Improved scaling of the disk space taken by the Bitcoin blockchain

Benjamin Loison

june 2021

1 Summary

The concept of blockchains is recent in computer science, defined for its famous application in Bitcoin by S. Nakamoto in 2008. Bitcoin is a blockchain allowing exchanges of the virtual currency Bitcoin in a decentralized way, that is to say without intervention of states or banks. One of the major challenges for blockchains is scaling, that is, maintaining stability and easy interaction with the blockchain even if the number of users increases by an order of magnitude. In the case of Bitcoin, in order to verify transactions, users, also known as nodes, must check that every transaction on the network is correct. However, to do this, they must trace the source of the money throughout Bitcoin's 358 Gb history. This amount of linearly stored data in use of the blockchain, on the one hand, slows down the initialization of users (who then have to download the entire Bitcoin peer-to-peer network), and on the other hand, prevents ordinary users using, for example, their phones, from checking the network.
Our solution, being a special case of the article "Mining in Logarithmic Space" [4] dealing with blockchains in general, consists in storing only a verified current state of the monetary amount belonging to each user. This allows to reduce the bandwidth consumption and also the storage from 358 Gb to 4.3 Gb, although solutions for the storage already exist. Indeed a major problem for someone wanting to participate in the protocol is the initialization. In practice, initialization takes 10 days with a fiber connection since the download of the Bitcoin blockchain from the nodes is very slow, in addition to the 358 Gb that the initializing node must allocate to keep the Bitcoin blockchain. These two points discourage many enthusiasts, even though Bitcoin is intended to be a decentralized cryptocurrency secured by everyone's participation in the protocol.
In the particular case of Bitcoin, if we implement our approach while keeping the old protocol running, we notice that the current state of the new protocol can only be verified by the nodes running the new protocol. However, we can also notice that only the last verification is considered in our approach. Since this one is the result of the consensus of the majority of the nodes running the new protocol and the already initialized nodes switching to the new protocol can verify this hash independently, we can hope that our contribution guarantees the correctness of the data shared by the blockchain.
In this way, if such an approach were used to initialize the 10,000+ Bitcoin nodes, we could save over 3,500 Tb of bandwidth and storage. To allow new nodes to initialize using this fast and lightweight approach, we would need to propose a modification to Bitcoin Core implementing our approach and a modification to one of the software packages used to mine Bitcoin.

Table of contents

2 Introduction

The concept of blockchains, is a recent concept in computer science, first defined by its famous application in Bitcoin by S. Nakamoto in 2008 [1]. Bitcoin is a blockchain that allows exchanges of the virtual currency Bitcoin in a decentralized manner, that is, without the intervention of states or banks. When we talk about decentralization in computer networks, it means that they do not use the widespread master-slave model as when we connect to Google's servers with our computer for example, but rather a computer-to-computer model, also called peer-to-peer. It is interesting to note that the field of blockchains is one of the few areas where there is a significant advance of practice over theory, so Bitcoin was only proven safe under certain conditions in 2014 by Juan A. Garay [2]

One of the great difficulties for blockchains is scaling, that is, maintaining stability and easy interaction with the blockchain even if the number of users increases by an order of magnitude. In the case of Bitcoin, in order to verify monetary transactions, whether as a miner (user securing the network) or as a full node (user verifying the network), they must verify that every transaction on the network is legitimate and correct. However, to do this, they must trace the source of the money throughout Bitcoin's entire history, which currently weighs 358 Gb as of 2021 [3]. This amount of data stored linearly with the use of the blockchain, on the one hand slows down the initialization of miners and full nodes (which then have to download the entire blockchain from the Bitcoin peer-to-peer network), and on the other hand prevents ordinary users using, for example, their phones, from checking the network.

One idea might be to store only a current, also called snapshot, verified state of the monetary amount belonging to each user. So instead of going through the whole history of where the money came from, one can simply check the account balance. This makes it possible to get rid of the transaction history and thus most of the Bitcoin blockchain while keeping a very high level of security. My work is based on this ingenious idea from the article "Mining in Logarithmic Space" [4] which is however very general and which we will try to apply to the Bitcoin blockchain. Additional problems emerge since, for example, this article does not deal with the case of a fluctuating difficulty for miners to mine blocks, which is the case in Bitcoin. In practice, after digital application, the Bitcoin blockchain will be transformed from 358 Gb to 4.27 Gb, which allows a modern phone to easily check the Bitcoin network and allows a new user to initialize 84 times faster in the best case.

Indeed, until now, smartphones were based on the Simple Payment Verification (SPV) technique, which consists of relying on full nodes and having to wait for a certain number of confirmations in the blockchain to ensure that the payment made from the smartphone is taken into account by the network.

3 How Bitcoin Works

A blockchain is a storage technology that allows a single database to be distributed among different actors without the need for a trusted third party. Transactions, monetary in the case of Bitcoin, are issued to a few full nodes of the blockchain and signed by users wishing to make an electronic transfer. A node is any network actor participating in the consensus algorithm on the public register of monetary transactions that is Bitcoin.
In order to secure the network, the nodes transmit from one to another (knowing that a node is generally connected to about 10 other nodes) the transactions they receive, provided that they are legitimate and correct. That is to say, they must be issued by the depositary of the monetary funds involved and that these monetary funds exist in accordance with the history of previous transactions.
In order to converge the different users' databases due to the transactions propagating in the network, a level of difficulty is established by the history of the blockchain allowing everyone to try to solve a cryptographic problem of this difficulty. The user who solves this problem first can propagate the solution to the cryptographic problem and its associated database. The latter is of course verified by the other nodes receiving it. The nodes participating in this cryptographic problem are called miners and have a probability of success proportional to their computing power. The network is secure since it is assumed that the majority of the computing power is honest. It should also be noted that miners have an incentive to solve the cryptographic problem since they are rewarded with a fixed number of Bitcoin for success if they are the first. Moreover, if a group of users had more than half of the computing power of the network, they would have no incentive not to participate honestly, or else they would devalue their assets in Bitcoin.
The difficulty is determined so that on average the network solves the cryptographic problem every 10 minutes and it is at this average rate that the pending transactions are "validated". However, this validation can be questioned and we then speak of a block of transactions at a certain depth. If this depth is greater than 6, it is commonly considered that the probability of revoking this block is almost zero. Indeed, if two answers to the cryptographic problem posed are found at approximately the same time, then the two underlying transaction blocks may be different. In this case we name it : a fork. We must continue the consensus algorithm and not consider for the moment either of the two blocks but wait until one branch of blocks accumulates to be "significantly" longer than the other with, for example, 6 blocks in advance.
Lightweight devices such as phones that do not have the ability to store Bitcoin's history cannot therefore verify whether or not the payment they have made is actually being taken into account by the network. The current SPV method, which relies on full nodes transmitting the depth of the block in which their transactions are, is less secure than if these devices participated directly in the Bitcoin protocol.

4 Progress

4.1 The main idea of our approach

The challenge of my internship was to try to solve the problem of scaling, in terms of disk space and bandwidth, of a blockchain.
The idea of the article supporting the internship, "Mining in Logarithmic Space" [4], is to keep only the current state, also called UTXO set, and some blocks. These blocks are selected as being the most recent for each level of difficulty of the hashes. In the following when we talk about a hash, it means the hash of a block like for example :
00000000839a8e6886ab5951d76f411475428afc90947ee320161bbf18eb6048. The cryptographic pro-

blem mentioned above consists in finding a result by the hash function SHA-256[2] starting at least with a certain number of zeros (here the hash starts with 8 hexadecimal zeros). A hash function is a function that takes a sequence of bits of any size and outputs a sequence of l bits, 256 in the case of SHA-256, this sequence being called a hash. The idea of hash functions is to make a kind of "summary" of the data input to the function. These hash functions are designed to allow to trace back to an antecedent from an image only with the help of brute force. In this way, the miner makes many attempts to vary the antecedent a little and gives a random image by the hash function's definition. Thus everyone has a chance to mine a block that corresponds to his or her computing power. This proof of work is a way to establish a kind of democracy depending on the computing power of each person on the internet. The expression of the level of difficulty of the hashes consists in partitioning the hashes according to the number of zeros by which they begin exactly.
The proof of work that is the resolution of the cryptographic problem, as it is the case in Bitcoin, allows to apply the idea of this reference paper [4] and guarantees a very high security. Indeed the idea consists in putting the current state, also called UTXO set, in the blocks and in this way a node wishing to initialize only has to take the UTXO set of the 6th block starting from the end. Moreover this approach proves that the 6th last block is indeed the one it claims to be by noticing that some blocks have a rare hash compared to the others, i.e. by having a hash starting with n hexadecimal zeros, we deduce statistically speaking that there are 16 blocks whose hash starts with $n-1$ hexadecimal zeros and so on.

4.2 The advantages of our approach

I first studied the advantages of applying the "Mining in Logarithmic Space" [4] approach to the Bitcoin blockchain. They are done at 2 levels :

1. For miners and full nodes : since the initialization would be done by essentially receiving and storing the UTXO set of about 4.24 Gb of Bitcoin instead of the 358 Gb of the blockchain. Indeed, according to the table on page 9 [Figure 1], $\text{polylog}(n)c + k\delta$ is relatively negligible in front of a. Since n is 695 590 [5], c is up to 97 bytes, k is 6, δ is between 0 and 2 MB and a is 4.24 Gb.
2. For smartphones using SPV : these lightweight nodes can easily become full nodes since a weight of 4.24 Gb is bearable on a smartphone as opposed to the 358 Gb previously required. This way a smartphone can more quickly trust that its transaction is well registered in the Bitcoin blockchain and its participation as a full node also increases the security of the network. Mining with a smartphone is irrelevant.

4.3 The difficulties in applying our approach to Bitcoin

4.3.1 Presentation of the two problems

The problem of fluctuating difficulty

The first of the two major problems in the case of Bitcoin is to manage the evolution of the target T. T is a minimum difficulty, for example if T is such that a hash must start with n zeros then a hash starting with $n+1$ zeros is also above the difficulty T.

To return to the subject of the evolution of the target T, which is the case in Bitcoin every 2,016 blocks, in the paragraph "Variable difficulty" on page 31 [4], the authors remind us that the theory of their paper works on the condition of having a constant difficulty T. In the case of variable difficulty, in order to keep the optimal compression they present, they would have

to adapt their use of the proofs in the paper "The bitcoin backbone protocol : Analysis and applications" [2] to those in the paper "The bitcoin backbone protocol with chains of variable difficulty" [7].
Then, after reading this last paper and thinking about the problems related to changing the difficulty T, I realized that it was much simpler to take $T = T_0$ with T_0 the initial target of the blockchain and thus the lowest. This would not allow to have the optimal compression mentioned in the article but it would be enough. Moreover T cannot go below T_0 because the computational capacities of the computers are increasing : the computational requirement of the protocol remains constant and the number of participating members is increasing and thus the computational power of the blockchain network is increasing. Thus it is impossible to go below T_0 and there would be other problems related to such a collapse even in the case of the current Bitoin protocol.

Backwards compatibility

The second main issue is backward compatibility, so using nodes of the new protocol as well as the old one at the same time should not be problematic. This is called a velvet fork. For example, in the new protocol, one could identify oneself as having the new protocol to the node, and thus the use of new functionalities such as sending the Π proof, allowing rapid initialization, would be possible. In practice it would be necessary to add $\log n$ pointers in the blocks as in the theory. In this one the authors define a set of pointers called interlink set which consists in having the links of each block of level j to the last blocks of each level lower than j which precedes it chronologically [Figure 2].
However, I will realize later, with the specification of the binary format of the blocks, that adding data in these blocks is very complex or even impossible because the header and the content of each block can't contain additional data and the size of the header is fixed. I was initially thinking of doing a hard fork. A hard fork as opposed to a velvet fork consists in changing the protocol implying an impossibility of backward compatibility. This hard fork would have consisted in including in the block the current state, that would have allowed to send the block without the current state which it contains and thus, in this way, with the hash of the block, one could prove to a new node by presenting him a set of UTXO ("which is not in the chain of blocks") that this one is correct. However, in addition to the hard fork breaking the backward compatibility, this would make the hash of the blocks complicated for some components dedicated to it like the ASICs which would then take only about 1 MB as input to the hash function but several Gb because of the UTXO set. One could then include in the block only the hash of the UTXO set. However, some problems remain, such as the problem of numerous ASICs in the case of Bitcoin.

4.3.2 Only a hard fork to apply the theory ?

The theory seems to require a hard fork in the case of Bitcoin to deal with this second problem. However, I realized that for the implementation to be used in practice, it would have to be backwards compatible. So we should not change the format of the blocks (this way we can keep the mining power of Bitcoin). Then we have to rely on other assumptions we have, like that the majority of users are honest. We could then ask the whole network for the hash of the UTXO set and only ask for the UTXO set from one of the nodes that sent the majority hash (and of course check it and if it doesn't match the majority hash by the hash function we ask another user). In this way the ASICs work in the same way as before, except that their initialization can be modified. The big disadvantage of this seemingly relevant method is, apart from the network cost of asking everyone (which may still be reasonable compared to

downloading the whole blockchain, since indeed a TCP handshake (used in Bitcoin) is only on the order of KB and there are only about 10,000 discovered nodes), is the fact that, as long as few people give the hash of the UTXO set, the majority hash is not necessarily the real one in the network. However, as users switch to the new protocol, they check the hash of the new protocol against the data they got from the old one and can by their majority "revoke" the old hash of the UTXO set.
However, the scalability of network initialization is questionable for this last idea since even if in the best case scenario where we establish a connection with everyone directly (without having to discover them), our improvement would favor the enlargement of this same network and with a linear complexity in the number of new users, this approach would not be sustainable in the long run. However, a probabilistic approach can perhaps ensure a very high percentage of certainty by not contacting the whole network.
Moreover, the set of UTXO that we are trying to obtain is of the order of k blocks before the end of the chain (it can be strictly more than k blocks if it takes more than 10 minutes to get everyone's hash). The other nodes, helping with initialization, can quite easily store the order of k last blocks. Each of these nodes does not necessarily have to have the hash of every UTXO set associated with every block ready to be sent, but can easily determine it by backtracking with knowledge of the blocks following the requested one and a hash of the UTXO set of one of these following blocks. In the scenario of high demand (each block hash is requested more than once), hashing for each block its UTXO set and storing it is however faster.

There remains the problem of keeping for each block a pointer to the blocks that precede it temporally speaking for each level of difficulty. As said before we cannot integrate this set of $\log n$ pointers in the blocks, we could however generate it from the information already present in Bitcoin. Indeed, each block refers to the block that precedes it in the blockchain, for each block we can obtain its hash using the hash function and we can download in a safe way as the UTXO set all the headers of the blockchain. Thus we have all the information to regenerate this set of pointers that the support article [4] uses. Moreover we can notice that the size of all the headers is very small since they weigh only about 62 MB in total because there are about 700,000 blocks where each header can weigh only up to 97 bytes.

4.3.3 The chosen solution : the use of the coinbase

My supervisor then pointed out a more elegant solution. There is only one place where you can put a little extra data that is not needed in the old protocol : the coinbase. The coinbase is the transaction that in each block rewards the miner who mined it monetarily. In each transaction a script with a language defined by Bitcoin allows to proceed to verifications. Among the instructions of this language is OP_NOP which does nothing [9]. However, you can't store much data in it. Nevertheless, if we notice that contrary to the theory, it is not necessary to store all the pointers (as previously stated), we can see that the location is sufficient.

4.4 Pre-implementation statistics

At that time we were unsure of how exactly their implementation worked in practice. We then contacted by email the authors of the paper on which we based our work. However, we only received an answer after two months of waiting. This was not really a problem because the main idea of the paper that one can use the difficulty levels of the hashes to prove a work done was the basis from which I thought about how exactly to proceed with the implementation. Before moving on to any implementation, I did some statistics on the number of hashes starting with n hexadecimal zeros [Figure 3]. We often talk about hexadecimal or binary zeros starting

the hash to imagine the difficulty although in practice a difficulty where the hash starts with 010 is greater than a difficulty where the hash starts with 011, while the number of starting zeros does not vary. In relation to these statistics of the number of blocks per hexadecimal level, I thought I was getting a complete binary tree but in reality it seems logical that this is not the case. Indeed the majority of miners came later when the difficulty was high, around difficulty level 18, and since Bitcoin's difficulty has tended to increase continuously since the beginning, we get this distribution.
This was also an opportunity for me to convince myself that I knew how to make a program analyzing the Bitcoin blockchain that I downloaded with Bitcoin Core respecting the binary format available online. Afterwards I noticed that in practice the binary zeros of the hashes involve hexadecimal zeros every 4 bits and therefore the binary zeros were more accurate [Figure 4]

4.5 Results

After implementing and optimizing Algorithm 1 from the support article in C++, I ran it with Bitcoin blocks in two different ways :

1. by inputting the entire Bitcoin blockchain in one go

2. by giving as input an empty string and the first block of Bitcoin and then the resulting compressed string of this function concatenated with the second block of Bitcoin and so on over all blocks of Bitcoin.

I then noticed that theorem 3 of the article was verified in practice. It consists briefly in verifying that compressing each block and adding it to the compressed of the previous blocks is ultimately identical to compressing the whole blocks in one go. The veracity of the theorem is of crucial importance so that the full nodes of the old protocol and those of the new protocol have the same compressed chain after switching to the new protocol.

The compression of the entire Bitcoin blockchain results in a proof for fast initialization noted $\Pi = \pi\chi$ of 2,065 blocks which after individual measurement weighs 0.96 Gb. I made a vector representation of it [Figure 5]. We can notice that the rectangles at the beginning are uniform because the difficulty increases and they are the most recent $2 * m$ (with $m = 3 * k$) blocks for each level of difficulty. Then more recently we notice that the blocks of higher level of $\pi\chi$ appear, knowing that this highest level must contain at least $2 * m$ blocks. Focusing on the cloud of the most recent part of the graph (10/2017), we observe that the first blocks of this cloud are the rarest because of higher difficulty level ℓ : statistically they are older than the $2 * m$ most recent blocks of each intermediate level. Then since the blocks of level $\ell - i$ (with $i \leq \ell$) are simpler to mine and that in $\pi\chi$ only the most recent $2 * m$ appear, we observe in a general way in this cloud that the more time goes by, the more the difficulty level of the blocks decreases. In spite of this gradual descent of the level of difficulty of the blocks, we can observe the appearance of a block of higher difficulty level intercalating in this descent. Finally, at the very end, we observe the last k blocks of the blockchain which have been taken as is in accordance with algorithm 1. [Figure 6]
According to [10] we conclude that the theory put in practice allows to convert precisely the 354 Gb in 4.27 (4.24 + 0.03) Gb which makes a difference of 349.73 Gb. This amounts to dividing the size of the blockchain by a factor of 84.

Not counting the Python and C++ algorithms that preprocess the blocks to list them with the time they were generated, the Bitcoin Core files they are stored in and where in them and their associated hashes, my C++ implementation is 800 lines long. I don't think it's particularly interesting to include it in this report but it is however available on GitHub [11]. However, to talk about it briefly, I would point out that by means of macro the user can change the processing mode : compression of blocks one by one or compression of all at once. Also in my algorithms to distinguish blocks from each other I only used a reference to the location of the block in the Bitcoin Core file containing it.

4.6 The work that remains to be done to make this approach a reality

The goal is to adapt and integrate my code to a mining and full node software in order to give potential tools helping to run the Bitcoin protocol. We must then succeed in mining a block in the official blockchain of Bitcoin to integrate our modified coinbase transaction in order to initialize the new version of the protocol allowing the fast initialization presented by our approach. We should therefore try to contact a mining pool that could help us with this and contact the Bitcoin development team to let them know that an implementation in recognized software in the field of this approach has been made and that the theory supports it. An intermediate step would be to proceed in the same way as on the official Bitcoin blockchain but on the already existing Bitcoin test network. Indeed, such a network already exists and it does not require the power of a mining pool but only that of a processor to carry out our integration tests.

5 Evaluation of the technical contribution

The idea of reducing each user's bandwidth and hard drive usage by about 350 Gb is appealing, since at the scale of Bitcoin's 10,000+ nodes, this represents about 3,500 Tb. However, it is important to be clear about the advantages and disadvantages of this method from different points of view :

1. For light nodes becoming heavier nodes by uploading and storing 4.3 Gb does not increase their verification speed since they have to wait at least 6 blocks after the transaction they are studying to consider it permanently validated. However, the security of those switching to the new protocol not only increases the security of Bitcoin by acting as verification nodes, but also allows them not to have to depend on other nodes and this contributes to a gain in security.
2. For enthusiasts, this makes it easier to try to contribute to the Bitcoin protocol, since it requires much less data to be downloaded to initialize, which is time consuming on such a peer-to-peer network.
3. With our approach, each user no longer has a Bitcoin transaction history. This can be seen as a positive for reducing storage space and bandwidth, but it can make it less reliable, for a new node, to try to download this history anyway by running the old protocol since fewer users are running it (for a new node).
4. One of the most important negative points is that in case of a fork considered solved by some (because one of the branches has a 6 blocks lead), if the other branch manages to compete again with the other branch and win the fork, users who thought this was impossible will not be able to easily go back and use the other branch. It should be noted here that there have been a few forks in the Bitcoin blockchain, and theoretically, although statistically insignificant, an old branch that is no longer relevant could compete and win the fork with the branch that was in the majority until then. It would therefore

be necessary to record all the blocks of the alternative branches, or at least the changes to the UTXO set before the fork, but this improvement was beyond the scope of my work.

6 Meta-information

The state of the art of the blockchain domain as a basis for reflection to restrict to the case of Bitcoin

I have already had an introduction to the topics of cryptography and blockchain at the MathInFoly summer school in 2019. It was during this course that I learned the fundamentals and developed the desire to go deeper into these topics.

I did my entire internship remotely because of the current health crisis.
Before having chosen the final subject of my internship with my supervisor, she sent me four research articles in the field of blockchains, totaling 65 pages ([12], [13], [14], [15]). These allowed me to distinguish the different research works that can be done : making a general case of a particular case and vice versa, making incomplete demonstrations, having a new idea in a domain and seeing where it leads us, deepening already existing works...

In order to fully understand the stakes of blockchain, I read many Wikipedia articles covering a wide spectrum from blockchain tools to application cases. This reading took me a few days and was done before the beginning of the 6 weeks internship.
After a few articles the idea of reducing local storage and total transmission of the blockchain in order to facilitate the scaling of blockchains interested us particularly.

During my internship, my supervisor sent me 8 research papers totalling 390 pages and I took the initiative to read another one of more than 30 pages so that I could understand the field of blockchains and more precisely the state of the art of the studied theme ([4], [1], [16], [17], [2], [18], [19], [7], [20]). Reading these different articles occupied a third of the time of the training course since during these readings I managed to better define and find solutions to the problems of the application of "Mining in Logarithmic Space" [4] to the case of Bitcoin.

In parallel to my reflection on the theoretical approach and in order to make statistics and prepare the implementation I downloaded the entire Bitcoin blockchain thanks to the renowned software in the field Bitcoin Core. It was also the occasion to notice the difficulty of becoming a full node since indeed the download through the Bitcoin peer-to-peer network required 10 days on a fibered network, which still underlines the interest of reducing the amount of bandwidth and disk space necessary to initialize the Bitcoin protocol.
Two weeks into the course, my supervisor and I could not figure out exactly whether or not the snapshot of the current state noted a was in each block. We also did not understand what was being sent, only the block or the block with the snapshot, and what exactly was being hashed. We then sent an email to the authors of the article supporting the course to ask them for clarification. Waiting for their answer didn't put my work on hold and after two weeks I didn't think I would receive an answer. However, after the internship, two months after our request, one of the authors of the paper took the time to answer us. In practice, this did not really change our technical solution.

Programming

During the internship I programmed a little in Python, or C++ as soon as the task became heavy without efficient multithreading and precise memory management, in order to obtain statistics that are not easily found on the internet such as the distribution of the hashes of the blocks sorted by the number of hexadecimal or binary zeros or the respect or not of the chronological order of storage of the blocks, their sizes...
This programming to obtain these statistics took me little time compared to the last week of the internship, where after a group consultation with my supervisor and one of her colleagues doing practical work on blockchains, we decided that I was going to implement our approach in the particular case of Bitcoin that we had studied and adapted from the general case of the paper "Mining in Logarithmic Space" [4]. In particular, we had to implement Algorithm 1 of the paper and verify the veracity of Theorem 3 in practice. This was not an obvious task because my implementation had to be particularly efficient in compressing the 700,000 blocks of the Bitcoin blockchain in two different ways. Especially in the case of block-by-block compression, it is quite easy to notice the restriction of algorithmic optimization methods since the process calling the "Dissolve" function of Algorithm 1 [Figure 6] is iterative and it itself in its "for" loop line 7 is iterative and cannot be multi-threaded. Indeed the execution of a loop has repercussions on the following ones, the only possible optimization was to choose the data structures as well as possible by analyzing the repercussions of an iteration of the loop to the other one and of a call to the other one of this function "Dissolve". In particular, we had to understand precisely the binary format of the blocks. In the end, the algorithm took 24 hours to execute in the case of block-by-block compression with what was already compressed.

7 Appendices

FIGURE 1 – Excerpt from the table on page 9 of "Mining in Logarithmic Space" [4] (BTC means Bitcoin)

Proposal	Storage	Communication	Can mine?
BTC Full	$n(c+\delta)$	$n(c+\delta)$	yes
BTC SPV	nc	nc	no
Ethereum	$nc + k\delta + a$	$nc + k\delta + a$	yes
This work	$poly\log(n)c + k\delta + a$	$poly\log(n)c + k\delta + a$	yes

Table 1. A comparison of our results and previous work. n: the number of blocks in the chain; δ: size of transactions in a block; c: block header size; a: size of snapshot; k: common prefix parameter

FIGURE 2 – Set of pointers for "Mining in Logarithmic" Space [4] necessary for the proper execution of their approach

Fig. 2. The interlinked blockchain. Each superblock is drawn taller according to its level. A new block links to all previous blocks that have not been overshadowed by higher levels in the meantime.

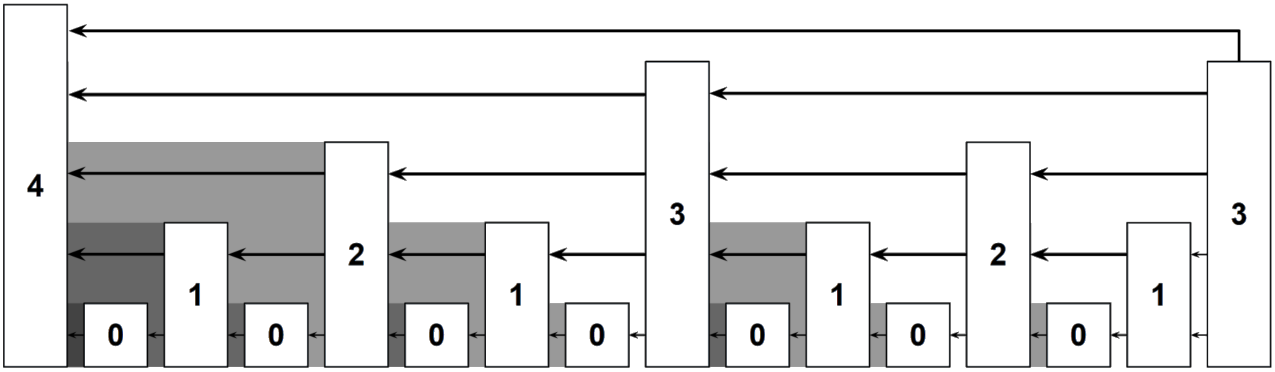

FIGURE 3 – Distribution of Bitcoin block hashes by difficulty m (n) where m is the number of hexadecimal zeros at the beginning of the hash and n is the number of hashes starting precisely with m hexadecimal zeros

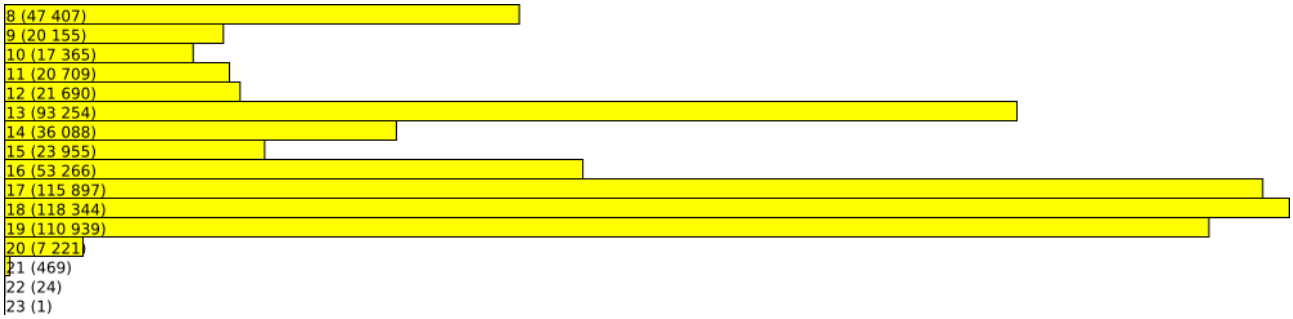

FIGURE 4 – Distribution of Bitcoin block hashes by difficulty m (n) where m is the number of binary zeros at the beginning of the hash and n is the number of hashes starting precisely with m binary zeros

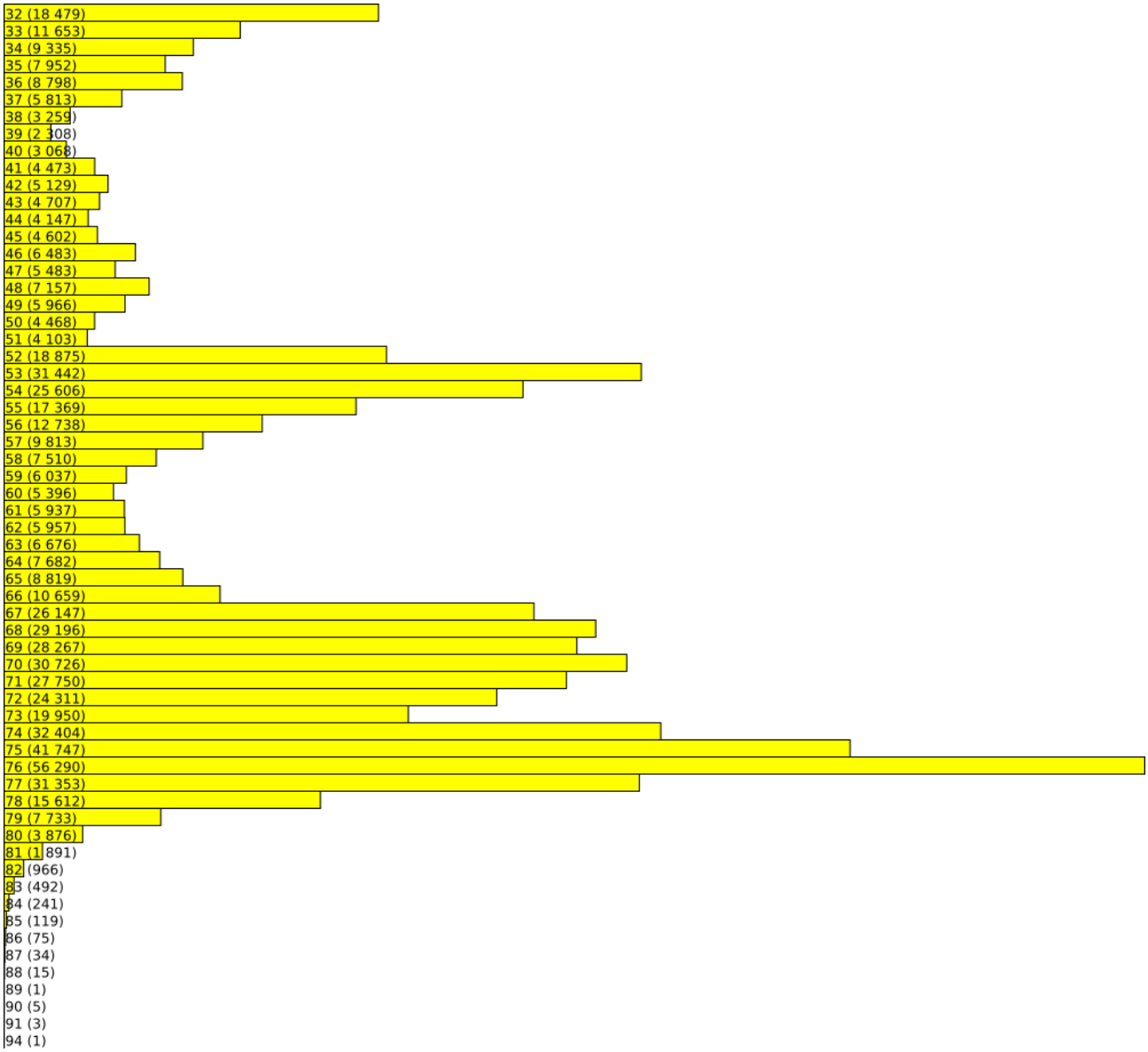

FIGURE 5 – Distribution of the hashes of $\pi\chi$, where each block has a width of 1 pixel

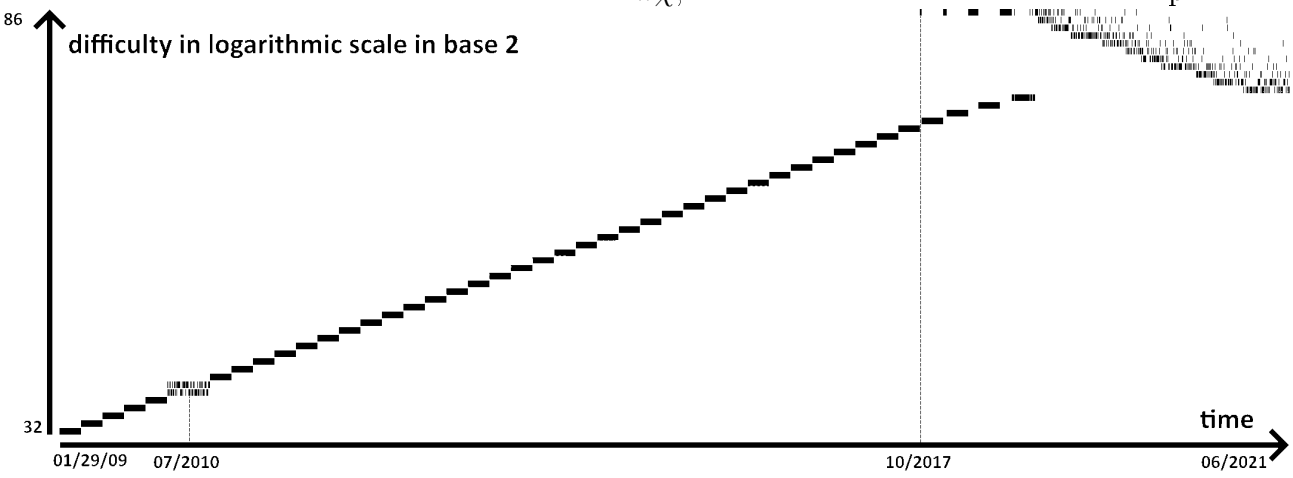

FIGURE 6 – Algorithm 1 of "Mining in Logarithmic Space" [4] allowing to compress a blockchain.

C is the chain of blocks
$C^* \uparrow^\mu$ denotes C^* blocks exactly of level of difficulty μ
$C^* \uparrow^\mu \{b:\}$ denotes $C^* \uparrow^\mu$ blocks more recent than the b block

Algorithm 1 Chain compression algorithm for transitioning a full miner to a logspace miner. Given a full chain, it compresses it into logspace state.

function $\mathsf{Dissolve}_{m,k}(\mathcal{C})$
  $\mathcal{C}^* \leftarrow \mathcal{C}[:-k]$
  $\mathcal{D} \leftarrow \emptyset$
  if $|\mathcal{C}^*| \geq 2m$ **then**
    $\ell \leftarrow \max\{\mu : |\mathcal{C}^*\uparrow^\mu \mid \geq 2m\}$
    $\mathcal{D}[\ell] \leftarrow \mathcal{C}^*\uparrow^\ell$
    for $\mu \leftarrow \ell - 1$ down to 0 **do**
      $b \leftarrow \mathcal{C}^*\uparrow^{\mu+1} [-m]$
      $\mathcal{D}[\mu] \leftarrow \mathcal{C}^*\uparrow^\mu [-2m:] \cup \mathcal{C}^*\uparrow^\mu \{b:\}$
    end for
  else
    $\mathcal{D}[0] \leftarrow \mathcal{C}^*$
  end if
  $\chi \leftarrow \mathcal{C}[-k:]$
  return $(\mathcal{D}, \ell, \chi)$
end function
function $\mathsf{Compress}_{m,k}(\mathcal{C})$
  $(\mathcal{D}, \ell, \chi) \leftarrow \mathsf{Dissolve}_{m,k}(\mathcal{C})$
  $\pi \leftarrow \bigcup_{\mu=0}^{\ell} \mathcal{D}[\mu]$
  return $\pi\chi$
end function

8 Sources

1. S. Nakamoto. Bitcoin : A peer-to-peer electronic cash system. 2008
2. J. Garay, A. Kiayias, N. Leonardos. The bitcoin backbone protocol : Analysis and applications (revised 2020)
3. https://www.blockchain.com/charts/blocks-size
4. Aggelos Kiayias, Nikos Leonardos and Dionysis Zindros. Mining in Logarithmic Space. 2021
5. https://www.blockchain.com/btc/blocks
6. https://en.bitcoin.it/wiki/Block
7. J. A. Garay, A. Kiayias, N. Leonardos. The bitcoin backbone protocol with chains of variable difficulty
8. https://bitnodes.io/
9. https://en.bitcoin.it/wiki/Script
10. https://statoshi.info/d/000000009/unspent-transaction-output-set?orgId=1&refresh=10m
11. https://github.com/Benjamin-Loison/Mining-in-Logarithmic-Space/blob/main/main.cpp

12. Geoffrey Saunois, Frédérique Robin, Emmanuelle Anceaume, Bruno Sericola. Permissionless Consensus based on Proof-of-Eligibility
13. Antoine Durand, Emmanuelle Anceaume, Romaric Ludinard. StakeCube : Combining Sharding and Proof-of-Stake to build Fork-free Secure Permissionless Distributed Ledgers
14. Krishnendu Chatterjee, Amir Kafshdar Goharshady, Arash Pourdamghani. Hybrid Mining
15. Marshall Ball, Alon Rosen, Manuel Sabin, Prashant Nalini Vasudevan. Proofs of Useful Work

16. Joseph Bonneau, Andrew Miller, Jeremy Clark, Arvind Narayanan, Joshua A. Kroll, Edward W. Felten. SoK : Research Perspectives and Challenges for Bitcoin and Cryptocurrencies
17. Emmanuelle Anceaume. Which Abstractions for the Blockchain Technology
18. Thomas Lajoie-Mazenc. Increasing the robustness of the Bitcoin crypto-system in presence of undesirable behaviours
19. Aggelos Kiayias, Andrew Miller, Dionysis Zindros. Non-Interactive Proofs of Proof-of-Work
20. Meni Rosenfeld. Predicting Block Halving Party Times

9 Acknowledgments

I am happy to have found an internship allowing me to deepen the field of blockchains. This thematic links cryptography (with mainly hash functions), network architecture and data architecture, as well as the verification of changes in its data. This modern field is particularly interesting because it allows secure, light and decentralized protocols to share data representing real facts.
I would particularly like to thank my supervisor Emmanuelle Anceaume from IRISA laboratory. She knew how to direct me to relevant articles describing the state of the art of the domain. She has always been attentive, available and willing to listen. I would also like to thank Romaric Ludinard who made himself available to give me some precious advices on the implementation part.