Chapter 27

Advanced Blockchain

In this chapter we study various advanced blockchain concepts, which are popular in research.

27.1 Selfish Mining

Satoshi Nakamoto suggested that it is rational to be altruistic, e.g., by always attaching newly found block to the longest chain. But is it true?

Definition 27.1 (Selfish Mining)**.** *A selfish miner hopes to earn the reward of a larger share of blocks than its hardware would allow. The selfish miner achieves this by temporarily keeping newly found blocks secret.*

Algorithm 27.2 Selfish Mining

```
1: Idea: Mine secretly, without immediately publishing newly found blocks
2: Let $d_p$ be the depth of the public blockchain
3: Let $d_s$ be the depth of the secretly mined blockchain
4: if a new block $b_p$ is published, i.e., $d_p$ has increased by 1 then
5:   if $d_p > d_s$ then
6:     Start mining on that newly published block $b_p$
7:   else if $d_p = d_s$ then
8:     Publish secretly mined block $b_s$
9:     Mine on $b_s$ and publish newly found block immediately
10:  else if $d_p = d_s - 1$ then
11:    Publish all secretly mined blocks
12:  end if
13: end if
```

Theorem 27.3 (Selfish Mining)**.** *It may be rational to mine selfishly, depending on two parameters α and γ, where α is the ratio of the mining power of the selfish miner, and γ is the share of the altruistic mining power the selfish miner can reach in the network if the selfish miner publishes a block right after seeing a newly published block. Precisely, the selfish miner share is*

$$\frac{\alpha(1-\alpha)^2(4\alpha+\gamma(1-2\alpha))-\alpha^3}{1-\alpha(1+(2-\alpha)\alpha)}.$$

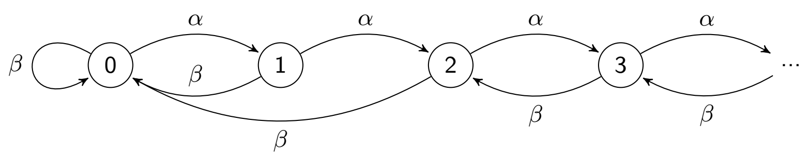

Figure 27.4: Each state of the Markov chain represents how many blocks the selfish miner is ahead, i.e., $d_s - d_p$. In each state, the selfish miner finds a block with probability α, and the honest miners find a block with probability $\beta = 1 - \alpha$. The interesting cases are the "irregular" β arrow from state 2 to state 0, and the β arrow from state 1 to state 0 as it will include three subcases.

Proof. We model the current state of the system with a Markov chain, see Figure 27.4.

We can solve the following Markov chain equations to figure out the probability of each state in the stationary distribution:

$$\begin{aligned} p_1 &= \alpha p_0 \\ \beta p_{i+1} &= \alpha p_i, \text{ for all } i > 1 \\ \text{and } 1 &= \sum_i p_i. \end{aligned}$$

Using $\rho = \alpha/\beta$, we express all terms of above sum with p_1:

$$1 = \frac{p_1}{\alpha} + p_1 \sum_{i \geq 0} \rho^i = \frac{p_1}{\alpha} + \frac{p_1}{1-\rho}, \text{ hence } p_1 = \frac{2\alpha^2 - \alpha}{\alpha^2 + \alpha - 1}.$$

Each state has an outgoing arrow with probability β. If this arrow is taken, one or two blocks (depending on the state) are attached that will eventually end up in the main chain of the blockchain. In state 0 (if arrow β is taken), the honest miners attach a block. In all states i with $i > 2$, the selfish miner eventually attaches a block. In state 2, the selfish miner directly attaches 2 blocks because of Line 11 in Algorithm 27.2.

State 1 in Line 8 is interesting. The selfish miner secretly was 1 block ahead, but now (after taking the β arrow) the honest miners are attaching a competing block. We have a race who attaches the next block, and where. There are three possibilities:

- Either the selfish miner manages to attach another block to its own block, giving 2 blocks to the selfish miner. This happens with probability α.
- Or the honest miners attach a block (with probability β) to their previous honest block (with probability $1 - \gamma$). This gives 2 blocks to the honest miners, with total probability $\beta(1 - \gamma)$.
- Or the honest miners attach a block to the selfish block, giving 1 block to each side, with probability $\beta\gamma$.

The blockchain process is just a biased random walk through these states. Since blocks are attached whenever we have an outgoing β arrow, the total number of blocks being attached per state is simply $1+p_1+p_2$ (all states attach a single block, except states 1 and 2 which attach 2 blocks each).

As argued above, of these blocks, $1-p_0+p_2+\alpha p_1-\beta(1-\gamma)p_1$ are blocks by the selfish miner, i.e., the ratio of selfish blocks in the blockchain is

$$\frac{1-p_0+p_2+\alpha p_1-\beta(1-\gamma)p_1}{1+p_1+p_2}.$$

□

Remarks:

- If the miner is honest (altruistic), then a miner with computational share α should expect to find an α fraction of the blocks. For some values of α and γ the ratio of Theorem 27.3 is higher than α.
- In particular, if $\gamma = 0$ (the selfish miner only wins a race in Line 8 if it manages to mine 2 blocks in a row), the break even of selfish mining happens at $\alpha = 1/3$.
- If $\gamma = 1/2$ (the selfish miner learns about honest blocks very quickly and manages to convince half of the honest miners to mine on the selfish block instead of the slightly earlier published honest block), already $\alpha = 1/4$ is enough to have a higher share in expectation.
- And if $\gamma = 1$ (the selfish miner controls the network, and can hide any honest block until the selfish block is published) any $\alpha > 0$ justifies selfish mining.

27.2 Ethereum

Definition 27.5 (Ethereum)**.** *Ethereum is a distributed state machine. Unlike Bitcoin, Ethereum promises to run arbitrary computer programs in a blockchain.*

Remarks:

- Like the Bitcoin network, Ethereum consists of nodes that are connected by a random virtual network. These nodes can join or leave the network arbitrarily. There is no central coordinator.
- Like in Bitcoin, users broadcast cryptographically signed transactions in the network. Nodes collate these transactions and decide on the ordering of transactions by putting them in a block on the Ethereum blockchain.

Definition 27.6 (Smart Contract)**.** *Smart contracts are programs deployed on the Ethereum blockchain that have associated storage and can execute arbitrarily complex logic.*

Remarks:

- Smart Contracts are written in higher level programming languages like Solidity, Vyper, etc. and are compiled down to EVM (Ethereum Virtual Machine) bytecode, which is a Turing complete low level programming language.
- Smart contracts cannot be changed after deployment. But most smart contracts contain mutable storage, and this storage can be used to adapt the behavior of the smart contract. With this, many smart contracts can update to a new version.

Definition 27.7 (Account)**.** *Ethereum knows two kinds of accounts. Externally Owned Accounts (EOAs) are controlled by individuals, with a secret key. Contract Accounts (CAs) are for smart contracts. CAs are not controlled by a user.*

Definition 27.8 (Ethereum Transaction)**.** *An Ethereum transaction is sent by a user who controls an EOA to the Ethereum network. A transaction contains:*

- *Nonce: This "number only used once" is simply a counter that counts how many transactions the account of the sender of the transaction has already sent.*
- *160-bit address of the recipient.*
- *The transaction is signed by the user controlling the EOA.*
- *Value: The amount of Wei (the native currency of Ethereum) to transfer from the sender to the recipient.*
- *Data: Optional data field, which can be accessed by smart contracts.*
- *StartGas: A value representing the maximum amount of computation this transaction is allowed to use.*
- *GasPrice: How many Wei per unit of Gas the sender is paying. Miners will probably select transactions with a higher GasPrice, so a high GasPrice will make sure that the transaction is executed more quickly.*

Remarks:

- There are three types of transactions.

Definition 27.9 (Simple Transaction)**.** *A simple transaction in Ethereum transfers some of the native currency, called Wei, from one EOA to another. Higher units of currency are called Szabo, Finney, and Ether, with* 10^{18} *Wei* = 10^6 *Szabo* = 10^3 *Finney = 1 Ether. The data field in a simple transaction is empty.*

Definition 27.10 (Smart Contract Creation Transaction)**.** *A transaction whose recipient address field is set to* 0 *and whose data field is set to compiled EVM code is used to deploy that code as a smart contract on the Ethereum blockchain. The contract is considered deployed after it has been mined in a block and is included in the blockchain at a sufficient depth.*

Definition 27.11 (Smart Contract Execution Transaction)**.** *A transaction that has a smart contract address in its recipient field and code to execute a specific function of that contract in its data field.*

Remarks:

- Smart Contracts can execute computations, store data, send Ether to other accounts or smart contracts, and invoke other smart contracts.
- Smart contracts can be programmed to self destruct. This is the only way to remove them again from the Ethereum blockchain.
- Each contract stores data in 3 separate entities: storage, memory, and stack. Of these, only the storage area is persistent between transactions. Storage is a key-value store of 256 bit words to 256 bit words. The storage data is persisted in the Ethereum blockchain, like the hard disk of a traditional computer. Memory and stack are for intermediate storage required while running a specific function, similar to RAM and registers of a traditional computer. The read/write gas costs of persistent storage is significantly higher than those of memory and stack.

Definition 27.12 (Gas)**.** *Gas is the unit of an atomic computation, like swapping two variables. Complex operations use more than 1 Gas, e.g., ADDing two numbers costs 3 Gas.*

Remarks:

- As Ethereum contracts are programs (with loops, function calls, and recursions), end users need to pay more gas for more computations. In particular, smart contracts might call another smart contract as a subroutine, and StartGas must include enough gas to pay for all these function calls invoked by the transaction.
- The product of StartGas and GasPrice is the maximum cost of the entire transaction.
- Transactions are an all or nothing affair. If the entire transaction could not be finished within the StartGas limit, an Out-of-Gas exception is raised. The state of the blockchain is reverted back to its values before the transaction. The amount of gas consumed is not returned back to the sender.

Definition 27.13 (Block)**.** *In Ethereum, like in Bitcoin, a block is a collection of transactions that is considered a part of the canonical history of transactions. Among other things, a block contains: pointers to parent and up to two uncles, the hash of the root node of a trie structure populated with each transaction of the block, the hash of the root node of the state trie (after transactions have been executed)*

Chapter Notes

Selfish mining has already been discussed shortly after the introduction of Bitcoin [RHo10]. A few years later, Eyal and Sirer formally analyzed selfish mining [ES14]. If the selfish miner is two or more blocks ahead, this original research suggested to always answer a newly published block by releasing the oldest unpublished block, so have two blocks at the same level. The idea was that honest miners will then split their mining power between these two blocks. However, what matters is how long it takes the honest miners to find the next block to extend the public blockchain. This time does not change whether the honest miners split their efforts or not. Hence the case $d_p < d_s - 1$ is not needed in Algorithm 27.2.

Similarly, Courtois and Bahack [CB14] study subversive mining strategies. Nayak et al. [NKMS15] combine selfish mining and eclipse attacks. Algorithm 27.2 is not optimal for all parameters, e.g., sometimes it may be beneficial to risk even a two-block advantage. Sapirshtein et al. [SSZ15] describe and analyze the optimal algorithm.

Vitalik Buterin introduced Ethereum in the 2013 whitepaper [But13]. In 2014, Ethereum Foundation was founded to create Ethereum's first implementation. An online crowd-sale was conducted to raise around 31,000 BTC (around USD 18 million at the time) for this. In this sense, Ethereum was the first ICO (Initial Coin Offering). Ethereum has also attempted to write a formal specification of its protocol in their yellow paper [Gav18]. This is in contrast to Bitcoin, which doesn't have a formal specification.

Bitcoin's blockchain forms as a chain, i.e., each block (except the genesis block) has a parent block. The longest chain with the highest difficulty is considered the main chain. GHOST [SZ15] is an alternative to the longest chain rule for establishing consensus in PoW based blockchains and aims to alleviate adverse impacts of stale blocks. Ethereum's blockchain structure is a variant of GHOST. Other systems based on DAGs have been proposed in [SLZ16], [SZ18], [LLX$^+$18], and [LSZ15].

Bibliography

[But13] Vitalik Buterin. A Next-Generation Smart Contract and Decentralized Application Platform, 2013. Available from: `https://github.com/ethereum/wiki/wiki/White-Paper`.

[CB14] Nicolas T. Courtois and Lear Bahack. On subversive miner strategies and block withholding attack in bitcoin digital currency. *CoRR*, abs/1402.1718, 2014.

[ES14] Ittay Eyal and Emin Gün Sirer. Majority is not enough: Bitcoin mining is vulnerable. In *Financial Cryptography and Data Security*, pages 436–454. Springer, 2014.

[Gav18] Gavin Wood. Ethereum: A Secure Decentralised Generalised Transaction Ledger, Byzantium Version, 2018. Available from: `https://ethereum.github.io/yellowpaper/paper.pdf`.

[LLX$^+$18] Chenxing Li, Peilun Li, Wei Xu, Fan Long, and Andrew Chi-Chih Yao. Scaling nakamoto consensus to thousands of transactions per second. *CoRR*, abs/1805.03870, 2018.

[LSZ15] Yoad Lewenberg, Yonatan Sompolinsky, and Aviv Zohar. Inclusive block chain protocols. In *Financial Cryptography and Data Security*, pages 528–547. Springer, 2015.

[NKMS15] Kartik Nayak, Srijan Kumar, Andrew Miller, and Elaine Shi. Stubborn mining: Generalizing selfish mining and combining with an eclipse attack. Technical report, IACR Cryptology ePrint Archive 2015, 2015.

[RHo10] RHorning. Mining cartel attack, 2010.

[SLZ16] Yonatan Sompolinsky, Yoad Lewenberg, and Aviv Zohar. Spectre: A fast and scalable cryptocurrency protocol. Cryptology ePrint Archive, Report 2016/1159, 2016. `https://eprint.iacr.org/2016/1159`.

[SSZ15] Ayelet Sapirshtein, Yonatan Sompolinsky, and Aviv Zohar. Optimal selfish mining strategies in bitcoin. *arXiv preprint arXiv:1507.06183*, 2015.

[SZ15] Yonatan Sompolinsky and Aviv Zohar. Secure high-rate transaction processing in bitcoin. In *Financial Cryptography and Data Security*, pages 507–527. Springer, 2015.

[SZ18] Yonatan Sompolinsky and Aviv Zohar. Phantom: A scalable blockdag protocol. Cryptology ePrint Archive, Report 2018/104, 2018. `https://eprint.iacr.org/2018/104`.