of uncles (see Section 3.2.2) it is important to ensure that the selected block $X = \mathtt{S}(X_1, X_2, ..., X_N)$ is included as well otherwise a block included in the main chain will not be rewarded.

3.3 Implementation as a soft-fork

When implemented in Bitcoin as a soft-fork, a reference to an uncle header can be stored in the scriptpub of an output of the coinbase transaction (e.g. in OP_RETURN payload), or in the coinbase field. To allow reward sharing, the coinbase transaction must pay new rewards to scriptpub OP_DROP OP_TRUE, so all coinbases scriptpub can be easily grabbed by miners. However a soft-fork prevents anyone from grabbing them. The block where a coinbase becomes mature must include a special Coinbase Split Transaction (CST) that splits the matured reward exactly as the DECOR rule establishes, based on previously published uncles and the dropped scriptpub scripts. A block missing the CST or having an erroneous CST is considered invalid.

4 LAMI

Bitcoin forwards each block by packing the block header with all the transactions contained in the block. This strategy, while being the most easy to analyze, is known to perform badly both regarding block propagation latency and bandwidth usage, which is doubled. Bitcoin miners partially solved this problem using the Fast Relay Network (FRN): FRN is a centralized backbone that relays blocks in a compressed form, and is maintained by a single user. The FRN provides more fairness to miners, as it prevents bigger mining pools for taking advantage of better network connectivity. However, for the same reason, the FRN can be a target of attack by a big pool. Since the FRN is a non-profit community service, the attack can take several forms, from denial-of-service attacks to any social engineering techniques.

Therefore we propose the following changes in the protocol for propagating blocks efficiently while not relying on a centralized party:

1. Header First propagation (HEFIP)

2. Temporary Mining on Unverified Parent Blocks (TEMUP) also strangely referred as "SPV Mining".

4.1 Header-first propagation of blocks (HEFIP)

We propose that blocks are sent in two stages: in the first stage only the block header is sent. Then the full block is sent (any block compression algorithm such as ILBTs also helps). Block headers are verified at every node and if correct, they are pushed into peers without an INV round-trip. Headers up to 6 blocks old (as specified by its height) are broadcast. This gives enough time to miners for including stale headers as uncles in blocks. Since the transactions in a block are generally already known to the network, there is no benefit in transmitting them again. Using 2SBP the channel capacity is doubled, allowing more transactions to be stored in each block. After each node has received the block header and the transaction hash list associated with the block header, the node attempts to reconstruct the block in order to verify it. The peer will fetch from a peer any transaction contained in the block but missing in his transaction pool.

4.2 Temporary Mining on unverified parents (TEMUP)

Nodes can then start mining an empty block (coinbase only) on top of a header even if the transactions are still missing during a fixed interval. After that interval, they must resume mining with whatever block they were mining before. Most of the time no block is solved before the transactions arrive, and the does not solve an empty block. Even if the average block interval is reduced to 30 seconds, empty blocks are produced with very low probability and they do not affect the bandwidth and block-chain storage usage, because of their small size.

5 Conclusion

In this work we introduced DECOR+LAMI: DECOR is a framework to improve the scalability and security of blockchain protocols which rely on two ideas: (1) provide a deterministic way for solving conflicts when mining and (2) align block rewards computation with the expected behavior of the protocol. We believe that DECOR can help in particular to decrease the block time interval and make selfish mining attacks less effective. A main precondition for using DECOR is that the miners share almost the same view of the published block of the blockchain at any point of time. In order to achieve this goal we introduced LAMI, a set of implementation optimizations that reduces latency in block propagation. Future lines of research include:

- Formalizing the class of reward functions `R` and analyzing how such functions can be used for shaping the behavior of miners towards specific goals. In particular study how generic selfish mining attacks could be prevented by using an appropriate reward mechanism.
- Understanding the relationship between the selection function and reward function.
- Providing new instantiations of DECOR and compare them to existing solutions.

A Selfish mining analysis

A.1 Idea of the attack

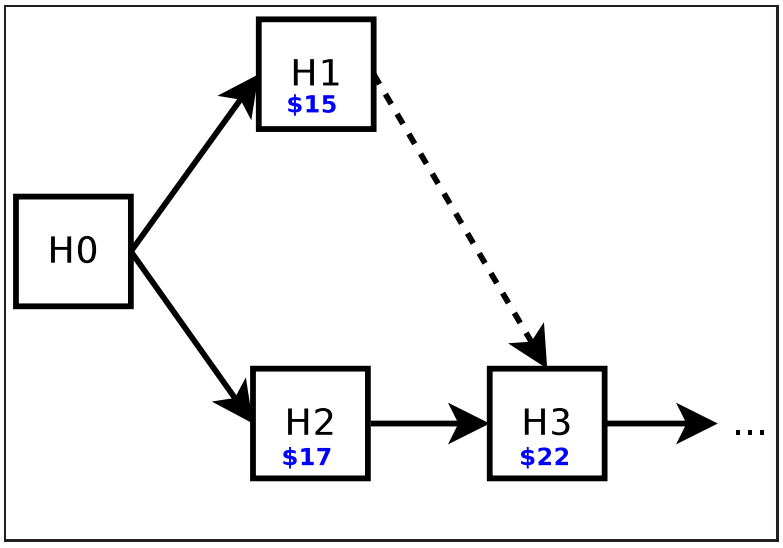

Figure 4: Before the attack

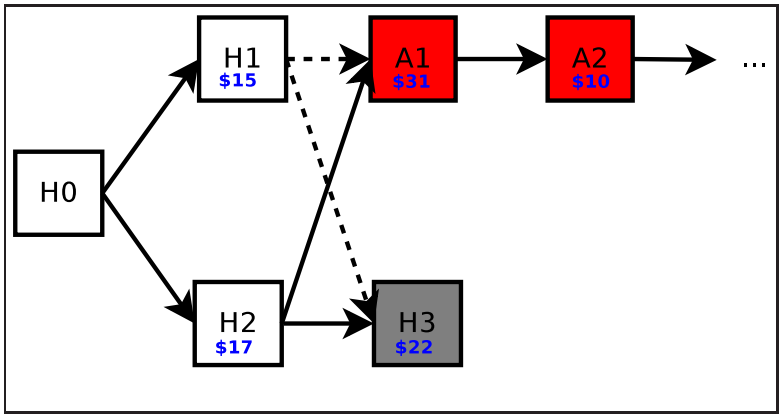

Figure 5: After the attack

Consider the following scenario (see Figure 4): Assume that blocks $H0$, $H1$, $H2$ and $H3$ have been produced by honest miners that follow an inclusive blockchain protocol.

Assume that when presented the alternative to mine on top of $H1$ or $H2$, the honest miners decide to mine of top of $H2$. Then according to the inclusion blockchain protocol, the uncle $H1$ is included as a reference in the block $H3$ and thus the miner who produced $H1$ will also be rewarded.

The idea of selfish mining is to force honest miners to work on blocks that will not get rewarded. By doing so the selfish miner is able to get more reward for the same computational power compared to honest miners. By providing an incentive to include uncles, it may seem that inclusive blockchain protocols make it more difficult to perform selfish mining, as uncles nodes – while not being part of the main chain – will still be assigned a reward.

However the fact that only uncles are referenced by blocks of the main chain still opens the path for selfish mining attacks.

In order to illustrate the claim above let us consider (see Figure 5) that an attacker mines privately blocks $A1$ and $A2$ on top of $H1$ and $A1$ respectively. The selfish miner will thus try to produce two blocks consecutively while following the protocol.

Note that in Figure 5, indeed the selfish miner – like the honest ones – mines on top (plain line) of $H1$ and includes a references to $H2$ (dashed line) the uncle.

After successfully mining two blocks in a row the chain $H0, H1, A1, A2$ becomes the longest and thus all the miners will keep mining on top of $A2$. The problem that occurs is that $H3$ which has been mined honestly will not be referenced as an uncle and thus not be assigned any reward. This means that the computational power invested in order to produce $H3$ will be wasted and thus the selfish mining attack is successful.

The state machine depicted in Figure 6 models the possible configurations and their respective probabilities of the selfish mining strategy described above. The states are:

- **0**: The selfish and honest pools follow the main chain.
- **U0**: There is an uncle which happens with probability θ.
- **1,2,...**: The selfish pool is ahead of the main chain by $1, 2, ...$ blocks.
- **0'**: The selfish pool and the honest pool are competing with their respective chain of length 1 (from the fork).

A.2 Probabilities

By analyzing the state machine depicted in Figure 6, we obtain the following equations:

$$\theta p_0 = (1-\alpha)p_{U_0} + (1-\alpha)p_1 + (1-\alpha)p_2 \quad (1)$$
$$p_{U_0} = \theta p_0 \quad (2)$$
$$p_1 = \alpha p_{U_0} \quad (3)$$
$$p_{0'} = (1-\alpha)p_1 \quad (4)$$
$$\alpha p_1 = (1-\alpha)p_2 \quad (5)$$
$$\forall k \geq 2 : \alpha p_k = (1-\alpha)p_{k+1} \quad (6)$$
$$\sum_{k=0}^{\infty} p_k + p_{0'} + p_{U_0} = 1 \quad (7)$$

We obtain (5) by observing that (1) $\Rightarrow \theta p_0 = \theta p_0 - \alpha p_{U_0} + (1-\alpha)p_1 + (1-\alpha)p_2$. Then by replacing αp_{U_0} with p_1 we obtain $p_1 = (1-\alpha)p_1 + (1-\alpha)p_2$ so we can deduce that $\alpha p_1 = (1-\alpha)p_2$.

From (5) and (6) we can deduce

$$\forall k \geq 2 : p_k = \left(\frac{\alpha}{1-\alpha}\right)^{k-1} p_1 \quad (8)$$

If we replace expression for p_2 from Equation (8) into equation (1) we obtain:

$$\theta p_0 = (1-\alpha)p_{U_0} + (1-\alpha)p_1 + (1-\alpha)\frac{\alpha}{1-\alpha}p_1 \quad (9)$$
$$\theta p_0 = (1-\alpha)p_{U_0} + p_1 \quad (10)$$
$$p_0 = \frac{1}{\alpha\theta}p_1 \quad (11)$$

```
on Init:
        public_chain := publicly known blocks
        private_chain := publicly known blocks
        private_branch_len := 0

on Fork: //That is there will be an uncle
        Pick the block with the lowest reward // (uncle)
        Start mining privately on top of this block

on My Pool found a block:
        Δ_prev := length(private_chain)−length(public_chain)
        append new block to private chain
                if Δ_prev = 1: //Avoid the punishment fee
                        reference uncle
        private_branch_len := private_branch_len +1
        if (Δ_prev = 0) and (private_branch_len = 2):
                publish all the private chain
                private_chain_len := 0

on Others found a block:
        Δ_prev := length(private_chain) − length(public_chain)
        append new block to public chain
        if Δ_prev = 0:
                private_chain := public_chain
                private_branch_len := 0
        elif Δ_prev = 1:
                publish last block or private chain
                  // Try to use a high fee so
                  // that this chain wins the official one
        elif Δ_prev = 2:
                publish all private chain
                private_chain_length := 0
        else:
                publish first unpublished block
                keep mining on top of private chain
```

Listing 6: Selfish mining strategy for inclusive blockchain protocols

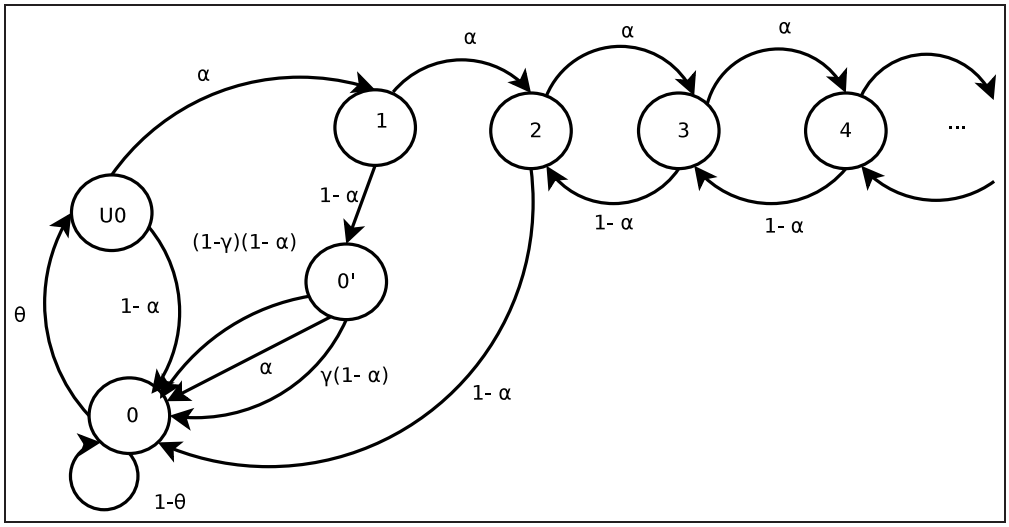

Figure 6: State machine with transition frequencies

Now can can express all probabilities defined in (7) in function of p_1 using Equations (8), (2) and (11):

$$\sum_{k=0}^{\infty} p_k + p_{0'} + p_{U_0} = 1 \tag{12}$$

$$p_0 + \sum_{k=1}^{\infty} \left(\frac{\alpha}{1-\alpha}\right)^{k-1} p_1 + p_{0'} + p_{U_0} = 1 \tag{13}$$

$$\frac{1}{\alpha\theta}p_1 + \frac{1-\alpha}{1-2\alpha}p_1 + (1-\alpha)p_1 + \frac{1}{\alpha}p_1 = 1 \tag{14}$$

$$(1-2\alpha)\alpha p_1 + \alpha^2\theta p_1 + \alpha^2\theta(1-2\alpha)(1-\alpha)p_1 + \alpha\theta(1-2\alpha)p_1 = \alpha^2\theta(1-2\alpha) \tag{15}$$

$$\frac{\alpha^2\theta(1-2\alpha)}{(1-2\alpha)\alpha + \alpha^2\theta + \alpha^2\theta(1-2\alpha)(1-\alpha) + \alpha\theta(1-2\alpha)} = p_1 \tag{16}$$

$$\frac{\alpha^2\theta - 2\alpha^3\theta}{\alpha - 2\alpha^2 + \alpha^2\theta + (\alpha^2\theta - 2\alpha^3\theta)(1-\alpha) + \alpha\theta - 2\alpha^2\theta} = p_1 \tag{17}$$

$$\frac{\alpha^2\theta - 2\alpha^3\theta}{\alpha - 2\alpha^2 - 3\alpha^3\theta + 2\alpha^4\theta + \alpha\theta} = p_1 \tag{18}$$

We now can deduce all probabilities:

$$p_0 = \frac{(1-2\alpha)}{2\alpha^3\theta - 3\alpha^2\theta - 2\alpha + \theta + 1} \tag{19}$$

$$p_1 = \frac{\theta(\alpha - 2\alpha^2)}{2\alpha^3\theta - 3\alpha^2\theta - 2\alpha + \theta + 1} \tag{20}$$

$$p_2 = \frac{\theta(\alpha^2 - 2\alpha^3)}{-2\alpha^4\theta + 5\alpha^3\theta - \alpha^2(3\theta + 2) - \alpha(\theta - 1) + \theta + 1} \tag{21}$$

$$p_k, k \geq 3 = \left(\frac{\alpha}{1-\alpha}\right)^{k-1} \frac{\theta(\alpha - 2\alpha^2)}{2\alpha^3\theta - 3\alpha^2\theta - 2\alpha + \theta + 1} \tag{22}$$

$$p_{0'} = \frac{\theta(2\alpha^3 - 3\alpha^2 + \alpha)}{2\alpha^3\theta - 3\alpha^2\theta - 2\alpha + \theta + 1} \tag{23}$$

$$p_{U_0} = \frac{\theta(1-2\alpha)}{2\alpha^3\theta - 3\alpha^2\theta - 2\alpha + \theta + 1} \tag{24}$$

A.3 Revenue

For the sake of clarity assume that the reward to be shared between blocks is 1. For example in the case of a chain with no uncles / forks then each block will be assign reward 1. In the case there is an uncle, then the uncle will receive reward 0.5 and the block at the tip of the chain will also be assigned reward 0.5.

Note that this convention does not affect the final result (relative revenue) as what matters is the ratio between the revenue of honest and selfish miners respectively.

In the following table we do not consider the revenue obtained with block $H1$ and $H2$ because the attack of the adversary begins *after* these two blocks are produced and are thus not part of the *relative* revenue calculation.

State	**Transition**	**Honest**	**Adversary**	**Adversary with punishment fee (p)**
0	θ	0 (to be determined later)	0	0
0	$1-\theta$	0 (reset of the game, adversary did not start attack)	0	0
$U0$	α	0	0 (to be determined later)	0
$U0$	$1-\alpha$	0.5 (rewards are shared)	0	0
1	α	0	0 (to be determined later)	0
1	$1-\alpha$	0 (to be determined later)	0 (to be determined later)	0
$0'$	α	0	1.5	$1 + .5\mathtt{p}$
$0'$	$\gamma(1-\alpha)$	1	0.5	$0.5\mathtt{p}$
$0'$	$(1-\gamma)(1-\alpha)$	1.5	0	0
2	α	0 (to be determined later)	0 (to be determined later)	0
2	$1-\alpha$	0	1.5	$1 + 0.5\mathtt{p}$
3	α	0 (to be determined later)	0 (to be determined later)	0
3	$1-\alpha$	0 (to be determined later)	0 (to be determined later)	0

Table 2: Revenue based on states and transitions.

The relative revenue formula is:

$$R = \frac{r_a}{r_a + r_h} \tag{25}$$

$$= \frac{1.5p_{0'}\alpha + 0.5\gamma(1-\alpha)p_{0'} + 1.5(1-\alpha)p_2}{0.5(1-\alpha)p_{U_0} + \gamma(1-\alpha)p_{0'} + 1.5(1-\gamma)(1-\alpha)p_{0'} + 1.5p_{0'}\alpha + 0.5\gamma(1-\alpha)p_{0'} + 1.5(1-\alpha)p_2} \tag{26}$$

The curve for the relative revenue R is plotted in Figure 2. We can make the following observations:

- R does not depend on θ as the attacker waits for an uncle to appear to start the attack.
- The mining strategy is profitable when:
  - $\alpha > 0$ when $\gamma = 1$
  - $\alpha > 0.21$ when $\gamma = 0.5$
  - $\alpha > 0.34$ when $\gamma = 0$

References

[1] Ali, M., Nelson, J., Labs, B., Shea, R., Labs, B., Freedman, M. J., Ali, M., Nelson, J., Shea, R., and Freedman, M. J. Blockstack : A Global Naming and Storage System Secured by Blockchains. In *USENIX Annual Technical Conference* (Denver, CO, jun 2016), USENIX Association, pp. 181–194.

[2] Ben-Sasson, E., Chiesa, A., Garman, C., Green, M., Miers, I., Tromer, E., and Virza, M. Zerocash: Decentralized Anonymous Payments from Bitcoin. In *2014 IEEE Symposium on Security and Privacy*. 2014, pp. 459—-474.

[3] Buterin, V. Ethereum - White Paper. https://www.ethereum.org/pdfs/EthereumWhitePaper.pdf, 2015.

[4] Canetti, R., Goldreich, O., and Halevi, S. The Random Oracle Methodology, Revisited. *Journal of the ACM 51*, 4 (jul 2004), 557–594.

[5] Eyal, I., Gencer, A. E., Sirer, E. G., and van Renesse, R. Bitcoin-NG: A Scalable Blockchain Protocol. In *13th USENIX Symposium on Networked Systems Design and Implementation (NSDI 16)*. 2016, pp. 45–49.

[6] Eyal, I., and Sirer, E. G. Majority is not enough: Bitcoin mining is vulnerable. In *International Conference on Financial Cryptography and Data Security*, vol. 8437. 2014, pp. 436–454.

[7] Garay, J., Kiayias, A., and Leonardos, N. The Bitcoin Backbone Protocol: Analysis and Applications. In *Advances in Cryptology - EUROCRYPT 2015*, E. Oswald and M. Fischlin, Eds. Springer Berlin / Heidelberg, 2015, pp. 281–310.

[8] Kiayias, A., and Panagiotakos, G. Speed-Security Tradeoffs in Blockchain Protocols. https://eprint.iacr.org/2015/1019, 2015.

[9] Kiayias, A., and Panagiotakos, G. On Trees, Chains and Fast Transactions in the Blockchain. http://eprint.iacr.org/2016/545, 2016.

[10] Kosba, A., Miller, A., Shi, E., Wen, Z., and Papamanthou, C. Hawk: The Blockchain Model of Cryptography and Privacy-Preserving Smart Contracts. eprint.iacr.org/2015/675.pdf, 2015.

[11] Lerner, S. D. Uncle Mining: An Ethereum consensus protocol flaw. https://bitslog.wordpress.com/2016/04/28/uncle-mining-an-ethereum-consensus-protocol-flaw/, 2016.

[12] Lewenberg, Y., Sompolinsky, Y., and Zohar, A. Inclusive block chain protocols. In *International Conference on Financial Cryptography and Data Security*, vol. 8975. Springer, 2015, pp. 528–547.

[13] Nakamoto, S. Bitcoin: A Peer-to-Peer Electronic Cash System. https://bitcoin.org/bitcoin.pdf, 2008.

[14] Pass, R., Seeman, L., and Shelat, A. Analysis of the Blockchain Protocol in Asynchronous Networks. http://eprint.iacr.org/2016/454.pdf, 2016.

[15] Peck, M. E. Adam Back Says the Bitcoin Fork Is a Coup. http://spectrum.ieee.org/tech-talk/computing/networks/the-bitcoin-for-is-a-coup, 2015.

[16] Sompolinsky, Y., and Zohar, A. Secure high-rate transaction processing in bitcoin. In *International Conference on Financial Cryptography and Data Security*, vol. 8975. Springer Berlin Heidelberg, 2015, pp. 507–527.

[17] Wuille, P. Segregated Witness and its Impact on Scalability. https://www.youtube.com/watch?v=NOYNZB5BCHM, 2015.