Amélioration du passage à l'échelle de l'espace pris par la blockchain Bitcoin

Benjamin Loison

juin 2021

1 Résumé

Le concept des blockchains est récent en informatique, défini pour sa célèbre application dans Bitcoin par S. Nakamoto en 2008. Bitcoin est une blockchain permettant des échanges de la monnaie virtuelle Bitcoin de manière décentralisée, c'est-à-dire sans intervention d'États ou de banques. Une des grandes difficultés pour les blockchains est le passage à l'échelle, c'est-à-dire le fait de maintenir la stabilité et une interaction aisée avec la blockchain même si le nombre d'utilisateurs augmente d'un ordre de grandeur. Dans le cas de Bitcoin, afin de vérifier les transactions, les utilisateurs, aussi appelés noeuds, doivent vérifier que chaque transaction du réseau est correcte. Cependant, pour faire cela, ils doivent retracer la provenance de l'argent dans tout l'historique de Bitcoin qui pèse 358 Go. Cette quantité de données stockées de manière linéaire en utilisation de la blockchain, d'une part ralentit l'initialisation des utilisateurs (qui doivent alors télécharger sur le réseau pair-à-pair Bitcoin l'intégralité de la blockchain), et d'autre part empêche les utilisateurs lambdas utilisant par exemple leur téléphone, de vérifier le réseau.
Notre solution, étant un cas particulier de l'article "Mining in Logarithmic Space" [4] traitant des blockchains de manière générale, consiste à stocker uniquement un état courant vérifié du montant monétaire appartenant à chaque utilisateur. Cela permet de réduire la consommation en bande passante et aussi en stockage de 358 Go à 4.3 Go, bien que des solutions pour le stockage existent déjà. Effectivement un problème majeur pour quelqu'un voulant participer au protocole est l'initialisation. En pratique l'initialisation dure 10 jours avec une connexion fibre puisque le téléchargement de la blockchain de Bitcoin à cause des noeuds est très lent, en plus des 358 Go que le noeud initialisant doit allouer pour conserver la blockchain de Bitcoin. Ces deux points découragent de nombreux amateurs, alors que Bitcoin se veut justement être une cryptomonnaie décentralisée sécurisée par la participation de tout le monde au protocole.
Dans le cas particulier de Bitcoin, si on implémente notre approche tout en conservant celle de l'ancien protocole en fonctionnement, on remarque que l'état courant du nouveau protocole ne peut être vérifié que par les noeuds exécutant le nouveau protocole. Cependant on pourra aussi remarquer que seule la dernière vérification est considérée dans notre approche. Puisque celle-ci est le fruit du consensus de la majorité des noeuds exécutant le nouveau protocole et que les noeuds déjà initialisés passant au nouveau protocole peuvent vérifier ce haché de manière indépendante, on peut espérer que notre contribution garantisse la correction des données partagées par la blockchain.
De cette manière si une telle approche était utilisée à l'initialisation des plus de 10 000 noeuds Bitcoin, on pourrait économiser plus de 3 500 To de bande passante et de stockage. Pour permettre aux nouveaux noeuds s'initialisant de bénéficier de cette initialisation rapide et légère, il faudrait proposer une modification de Bitcoin Core implémentant notre approche et une modification d'un des logiciels utilisé pour miner du Bitcoin.

Table des matières

2 Introduction

Le concept des blockchains, est un concept récent en informatique, défini la première fois par sa célèbre application dans Bitcoin par S. Nakamoto en 2008 [1]. Bitcoin est une blockchain permettant des échanges de la monnaie virtuelle Bitcoin de manière décentralisée, c'est-à-dire sans intervention d'États ou de banques. Lorsque l'on parle de décentralisation dans les réseaux informatiques, cela signifie que ceux-ci n'utilisent pas le modèle maître-esclave très répandu comme lorsque l'on se connecte sur les serveurs de Google avec notre ordinateur par exemple, mais plutôt un modèle de pair-à-pair, aussi appelé peer-to-peer, d'ordinateur à ordinateur. Il est intéressant de remarquer que le domaine des blockchains est un des rares domaines où l'on constate une avance importante de la pratique sur la théorie, ainsi Bitcoin n'a été démontré sûr sous certaines conditions qu'en 2014 par Juan A. Garay [2]
Une des grandes difficultés pour les blockchains est le passage à l'échelle, c'est-à-dire le fait de maintenir la stabilité et une interaction aisée avec la blockchain même si le nombre d'utilisateurs augmente d'un ordre de grandeur. Dans le cas de Bitcoin afin de vérifier les transactions monétaires, que ce soit en tant que mineur (utilisateur sécurisant le réseau) ou en tant que noeud complet (utilisateur vérifiant le réseau), ceux-ci doivent vérifier que chaque transaction du réseau est légitime et correcte. Cependant, pour faire cela, ils doivent retracer la provenance de l'argent dans tout l'historique de Bitcoin qui pèse 358 Go actuellement en 2021 [3]. Cette quantité de données stockées de manière linéaire en fonction de l'utilisation de la blockchain, d'une part ralentit l'initialisation des mineurs et noeuds complets (qui doivent alors télécharger sur le réseau pair-à-pair Bitcoin l'intégralité de la blockchain), et d'autre part empêche les utilisateurs lambdas utilisant par exemple leur téléphone, de vérifier le réseau.
Une idée peut alors être de stocker uniquement un état courant, aussi appelé snapshot, vérifié du montant monétaire appartenant à chaque utilisateur. Ainsi au lieu de parcourir toute l'histoire de la provenance de l'argent, on peut simplement vérifier le solde du compte. Cela permet notamment de se débarrasser de l'historique des transactions et donc de la majeure partie de la blockchain de Bitcoin tout en gardant un niveau de sécurité très élevé. Mes travaux se basent sur cette idée ingénieuse de l'article "Mining in Logarithmic Space" [4] qui est toutefois très généraliste et que l'on va essayer d'appliquer à la blockchain de Bitcoin. Des problématiques supplémentaires émergent puisque par exemple cet article ne traite pas le cas d'une difficulté fluctuante pour les mineurs à miner les blocs, ce qui est le cas dans Bitcoin. En pratique, après application numérique, on transformera la blockchain de Bitcoin de 358 Go à 4.27 Go, ce qui permet notamment alors à un téléphone moderne de vérifier aisément le réseau Bitcoin et permet dans le meilleur des cas à un nouvel utilisateur de s'initialiser 84 fois plus rapidement.
Effectivement en pratique jusqu'à maintenant les smartphones se basaient sur la technique du Simple Payment Verification (SPV) qui consiste à dépendre de noeuds complets et d'avoir à attendre un certain nombre de confirmations dans la blockchain pour s'assurer que le paiement effectué depuis son smartphone est bien pris en compte pour de bon par le réseau.

3 Fonctionnement de Bitcoin

Une blockchain, aussi appelée en français chaîne de blocs, est une technologie de stockage permettant de distribuer une même base de données entre différents acteurs sans besoin d'un tiers de confiance. Les transactions, monétaires dans le cas de Bitcoin, sont émises à quelques noeuds complets de la blockchain et signées par les utilisateurs désirant effectuer un virement électronique. On appelle noeud tout acteur du réseau participant à l'algorithme de consensus sur le registre public des transactions monétaires qu'est Bitcoin.
Afin de sécuriser le réseau, les noeuds transmettent de proche en proche (sachant qu'un noeud est en général connecté à environ 10 autres noeuds) les transactions qu'ils reçoivent, à condition qu'elles soient légitimes et correctes. C'est-à-dire qu'il faut qu'elles soient respectivement bien émises par le dépositaire des fonds monétaires en jeu et que ces fonds monétaires existent bien en accord avec l'historique des transactions précédentes.
Afin de faire converger les différentes bases de données des utilisateurs dues aux transactions se propageant dans le réseau, un niveau de difficulté est établie par l'historique de la blockchain permettant à chacun d'essayer de résoudre un problème cryptographique de cette difficulté. L'utilisateur résolvant ce problème le premier peut propager la solution au problème cryptographique et sa base de données associée. Cette dernière est bien évidemment vérifiée par les autres noeuds la recevant. Les noeuds participant à ce problème cryptographique sont appelés des mineurs et ont une probabilité de succès proportionnelle à leur puissance de calcul. Le réseau est sécurisé puisque l'on fait l'hypothèse que la majorité de la puissance de calcul est honnête. Il est également à noter que les mineurs sont incités à résoudre le problème cryptographique puisqu'une récompense d'un nombre fixé de Bitcoin leur est accordée en cas de succès s'ils sont les premiers. De plus si un groupe d'utilisateurs regroupait plus de la moitié de la puissance du calcul du réseau, ils n'auraient aucun intérêt à ne pas participer honnêtement sinon ils dévaloriseront leurs acquis en Bitcoin.
La difficulté est déterminée pour qu'en moyenne le réseau résolve le problème cryptographique toutes les 10 minutes et c'est à ce rythme moyen que les transactions en attente sont "validées". Cependant cette validation peut être remise en question et on parle alors de bloc de transactions à une certaine profondeur. Si celle-ci est supérieure à 6, il est communément jugé que la probabilité de révoquer ce bloc est quasi nulle. En effet si deux réponses au problème cryptographique posé sont trouvées à peu près en même temps, alors les deux blocs de transactions sous-jacents peuvent être différents. Dans ce cas on parle de fork. Il faut continuer l'algorithme de consensus et ne considérer pour le moment aucun des deux blocs mais attendre qu'une branche de blocs s'accumulant devienne "largement" plus longue que l'autre avec par exemple 6 blocs d'avance.
Les appareils légers tels que les téléphones n'ayant pas la capacité de stocker l'historique de Bitcoin ne peuvent donc pas vérifier si le paiement qu'ils auraient effectué est bien pris en compte ou non par le réseau. La méthode actuelle SPV qui se base sur des noeuds complets qui leur transmettent la profondeur du bloc dans lequel sont leurs transactions est moins sécurisée que si ces appareils participaient directement au protocole Bitcoin.

4 Déroulement

4.1 L'idée principale de notre approche

L'enjeu de mon stage était d'essayer de résoudre le problème du passage à l'échelle, du point de vue de l'espace disque et de la bande passante, d'une blockchain.
L'idée de l'article support du stage, "Mining in Logarithmic Space" [4], est de conserver uniquement l'état courant, aussi appelé set d'UTXO, et certains blocs. Ces blocs sont sélectionnés car étant les plus récents pour chaque niveau de difficulté des hachés. Dans la suite lorsqu'on

parle de haché, cela signifie le haché d'un bloc comme par exemple : 00000000839a8e6886ab5951d76f411475428afc90947ee320161bbf18eb6048. Le problème cryptographique mentionné ci-dessus consiste à trouver un résultat par la fonction de hachage SHA-256[2] commençant au minimum par un certain nombre de zéros (ici le haché commence par 8 zéros hexadécimaux). Une fonction de hachage est une fonction prenant une suite de bits de taille aussi grande que l'on veut et donnant en sortie une suite de l bits, 256 dans le cas de SHA-256, cette suite étant appelé un haché. L'idée des fonctions de hachage est par leur caractère aléatoire de faire une sorte de "résumé" des données mises en entrée de la fonction. Ces fonctions de hachage sont conçues pour ne permettre de remonter à un antécédent à partir d'une image qu'à l'aide de la force brute. De cette manière, le mineur fait de nombreuses tentatives pour faire un peu varier l'antécédent et donne une image aléatoire par la fonction de hachage par définition de celle-ci. Ainsi chacun a une chance de miner un bloc qui correspond à sa puissance calculatoire. Cette preuve de travail est un moyen d'établir une sorte de démocratie dépendant de la puissance calculatoire de chacun sur internet. L'expression du niveau de difficulté des hachés consiste à partitionner les hachés en fonction du nombre de zéros par lequels ils commencent exactement.
La preuve de travail qu'est la résolution du problème cryptographique, comme c'est le cas dans Bitcoin, permet d'appliquer l'idée de ce papier de référence [4] et garantit une très grande sécurité. En effet l'idée consiste à mettre l'état courant, aussi appelé set d'UTXO, dans les blocs et de cette manière un noeud souhaitant s'initialiser n'a qu'à prendre le set d'UTXO du 6ème bloc en partant de la fin. De plus cette approche prouve que le 6ème dernier bloc est bien celui qu'il prétend être en remarquant que certains blocs ont un haché rare comparé aux autres, c'est-à-dire qu'en ayant un haché commençant par n zéros hexadécimaux, on en déduit statistiquement parlant qu'il y a 16 blocs dont le haché commence par $n-1$ zéros hexadécimaux et ainsi de suite.

4.2 Les atouts de notre approche

J'ai étudié dans un premier temps les atouts qu'apporterait l'application de l'approche de "Mining in Logarithmic Space" [4] à la blockchain de Bitcoin. Ils se font à 2 niveaux :

1. Pour les mineurs et noeuds complets : puisque l'initialisation se ferait en recevant et stockant essentiellement le set d'UTXO d'environ 4.24 Go de Bitcoin au lieu des 358 Go de la blockchain. Effectivement d'après le tableau page 9 [Figure 1], $\text{polylog}(n)c + k\delta$ est relativement négligeable devant a. Puisque n vaut 695 590 [5], c vaut jusqu'à 97 octets, k vaut 6, δ vaut entre 0 et 2 Mo [6] et a vaut 4.24 Go.
2. Pour les smartphones utilisant SPV : ces noeuds légers peuvent facilement devenir des noeuds complets vu qu'un poids de 4.24 Go est supportable sur un smartphone contrairement aux 358 Go précédemment nécessaires. De cette manière un smartphone peut plus rapidement avoir confiance dans le fait que sa transaction est bien inscrite dans la blockchain de Bitcoin et sa participation en tant que noeud complet permet aussi d'augmenter la sécurité du réseau. Miner avec un smartphone étant hors de propos.

4.3 Les difficultés pour appliquer notre approche à Bitcoin

4.3.1 Présentation des deux problèmes

Le problème de fluctuation de la difficulté

La première des deux grandes problématiques dans le cas de Bitcoin est de gérer l'évolution de la target T. T est une difficulté minimale, par exemple si T est tel qu'un haché doit

commencer par n zéros alors un haché commençant par $n+1$ zéros est aussi en-dessous de la difficulté T.

Pour revenir au sujet de l'évolution de la target T, ce qui est le cas dans Bitcoin tous les 2 016 blocs, dans le paragraphe "Variable difficulty" page 31 [4], les auteurs rappellent que la théorie de leur papier fonctionne à condition d'avoir une difficulté T constante. Dans le cas d'une difficulté variable afin de garder la compression optimale qu'ils présentent, il faudrait adapter leur usage des preuves de l'article "The bitcoin backbone protocol : Analysis and applications" à celles de l'article "The bitcoin backbone protocol with chains of variable difficulty" [7].
Puis, après avoir lu ce dernier papier et réfléchi aux problématiques liées au changement de la difficulté T, j'ai réalisé qu'il était beaucoup plus simple de prendre $T = T_0$ avec T_0 la target initiale de la blockchain et donc la plus faible. Cela ne permettrait pas d'avoir la compression optimale mentionnée dans l'article mais elle serait suffisante. De plus T ne peut pas descendre en-dessous de T_0 car les capacités calculatoires des ordinateurs augmentent : l'exigence calculatoire du protocole reste constante et le nombre de membres participant est croissant et donc la puissance calculatoire du réseau de la blockchain ne fait qu'augmenter. Ainsi il est impossible de descendre en dessous de T_0 et il y aurait surtout d'autres problématiques liées à un tel affaissement même dans le cas du protocle actuel de Bitoin.

La rétro-compatibilité

La seconde principale problématique est la rétro-compatibilité, ainsi l'utilisation en même temps des noeuds du nouveau protocole comme de l'ancien ne doit pas être problématique. On parle alors de velvet fork. On pourrait par exemple dans le nouveau protocole s'identifier comme ayant le nouveau protocole auprès du noeud ainsi l'usage des nouvelles fonctionnalités comme l'envoi de la preuve Π, permettant l'initialisation rapide, serait possible. En pratique il faudrait rajouter les log n pointeurs dans les blocs comme dans la théorie. Dans celle-ci les auteurs définissent un ensemble de pointeurs appelé interlink set qui consiste à avoir les liens de chaque bloc de niveau j aux derniers blocs de chaque niveau inférieur à j qui le précède chronologiquement [Figure 2].
Cependant, je me rendrai compte par la suite, avec la spécification du format binaire des blocs, que rajouter de la donnée dans ceux-ci est très complexe voire impossible car l'en-tête et le contenu de chaque bloc ne peut pas contenir de donnée additionnelle et que la taille de l'en-tête est fixée. Je pensais initialement faire un hard fork. Un hard fork contrairement à un velvet fork consiste à changer le protocole impliquant une impossibilité de la rétro-compatibilité. Ce hard fork aurait consisté à inclure dans le bloc l'état courant, cela aurait notamment permis d'envoyer le bloc sans l'état courant qu'il contient et donc, de cette manière, avec le haché du bloc, on pourrait prouver à un nouveau noeud en lui présentant un set d'UTXO ("qui n'est pas dans la chaîne de blocs") que celui-ci est correct. Cependant en plus du hard fork cassant la rétro-compatibilité, cela rendrait le hachage des blocs compliqué pour certains composants qui y sont dédiés comme les ASICs qui ne prendraient alors plus qu'environ 1 Mo en entrée de la fonction de hachage mais bien plusieurs Go à cause du set d'UTXO. On pourrait alors inclure dans le bloc seulement le haché du set d'UTXO. Cependant certains problèmes demeurent comme par exemple le problème des ASICs nombreux dans le cas de Bitcoin.

4.3.2 Seul un hard fork pour appliquer la théorie ?

La théorie semble nécessiter un hard fork dans le cas de Bitcoin face à ce deuxième problème. Cependant je me suis rendu compte que pour que l'implémentation puisse être utilisée en pratique, elle devrait être rétro-compatible. Il ne fallait donc pas changer le format des blocs (de cette manière cela permet de garder la puissance de minage de Bitcoin). Il faut alors se

baser sur d'autres hypothèses que l'on a, comme celle selon laquelle la majoritée des utilisateurs est honnête. On pourrait alors demander à l'entièreté du réseau le haché du set d'UTXO et ne demander son set d'UTXO qu'à un des noeuds ayant renvoyé le haché majoritaire (et bien évidemment le vérifier et si celui-ci ne correspond pas par la fonction de hachage au haché majoritaire on demande à un autre utilisateur). De cette manière les ASICs fonctionnent de manière identique à précédemment, sauf que leur initialisation peut être modifiée. Le gros désavantage de cette méthode en apparence pertinente est, à part le coût du réseau de demander à tout le monde (ce qui peut encore être raisonnable comparé à télécharger toute la blockchain, puisque effectivement un handshake TCP (utilisé dans Bitcoin) ne pèse que de l'ordre du Ko et qu'il n'y a qu'environ 10 000 noeuds découverts [8]), est le fait que, tant que peu de personnes donnent le haché du set d'UTXO, le haché majoritaire n'est pas forcément celui réel du réseau. Cependant au fur et à mesure que les utilisateurs passent au nouveau protocole, ceux-ci vérifient le haché du nouveau protocole à l'aide des données qu'ils ont obtenues de l'ancien et peuvent par leur majorité "révoquer" l'ancien haché du set d'UTXO.
Cependant la scalabilité d'initialisation réseau est discutable pour cette dernière idée puisque même si dans le meilleur scénario où l'on établit une connexion avec chacun directement (sans avoir à les découvrir), notre amélioration favoriserait l'agrandissement de ce même réseau et avec une complexité linéaire en le nombre de nouveaux utilisateurs, cette approche ne serait pas soutenable au long terme. Toutefois une approche probabiliste peut peut-être assurer un très haut pourcentage de certitude en ne contactant pas l'ensemble du réseau.
De plus le set d'UTXO que l'on cherche à se procurer est de l'ordre de k blocs avant la fin de la chaîne (il peut être de strictement plus de k blocs si l'obtention du haché de tout le monde met plus de 10 minutes). Les autres noeuds, aidant à l'initialisation, peuvent assez facilement stocker de l'ordre de k derniers blocs. Chacun de ces noeuds n'a pas forcément à avoir, prêt à être envoyé, le haché de chaque set d'UTXO associé à chaque bloc mais peut toutefois le déterminer facilement en faisant des retours en arrière avec la connaissance des blocs suivants à celui demandé et un haché du set d'UTXO d'un de ces blocs suivants. Dans le scénario d'une forte demande (chaque haché de bloc est demandé plus d'une fois), hacher pour chaque bloc son set d'UTXO et le stocker est toutefois plus rapide.

Il reste le problème de conserver pour chaque bloc un pointeur vers les blocs qui le précèdent temporellement parlant pour chaque niveau de difficulté. Comme dit précédemment on ne peut pas intégrer cet ensemble de $\log n$ pointeurs dans les blocs, on pourrait cependant le générer à partir des informations déjà présentes dans Bitcoin. Effectivement chaque bloc fait référence au bloc qui le précède dans la chaîne de blocs, pour chaque bloc on peut obtenir son haché à l'aide de la fonction de hachage et on peut télécharger de manière sûre comme le set d'UTXO l'ensemble des headers de la chaîne de blocs. Ainsi on a toutes les informations pour regénérer cet ensemble de pointeurs que l'article support [4] utilise. De plus on peut remarquer que la taille de tous les en-têtes est très faible puisqu'ils ne pèsent qu'environ 62 Mo au total car il y a environ 700 000 blocs où chaque en-tête ne peut peser que jusqu'à 97 octets.

4.3.3 La solution retenue : l'utilisation de la coinbase

Mon encadrante m'a alors fait remarquer une solution plus élégante. Seul un endroit permet de mettre un peu de données supplémentaires inutiles à l'ancien protocole : la coinbase. La coinbase est la transaction qui dans chaque bloc permet de récompenser monétairement le mineur l'ayant miné. Dans chaque transaction un script avec un langage défini par Bitcoin permet de procéder à des vérifications. Parmi les instructions de ce langage figure OP_NOP qui ne fait rien [9]. Cependant on ne peut pas stocker beaucoup de données dans celle-ci.

Néanmoins en remarquant que contrairement à la thèse défendue dans la théorie, il n'est pas nécessaire de stocker l'ensemble des pointeurs (comme énoncé précédemment), on constate que l'emplacement est suffisant.

4.4 Statistiques avant implémentation

A ce moment là nous étions incertains de la manière dont fonctionnait précisément leur implémentation en pratique. Nous avons alors contacté par email les auteurs du papier sur lequel nous nous basions. Cependant nous n'avons reçu une réponse qu'après deux mois d'attente. Cela n'a pas été véritablement un problème car l'idée principale du papier selon laquelle on peut utiliser les niveaux de difficultés des hachés pour prouver un travail effectué était la base à partir de laquelle j'ai réfléchi pour déterminer précisément comment procéder à l'implémentation. Avant de passer à une quelconque implémentation, j'ai fait quelques statistiques sur le nombre de hachés commençant par n zéros hexadécimaux [Figure 3]. On parle souvent des zéros hexadécimaux ou binaires débutant le haché pour s'imaginer la difficulté bien qu'en pratique une difficulté où le haché débute par 010 est plus grande qu'une difficulté où le haché débute par 011, alors que le nombre de zéros débutants ne varie pas. Par rapport à ces statistiques du nombre de blocs par niveau hexadécimal, je pensais obtenir un arbre binaire complet mais en réalité, il semble logique que ce ne soit pas le cas. Effectivement la majorité des mineurs est arrivée plus tard lorsque la difficulté était grande, vers le niveau de difficulté 18, et comme la difficulté de Bitcoin a tendance à augmenter continuellement depuis le début, on obtient cette répartition.
Cela a été l'occasion pour moi aussi de me convaincre que je savais faire un programme analysant la blockchain Bitcoin que j'ai téléchargé avec Bitcoin Core en respectant le format binaire disponible en ligne. Après coup j'ai remarqué qu'en pratique les zéros binaires des hachés impliquent des zéros hexadécimaux tous les 4 bits et donc que les zéros binaires étaient plus précis [Figure 4]

4.5 Les résultats

Après avoir implémenté et optimisé l'algorithme 1 de l'article support en C++, je l'ai exécuté avec les blocs de Bitcoin de deux manières différentes :

1. en donnant d'un seul coup en entrée toute la chaîne des blocs de Bitcoin

2. en donnant en entrée une chaîne vide et le premier bloc de Bitcoin puis la chaîne compressée résultante de cette fonction concaténée avec le deuxième bloc de Bitcoin et ainsi de suite sur l'ensemble des blocs de Bitcoin.

J'ai alors constaté que le théorème 3 de l'article était vérifié en pratique. Il consiste brièvement à vérifier que compresser chaque bloc et l'ajouter au compressé des blocs précédents est au final identique à la compression en un seul coup de l'entièreté des blocs. La véracité du théorème est d'une importance cruciale pour que les noeuds complets de l'ancien protocole et ceux du nouveau aient la même chaîne compressée après passage au nouveau protocole.

La compression de toute la chaîne de blocs de Bitcoin résulte en une preuve pour l'initialisation rapide notée $\Pi = \pi\chi$ de 2 065 blocs qui après mesure individuelle pèse 0.96 Go. J'en ai fait une représentation vectorielle [Figure 5]. On peut remarquer que les rectangles au début sont uniformes car la difficulté augmente et ce sont les plus récents $2 * m$ (avec $m = 3 * k$) blocs pour chaque niveau de difficulté. Puis plus récemment on remarque que les blocs de plus

haut niveau de $\pi\chi$ apparaissent, sachant que ce plus haut niveau doit contenir au minimum $2*m$ blocs. En se concentrant sur le nuage de la partie la plus récente du graphique (10/2017), on observe que les premiers blocs de ce nuage sont les plus rares car de plus haut niveau de difficulté ℓ : statistiquement ils sont plus vieux que les $2*m$ plus récents blocs de chaque niveau intermédiaire. Puis puisque les blocs de niveau $\ell - i$ (avec $i \leq \ell$) sont plus simples à miner et que dans $\pi\chi$ seuls les $2*m$ plus récents y figurent, on observe de façon générale dans ce nuage que plus le temps avance, plus le niveau de difficulté des blocs diminue. Malgré cette descente graduelle du niveau des blocs, on constate ponctuellement l'apparition d'un bloc de niveau de difficulté supérieure s'intercalant dans cette descente. Finalement à la toute fin on observe les k derniers blocs de la blockchain qui ont été pris tels quels en accord avec l'algorithme 1. [Figure 6]
D'après [10] on en conclut que la théorie mise en pratique permet de convertir précisément les 354 Go en 4.27 (4.24 + 0.03) Go ce qui fait une différence de 349.73 Go. Cela revient à diviser la taille de la blockchain par un facteur 84.

Sans compter les algorithmes Python et C++ qui prétraitent les blocs pour les lister avec l'heure à laquelle ils ont été générés, les fichiers de Bitcoin Core dans lequels ils sont stockés et où dans ceux-ci et leurs hachés associés, mon implémentation en C++ fait 800 lignes. Il ne me semble pas particulièrement intéressant de l'inclure dans ce rapport mais elle est cependant disponible sur GitHub [11]. Toutefois pour en parler brièvement, je ferais remarquer que par le biais de macro l'utilisateur peut changer le mode de traitement : compression des blocs un par un ou compression de tous d'un seul coup. De plus dans mes algorithmes pour distinguer les blocs les uns des autres j'ai seulement utilisé une référence à l'emplacement du bloc dans le fichier Bitcoin Core le contenant.

4.6 Le travail restant à faire pour concrétiser cette approche

L'objectif est d'adapter et d'intégrer mon code à un logiciel de minage et de noeud complet afin de donner de potentiels outils aidant à l'exécution du protocole Bitcoin. Il faut alors en plus réussir à miner un bloc dans la blockchain officielle de Bitcoin pour intégrer notre transaction coinbase modifiée afin d'initialiser la nouvelle version du protocole permettant l'initialisation rapide présentée par notre approche. Il faut donc essayer de contacter une mining pool qui pourrait nous y aider et contacter l'équipe de développement de Bitcoin pour leur faire savoir qu'une implémentation dans les logiciels reconnus dans le domaine de cette approche a été faite et que la théorie la supporte. Une étape intermédiaire consisterait à procéder de la même manière que sur la blockchain officielle de Bitcoin mais sur le réseau de test déjà existant de Bitcoin. Effectivement un tel réseau existe déjà et il ne nécessite pas la puissance d'une mining pool mais seulement celle d'un processeur pour effectuer nos tests d'intégrations.

5 Evaluation de la contribution technique

L'idée de réduire pour chaque utilisateur l'usage de la bande passante et de son disque dur d'environ 350 Go est séduisante puisqu'à l'échelle des plus de 10 000 noeuds de Bitcoin, cela représente environ 3 500 To. Cependant il faut être clair sur les avantages et désavantages de cette méthode à différents points de vue :

1. Pour les noeuds légers devenir des noeuds un peu plus lourds en téléchargeant et stockant 4.3 Go n'augmente pas leur rapidité de vérification puisqu'il faut dans tous les cas attendre au moins 6 blocs après celui intégrant la transaction étudiée pour la considérer

comme validée de façon permanente. Toutefois la sécurité de ceux passant au nouveau protocole permet d'une part d'augmenter la sécurité de Bitcoin en agissant comme des noeuds de vérification mais leur permet aussi de ne pas avoir à dépendre d'autres noeuds et cela contribue à un gain de sécurité.

2. Pour les amateurs, cela permet donc d'essayer de contribuer au protocole Bitcoin plus facilement puisqu'il faut télécharger beaucoup moins de données pour s'initialiser ce qui est long sur un tel réseau pair-à-pair.
3. Avec notre approche, chaque utilisateur n'a plus l'historique des transactions Bitcoin. Cela peut être vu comme un point positif pour la réduction de l'espace de stockage et de la bande passante mais cela peut rendre moins fiable le fait d'essayer de malgré tout télécharger cet historique en exécutant l'ancien protocole puisque moins d'utilisateurs font tourner celui-ci.
4. Un des points négatifs les plus importants est qu'en cas de fork considéré résolu par certains (car une des branches a une avance de 6 blocs), si l'autre branche arrive à reconcurrencer le fork et le gagner, les utilisateurs qui pensaient cette situation impossible ne pourront pas aisément revenir en arrière pour emprunter l'autre embranchement. Il est à remarquer ici qu'il y a eu quelques forks dans la blockchain de Bitcoin et théoriquement, bien que statistiquement négligeable, une ancienne branche n'étant plus d'actualité pourrait concurrencer et gagner le fork avec la branche jusque là majoritaire. Il faudrait donc enregistrer tous les blocs des branches alternatives ou du moins les modifications par rapport au set d'UTXO avant l'embranchement cependant cette amélioration dépassait le cadre de mon travail.

6 Méta-information

L'état de l'art du domaine de la blockchain comme base de réflexion pour restreindre au cas du Bitcoin

J'ai déjà eu une initiation aux thématiques de la cryptographie et de la blockchain lors de l'école d'été MathInFoly en 2019. C'est lors de ce stage que j'en ai appris les fondamentaux et que j'ai développé l'envie d'approfondir ces sujets.

J'ai effectué l'ensemble de mon stage à distance à cause de la crise sanitaire actuelle.
Avant d'avoir choisi de manière définitive le sujet de stage avec mon encadrante, celle-ci m'a transmis quatre articles de recherche dans le domaine des blockchains, totalisant 65 pages ([12], [13], [14], [15]). Ceux-ci m'ont permis de distinguer les différentes travaux de recherche qui peuvent être effectués : faire un cas général d'un cas particulier et inversement, faire des démonstrations incomplètes, avoir une idée nouvelle dans un domaine et voir où cela nous mène, approfondir des travaux déjà existants...

Afin de bien comprendre les enjeux de la blockchain, j'ai lu de nombreux articles Wikipedia balayant un large spectre allant des outils de la blockchain aux cas d'applications. Cette lecture m'a pris quelques jours et a été effectuée avant le début du stage de 6 semaines.
Après quelques articles l'idée de réduire le stockage local et la transmission totale de la blockchain afin de faciliter le passage à l'échelle des blockchains nous intéressait particulièrement.

Durant mon stage mon encadrante m'a transmis 8 papiers de recherche totalisant 390 pages et j'ai pris l'initiative d'en lire un autre de plus de 30 pages afin que je comprenne bien ce domaine que constituent les blockchains et plus précisément l'état de l'art de la thématique

étudiée ([4], [1], [16], [17], [2], [18], [19], [7], [20]). Lire ces différents articles a occupé un tiers du temps du stage puisque lors de ces lectures j'arrivais à mieux cerner et trouver des solutions aux problèmes de l'application de "Mining in Logarithmic Space" [4] au cas du Bitcoin.

Parallèlement à ma réflexion sur l'approche théorique et afin de réaliser des statistiques et préparer l'implémentation j'ai téléchargé l'entièreté de la blockchain de Bitcoin grâce au logiciel renommé dans le domaine Bitcoin Core. Ce fut aussi l'occasion de remarquer la difficulté de devenir un noeud complet puisqu'effectivement le téléchargement à travers le réseau pair-à-pair de Bitcoin a nécessité 10 jours sur un réseau fibré, ce qui souligne encore l'intérêt de réduire la quantité de bande passante et d'espace disque nécessaire à l'initialisation du protocole Bitcoin. Deux semaines après le début du stage, mon encadrante et moi n'arrivions pas à comprendre précisément si oui ou non la snapshot de l'état courant noté a était dans chaque bloc. Nous ne comprenions pas non plus la nature de ce que l'on envoyait, seulement le bloc ou le bloc accompagné de la snapshot, et ce que l'on hachait précisément. Nous avons alors envoyé un email aux auteurs de l'article support du stage pour leur demander des précisions. L'attente de leur réponse n'a pas mis en pause mes travaux et au bout de deux semaines je ne pensais plus recevoir de réponse. Toutefois après le stage, deux mois après notre demande, un des auteurs du papier a pris le temps de nous répondre. Cela n'a en pratique pas vraiment changé notre solution technique.

La programmation

Lors du stage j'ai un peu programmé en Python, ou C++ dès que la tâche devenait lourde sans multithreading efficace et gestion précise de la mémoire, afin d'obtenir des statistiques que l'on ne trouve pas facilement sur internet comme la répartition des hachés des blocs triés par le nombre de zéros hexadécimaux ou binaires ou le respect ou non en pratique de l'ordre chronologique de stockage des blocs, leurs tailles...
Cette programmation pour obtenir ces statistiques m'a pris peu de temps comparée à la dernière semaine du stage, où après consultation groupée de mon encadrante et d'un de ses collègues effectuant des travaux pratiques sur la thématique des blockchains, nous avons décidé que j'allais implémenter notre approche dans le cas particulier de Bitcoin que l'on avait étudié et adapté du cas général du papier "Mining in Logarithmic Space" [4]. Plus particulièrement il fallait principalement implémenter l'algorithme 1 du papier et vérifier la véracité en pratique du théorème 3. Ce n'était pas une tâche évidente car mon implémentation devait être particulièrement efficace pour compresser les 700 000 blocs de la blockchain de Bitcoin de deux manières différentes. En particulier dans le cas de la compression de bloc par bloc, il est assez facile de remarquer la restriction des méthodes d'optimisation algorithmique puisque le processus appelant la fonction "Dissolve" de l'algorithme 1 [Figure 6] est itératif et que lui même dans sa boucle "pour" ligne 7 est itératif et ne peut pas être multithreadé. En effet l'exécution d'un tour de boucle a des répercussions sur les suivants, la seule optimisation possible était de choisir au mieux les structures de données en analysant les répercussions d'un passage de boucle à l'autre et d'un appel à l'autre de cette fonction "Dissolve". Il a notamment fallu comprendre précisément le format binaire des blocs. Au final l'algorithme mettait 24 heures à s'exécuter dans le cas de la compression bloc par bloc avec ce qui était déjà compressé.

7 Annexes

FIGURE 1 – Extrait du tableau page 9 de "Mining in Logarithmic Space" [4] (BTC signifiant 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 – Ensemble de pointeurs de "Mining in Logarithmic Space" [4] nécessaire à la bonne exécution de leur approche

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.

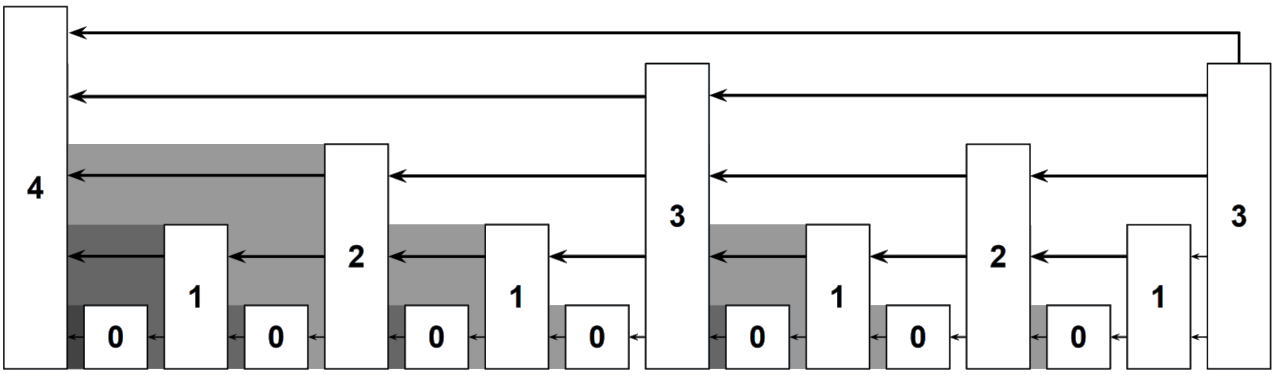

FIGURE 3 – Répartition des hachés des blocs de Bitcoin par difficulté m (n) où m est le nombre de zéros hexadécimaux au début du haché et n le nombre de hachés débutant précisément par m zéros hexadécimaux

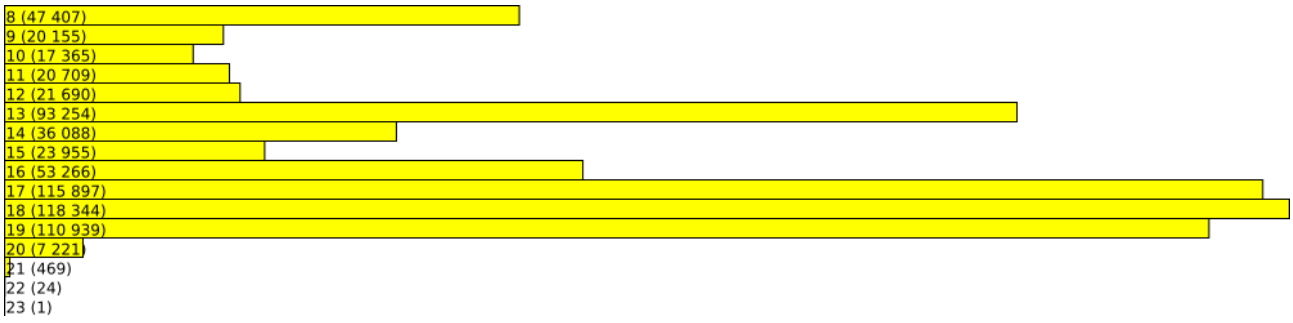

FIGURE 4 – Répartition des hachés des blocs de Bitcoin par difficulté m (n) où m est le nombre de zéros binaires au début du haché et n le nombre de hachés débutant précisément par m zéros binaires

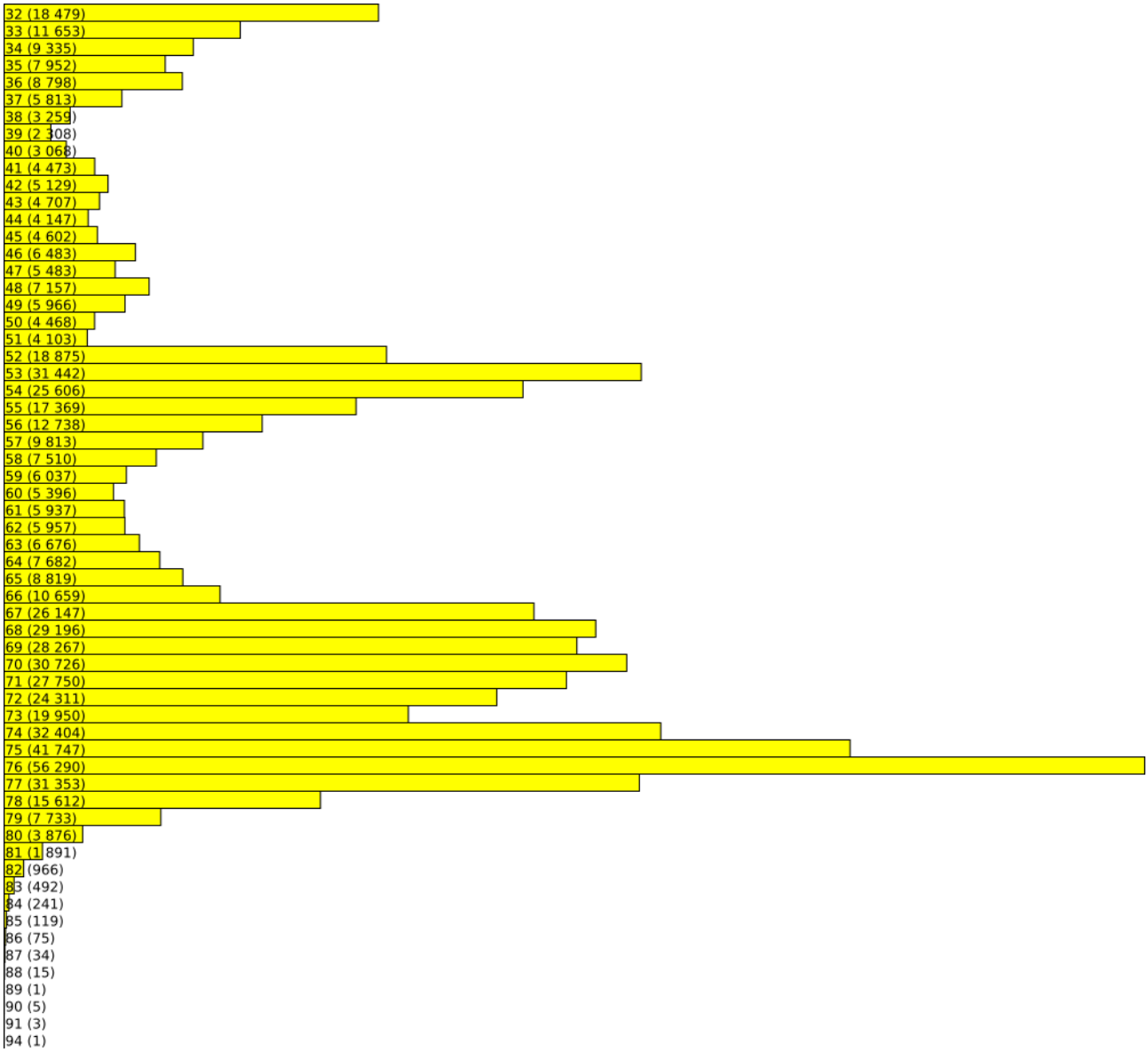

FIGURE 5 – Répartition des hachés de $\pi\chi$, où chaque bloc a une largeur de 1 pixel

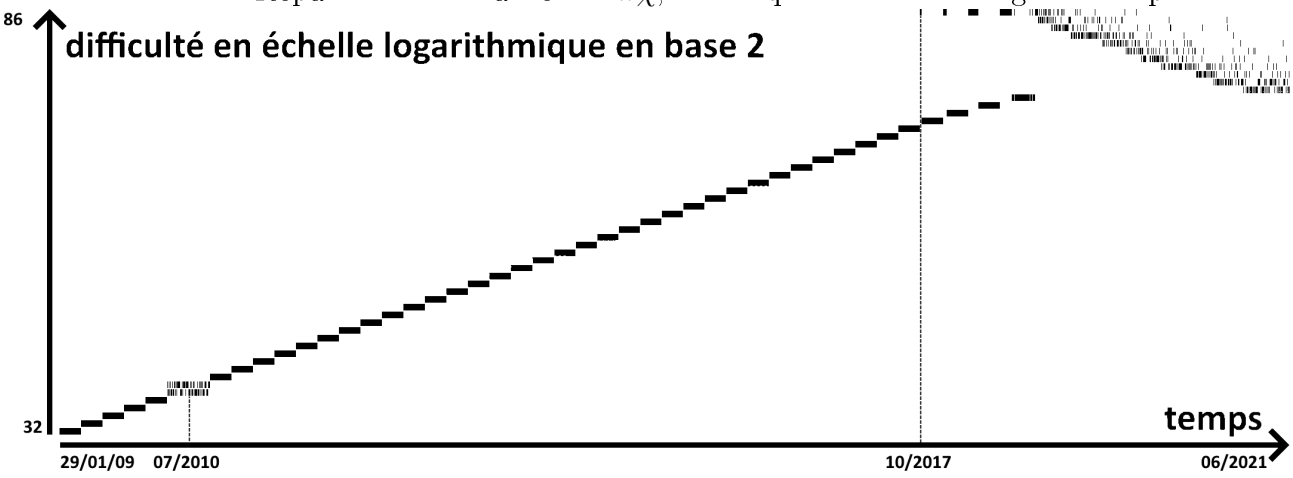

FIGURE 6 – Algorithme 1 de "Mining in Logarithmic Space" [4] permettant de compresser une blockchain.

C est la chaîne de blocs
$C^* \uparrow^\mu$ désigne les blocs de niveau de difficulté exactement μ de C^*
$C^* \uparrow^\mu \{b:\}$ désigne les blocs de $C^* \uparrow^\mu$ plus récents que le bloc b

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 Remerciements

Je suis heureux d'avoir trouvé un stage me permettant d'appronfondir le domaine des blockchains. Cette thématique lie la cryptographie (avec principalement les fonctions de hachage), l'architecture en réseau et l'architecture des données, ainsi que la vérification des changements de ses données. Ce domaine moderne est particulièrement intéressant car il permet d'obtenir des protocoles sûrs, légers et décentralisés pour partager des données représentant des faits réels.
Je tiens particulièrement à remercier mon encadrante de stage Emmanuelle Anceaume du laboratoire IRISA. Elle a su me diriger vers des articles pertinents décrivant l'état de l'art du domaine. Elle s'est toujours montrée attentive, disponible et à l'écoute. J'aimerais aussi remercier Romaric Ludinard qui s'est rendu disponible pour me donner quelques précieux conseils sur la partie implémentation.