ECE 595: Foundations of Blockchain Systems	**November 3, 2020**

Lecture 10

Lecturer: Sreeram Kannan — *Scribe: Milad Fotouhi*

Outline: So far we have checked the security of longest chain protocols and as an example we checked the one that has been deployed by bitcoin. We also talked about safety and liveness of the longest chain under a general network latency modeling. The throughput and latency of bitcoin also has been discussed. In this lecture we will revisit throughput and latency to check for the area and scopes of improvement. We try to solve throughput and latency problem on longest chain.
We will also explain Prism [2], as an approach which takes the longest chain and converts it into something much better performing in terms of latency and throughput but still retaining the security properties.

10.1 Bitcoin Performance

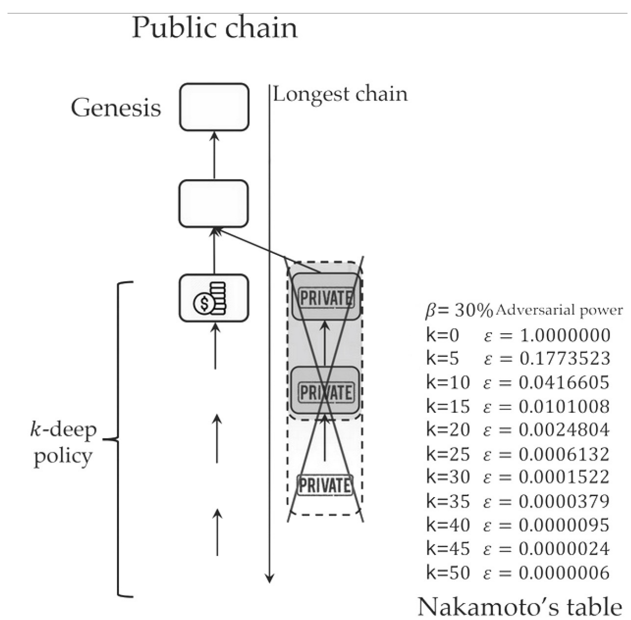

Figure 10.1: Illustration of k-deep policy.

Previously we have discussed that one of the biggest advantages of the longest chain protocol, particularly Bitcoin, is that it has been secure under an immense threat model that is under an entire permission-less system and theoretically it is known that it is secure against 50 percent adversary (this already has been proved in last section). But practically, this has really withstood the test of time. So there will be no attacks on Bitcoin however, there are other attributes of bitcoin, which are not as appealing, for example confirmation latency can be in the order of hours. Also, throughput (how many transactions per second can we process) is in the order of a few transactions per second. In contrast we can consider Visa or MasterCard, which can support a throughput of $10,000$ transactions per second with the latency of a few hundred milliseconds.

To understand this better lets recall that when we get a transaction that is accepted into a block, we do not immediately confirm that transaction because it's too risky. Our transaction needs to be deeply embedded into the longest chain for the transaction to be confirmed which is called the k-deep confirmation policy. As we discussed in section 8 the longer the k is, the better adversary protection. When we increase k, as long as the adversary has less than 50 percent adversity power, the probability that the adversary can construct more than K blocks faster than the honest notes can construct a block can be calculated. This is a race between two distinct random variables. Satoshi Nakamoto modeled this in the original Bitcoin paper and this probabilities has been actually calculated there.

As you can see in the figure 10.1, the table is showing that if we want very high availability, then we need to wait for K to become equals to 30 blocks. Considering the entire block time in Bitcoin being 10 minutes then the overall latency here will be 300 minutes. This is the main cost of latency in bitcoin. This latency is mainly due to two things. One is k which is the confirmation of depth that the other is intro block arrival which is 10 minutes. Reducing K will reduce reliability, and if we reduce the arrival time from 10 minutes to several seconds there will be more tries there will be more forking. Hence, we cannot increase performance without reducing the reliability or reducing the safety threshold.

As fig.10.2 illustrates, to make our entire block proposal time to be smaller than Δ we should have multiple nodes that proposed block simultaneously. Then we will have forking and it leads to slow growth of the longest chain because even though nine blocks having produced in this figure, the length has only grown by two. And this is a reduction in the security because now an adversary who does not even have 50 percent adversarial power can make four blocks during that time, but they're all coordinated, which means they're in the longest chain. Hence, there is a reduction in the security threshold as you increase the mining rate.

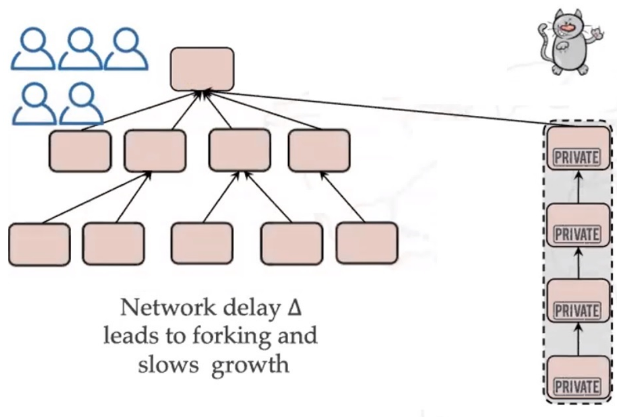

Figure 10.2: Bitcoin performance and security trade-off.

10.2 Existing Scalability Approaches

Native PoW	Convert to Permissioned
GHOST, Inclusive, Spectre, Phantom, Conflux, IOTA Tangle	Hybrid consensus, Thunderella, ByzCoin can scale
No protocol achieves both: • Provable security • Fast latency	Not secure against • Adaptive adversaries

Figure 10.3: Existing scalability approaches

There are a set of existing approaches which have been trying to improve the scalability problem that include GHOST [1], Inclusive [3], Spectre [6], Phantom [7], Conflux [4] and IOTA Tangle [5]. However, none of these protocols achieves both provable security and fast latency. There are also some approaches that convert the problem from a permission-less protocol to permissioned problem. Hybrid consensus, Thunderella, ByzCoin can scale are some examples of these approaches. The rough idea in these approaches is that you let everybody participate in approval work so they all can mine blocks. Here the miners of the previous blocks will form a committee, this committee is fixed in size and not prone to any Sybil attack. Now we can run some protocol inside this committee and convert the permission-less protocol into a permissioned problem. The permission will be given based on the who min more blocks during the previous stage. So the miners who have been mining previously now get to propose and form a committee and you just run a permissioned protocol inside that committee.

10.2.1 Adaptive Adversary

Adaptive adversary is an adversity that knows the entire state of the work, till this point; it can go on bribe whichever nodes it wants. In a purely mathematical model, we can think of it as the node can just go and somehow corrupt and control whichever node it wants. In a native permission-less PoW system, as long as the corrupted group of nodes controls less than 50 percent of the computing power, the protocol is secure. This is because, in a proof of work system after you make a block you have no power in the future to change the block. Even an adaptive adversary can't bribe the previous miners and change contents of the block. The only hope of the adversity is it has to bribe or corrupt enough nodes so that it actually controls 50 percent of the compute power right now.

10.2.2 Permissioned Protocol Vulnerability Against Adaptive Adversary

So as we said these approaches indeed convert the much harder problem, permission-less consensus, into a simpler permissioned consensus problem, but it will make it vulnerable to an adaptive adversary. Just consider adversity who can bribe the last hundred miners who mine the blocks and have the future power in this permissioned network. And so adaptive adversity can exploit this behavior because it can just bribe 50 nodes out of this or this hundred nodes. These hundred nodes selected for mining were themselves selected out of maybe hundreds of thousands or millions of nodes. They were selected because they represented a random sub-selection. If originally 20 percent of the mining power is adversarial when you do this kind of random choice roughly 20 percent of the nodes will be adversarial but what adversary is doing is acting after the selection has happened. Hence, these protocols are not secure against an afterword adversary, which was a unique strength across the proof of work protocols.

10.3 Prism

So far we discussed a bunch of protocols that designed specifically to solve the scaling problem, but they disrupted the core structure of bitcoin's longest chain protocol. What we're looking for is a native proof-of-work protocol that actually solves the latency network problem and remain secure under adaptive adversary just like bitcoin. Prism [2], is a new proof-of-work blockchain protocol, which can achieve security against up to 50 percent adversarial hashing power while achieving optimal throughput. In order to understand Prism, lets deconstruct Bitcoin into its sub-components, and then maybe we can put together these components correctly to get something with a better performance.

10.3.1 Deconstructing the Blockchain to Approach Physical Limits

We should get a core observation of bitcoin and to do that we should know what is the role of different blocks. The latency of bitcoin is coming form waiting for getting enough votes when we get enough voters we can confirm a block. An obvious way to solve this latency problem is to increase the rate of voting, if we make the voting faster then we could potentially confirm a block fast. But there is a problem, if we want to increase the rate of voting we have to increase the rate of proposing (rate of voting = rate of proposing) which will break the security due to excess forking. So, each block in bitcoin is performing a dual role of voting and proposing which is analogous to an election where every voter is also a candidate.

Lets consider the case where we separate the proposal blocks from voter blocks. With this schema we can accumulate votes faster. Voter chain doesn't contain any transaction, they just contain votes to a particular proposer block. Consider figure 10.4, there are 2 separate proposal blocks for second page of the ledger at level 2. And the voter chains are voting for these two blocks. Consider total number of voter chains being 1000 and based on these votes we can decide which block goes into the ledger. Figure 10.5 is deconstructing the basic blockchain structure into its atomic functionalities. The selection of the main chain in a blockchain protocol (e.g., the longest chain in Bitcoin) can be viewed as electing a leader block among all the blocks at each level of the blocktree, where the level of a block is defined as its distance (in number of blocks) from the genesis block. Blocks in a blockchain then serve three purposes:they stand for election to be leaders, they add transactions to the main chain, and they vote for ancestor blocks through parent link relationships. Here we explicitly separate these three functionalities by representing

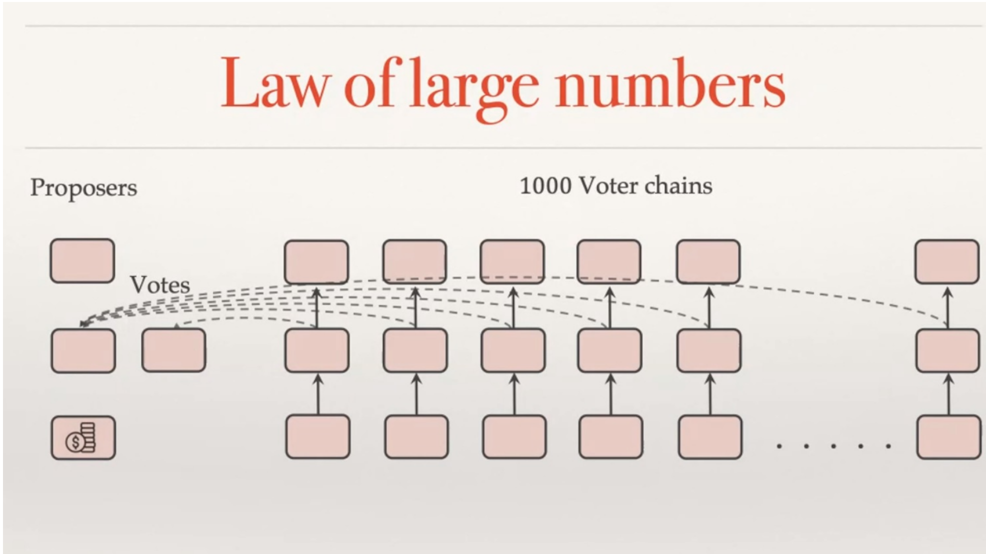

Figure 10.4: Deconstructing blockchain.

the blocktree in a conceptually equivalent form in Figure10.5 . In this representation, blocks are divided into three types: proposer blocks, transaction blocks and voter blocks. The voter blocks vote for transactions indirectly by voting for proposer blocks, which in turn link to transaction blocks. Proposer blocks are grouped according to their level in the original blocktree, and each voter block votes for a proposer blocks at a level to select a leader block among them. The elected leader blocks can then bring in the transactions to form the final ledger. The valid voter blocks are the ones in the longest chain of the voter tree, and these longest chains maintains the security of the whole system.

10.3.1.1 Scaling

This alternative representation of the traditional blockchain may seem more complex than the original blockchain representation but provides a natural path for scaling performance to approach physical limits. To increase the transaction throughput, one can simply increase the number of transaction blocks that a proposer block points to without compromising the security of the blockchain. This number is limited only by the physical capacity of the underlying communication network. To provide fast confirmation, one can increase the number of parallel voting trees, voting on the proposal blocks in parallel to increase the voting rate, until reaching the physical limit of confirming at a high latency and extremely high reliability. Note that even though the overall block generation rate has increased tremendously, the number of proposal blocks per

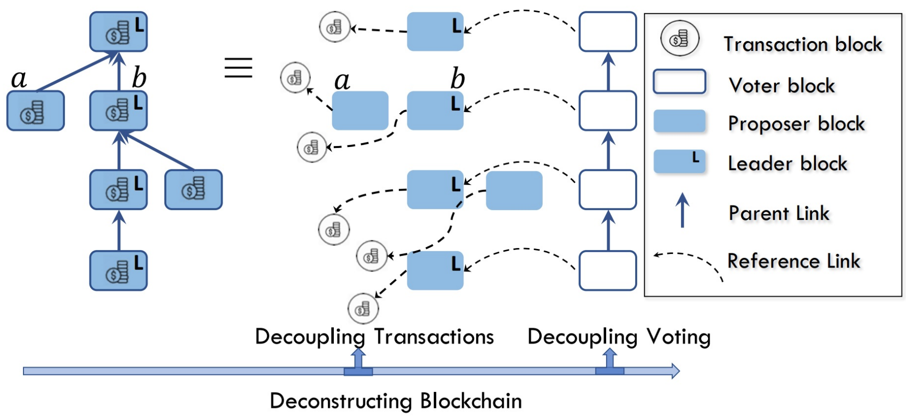

Figure 10.5: Deconstructing blockchain.

level remains small and manageable, and the voting blocks are organized into many separate voting chains with low block mining rate per chain and hence little forking.

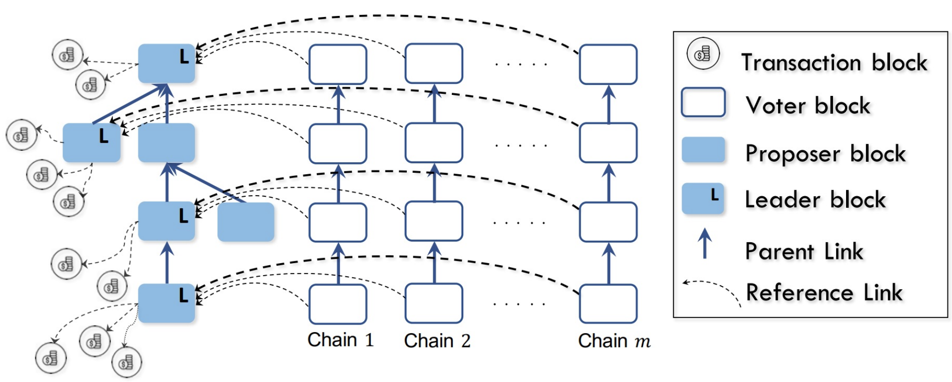

Figure 10.6: Prism

10.3.1.2 Sortition

There are three types of blocks: transaction, proposer, and voter blocks. Voter blocks are further sorted into blocks of different voting trees, this can be accomplished by using the random hash value when a block is successfully mined. This sortition splits the adversary power equally across the structures and does not allow it to focus its power to attack specific structures.

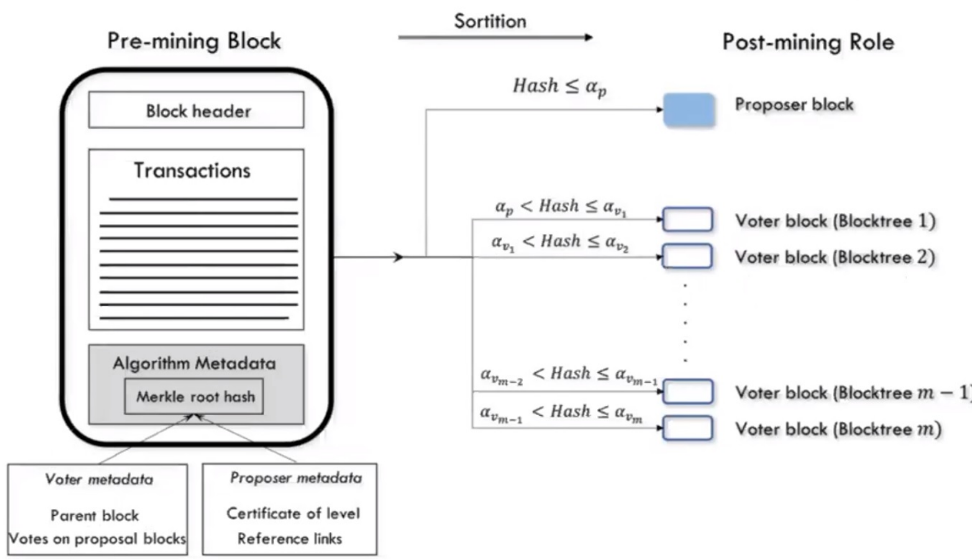

Figure 10.7: Prism sortition

10.3.1.3 How does mining happen?

When you mine a block, you do not know whether you're going to end up in a proposal chain or in a voter chain or in which voter chain. Not knowing this a-priori is a very important requirement and property of this protocol. When you create a pre-mining block you put in two kinds of data, one is the voting data containing the votes for proposal blocks and the corresponding parent blocks for each voter chains. And the other is proposer metadata which contains proof of level referencing a block from the the previous level of the proposer chain as shown in Figure 10.7. The Merkle root hash of the voter and proposer metadata serves as a commitment. The range of the output of the hash function is segregated into groups. If the hash function for the chosen nonce falls in the first range, then you get the propose a block and if it falls into the second range then you get to mine a block in the first voter chain and so on. When verifying a block, nodes will only look at that particular metadata content depending on the hash rate.

10.3.2 Confirmation Policy

Consider figure 10.4 that two blocks on the left have contradictory transactions. the confirmation policy is not as simple as count the number of votes right now and take a decision, we have to do something more. In this case if there are 1000 votes and one gets 100 and the other one gets 900 we can relatively be sure that the block with most votes will remain in the lead for future.

In case of bitcoin with say you get 80 percent reliability, if one block gets a little bit more than 500 votes, you will never confirm that block because it is likely that it's a block produced by an adversary. Now consider that all chains are ready to vote, and each block is stable except for 20 percent probability, then if the a block is legit then out of 1000 vote we expect it gets 800 or more. Instead of waiting for law of large numbers happening over time we have law of large numbers happening across the space, and because of this we will have much faster confirmation for this block. With one interval block arrival at a time you have actually confirmed a proposal block at very high reliability, so now we don't have to wait for a lot's of time to actually aggregate the voters. So we are essentially using weak protections but many chains in parallel and aggregating them and using the independence to say that we have gotten our self a strong net protections. In the longest-chain protocol, for fixed block size and network,the maximum tolerable adversarial hash power β is governed by the block production rate; the faster one produces blocks, the smaller the tolerable β. In Prism, we need to be able to tolerate β adversarial hash power in each of the voter trees and the proposer tree. Hence, following the observations of each of these trees individually must operate at the same rate as a single longest-chain blocktree in Bitcoin in order to be secure.The security of Prism is provided by the voter trees; a proposer block is confirmed by votes which are on the longest chains of these voter trees. Consider a conservative confirmation policy for Prism, where we wait for each vote on each voter tree to reach a confirmation reliability $1 - \epsilon$ before counting it. This would requires to wait for each vote to reach a depth of $k(\epsilon)$ in its respective tree,where $k(\epsilon)$ denotes the confirmation depth for reliability $1 - \epsilon$. This conservative confirmation rule immediately implies that Prism has the same security guarantee as that of each of the voter tree, i.e. that of Bitcoin.

10.3.2.1 Least Confirmation

There are various properties for Prism, mainly low latency confirmation but there are scenarios that we can not obtain low latency confirmation. If we have two blocks that both split roughly equally then we can not be sure that which one should remain in the ledger. So to be sure that we chose the right one we might need to vote until the last votes. We can't say which one will remain in the ledger but we know at least one of them will remain winner. As the blocks get dipper and dipper we will eventually be able to choose the winner.

10.3.2.2 Decoupled validity

A key aspect of Prism is that it has decoupled validating. When you propose a block you do not know which block is the possible ancestor. This is very different in bitcoin; when you propose a block in bitcoin it is already inside the chain and you know under which history this block is going to be passed. So we can not guaranty that all transactions will remain valid after the block got in. So in Prism we don't require that all transactions needs to be valid for the block to be valid. There is question about the schema of the blocks here, when we build a block if we want to float the block which is only going to be proposal block do we need to contain all the voter metadata? We can say that there is a subtlety here. The data structure that we use is Merkel tree and what we use for that Metadata is Merkel root and that allows you to reveal selected subset of data which comes from and belongs to particular root-hash, we can assert that without showing other data.

References

[1] Ittai Abraham, Dahlia Malkhi, Kartik Nayak, Ling Ren, and Alexander Spiegelman. Solida: A blockchain protocol based on reconfigurable byzantine consensus. *arXiv preprint arXiv:1612.02916*, 2016.

[2] Vivek Bagaria, Sreeram Kannan, David Tse, Giulia Fanti, and Pramod Viswanath. Prism: Deconstructing the blockchain to approach physical limits. In *Proceedings of the 2019 ACM SIGSAC Conference on Computer and Communications Security*, pages 585–602, 2019.

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

[4] Chenxing Li, Peilun Li, Dong Zhou, Wei Xu, Fan Long, and Andrew Yao. Scaling nakamoto consensus to thousands of transactions per second. *arXiv preprint arXiv:1805.03870*, 2018.

[5] Serguei Popov, Olivia Saa, and Paulo Finardi. Equilibria in the tangle. *Computers Industrial Engineering*, 136:160–172, Oct 2019.

[6] Yonatan Sompolinsky, Yoad Lewenberg, and Aviv Zohar. Spectre: A fast and scalable cryptocurrency protocol. *IACR Cryptol. ePrint Arch.*, 2016:1159, 2016.

[7] Yonatan Sompolinsky and Aviv Zohar. Phantom: A scalable blockdag protocol. *IACR Cryptol. ePrint Arch.*, 2018:104, 2018.