



Département des Technologie de l'information et de la
communication (TIC)
Filière Télécommunications
Orientation Sécurité de l'information

Travail de Bachelor

Création d'une cryptomonnaie écologique et confidentielle

Étudiant

Enseignant responsable

Année académique

Gil Balsiger

Prof. Alexandre Duc

2020-2021

Yverdon-les-Bains, le 20 juillet 2021

Département des Technologie de l'information et de la communication (TIC)
Filière Télécommunications
Orientation Sécurité de l'information
Étudiant : Gil Balsiger
Enseignant responsable : Prof. Alexandre Duc

Travail de Bachelor 2020-2021

Création d'une cryptomonnaie écologique et confidentielle

Résumé publiable

Dans ce travail... Ceci est le résumé publiable...

Étudiant :	Date et lieu :	Signature :
Gil Balsiger
Enseignant responsable :	Date et lieu :	Signature :
Prof. Alexandre Duc

Préambule

Ce travail de Bachelor (ci-après TB) est réalisé en fin de cursus d'études, en vue de l'obtention du titre de Bachelor of Science HES-SO en Ingénierie.

En tant que travail académique, son contenu, sans préjuger de sa valeur, n'engage ni la responsabilité de l'auteur, ni celles du jury du travail de Bachelor et de l'Ecole.

Toute utilisation, même partielle, de ce TB doit être faite dans le respect du droit d'auteur.

HEIG-VD

Vincent Peiris
Chef de département TIC

Yverdon-les-Bains, le 20 juillet 2021

PRÉAMBULE _____

Authentification

Le soussigné, Gil Balsiger, atteste par la présente avoir réalisé ce travail et n'avoir utilisé aucune autre source que celles expressément mentionnées.

Yverdon-les-bains, le 20 juillet 2021

Gil Balsiger

AUTHENTICATION _____

Cahier des charges

Problématique

Contexte

Le Bitcoin est l'une des cryptomonnaies les plus connues et une des plus utilisées aujourd'hui en 2021. Cependant, elle n'est pas parfaite et certains points sur son fonctionnement posent problème. Un des points importants est la vérification des transactions, basée sur l'algorithme de preuves de travail (Proof of Work), est très consommatrice d'énergie. A titre d'exemple, la puissance totale de tous les mineurs de Bitcoin regroupés ensemble permettrait d'alimenter un pays de taille comparable aux Pays-Bas [**BTC_cons**]. Un autre problème est que les transactions sont consultables publiquement ce qui pose un problème de confidentialité. Il est possible de voir le montant de chaque transaction et ainsi parcourir la blockchain pour trouver le solde d'un compte. Cela n'est pas adapté à des transactions plus sensibles comme des versements de salaire par exemple. De plus, les performances du Bitcoin en terme de transactions par secondes sont très faibles comparé, par exemple, au réseau Visa. Le réseau Bitcoin permet 4 à 7 transactions par secondes contre plus de 2'000 pour Visa.

Solutions existantes

Il existe une multitude de blockchains et cryptomonnaies. Cependant, beaucoup d'entre elles utilisent le même principe énergivore que Bitcoin ou sont des tokens sur la blockchain Ethereum qui n'est, en 2021, ni plus écologique ni plus confidentielle que Bitcoin (Ethereum est cependant en train d'effectuer une migration vers un algorithme de preuve d'enjeu).

Mais il y a tout de même des solutions existantes. Il existe des algorithmes beaucoup moins consommateurs d'énergie que la vérification par preuve de travail (Proof of Work) utilisée par Bitcoin comme la preuve d'enjeu (Proof of Stake) ou encore la preuve d'espace (Proof of Space). Concernant les problèmes de confidentialités, il existe des blockchains confidentielles comme Monero ou Zcash. Cependant, il n'y a pas de blockchain qui utilise un algorithme écologique et confidentiel à la fois.

Objectif principal

L'objectif principal dans le but de résoudre cette problématique est d'analyser les protocoles existant afin de trouver un protocole plus écologique que la preuve de travail et de développer une blockchain qui utilisera ce protocole de consensus pour vérifier les transactions. Le but est également de comprendre le fonctionnement des cryptomonnaies au travers du développement de l'une d'entre elles.

Cahier des charges

Objectifs

Travail théorique

- État de l'art des protocoles de consensus avec un poids particulier sur le Proof of Space
- État de l'art des applications de preuves à divulgation nulle de connaissance aux blockchains
- Explication du fonctionnement d'une cryptomonnaie en général

Travail pratique

L'objectif de ce travail pratique est l'implémentation d'une cryptomonnaie. Dans un premier temps, l'objectif est d'implémenter un algorithme de consensus par Proof of Space (ou Proof of Space-time) et de réaliser une blockchain utilisant cet algorithme. Dans un second temps, il sera question d'y intégrer un mécanisme de sécurisation de la blockchain au moyen preuves à divulgation nulle de connaissance (zero-knowledge proofs) afin de rendre les transactions confidentielles.

Si le temps le permet

- Implémentation d'un wallet pour gérer ses transactions
- Intégration de smart contracts à la blockchain

Livrables

Les livrables seront les suivants :

1. Une documentation contenant :
 - Une analyse de l'état de l'art des protocoles de consensus
 - Les choix effectués découlant de l'analyse
 - Spécifications de la blockchain
2. Une cryptomonnaie implémentée en Rust.

Table des matières

Préambule	v
Authentification	vii
Cahier des charges	ix
1 Introduction	1
1.1 Problématique	1
1.2 Objectifs	1
1.3 Pourquoi ce projet ?	2
1.4 Organisation	2
2 Fonctionnement des cryptomonnaies	3
2.1 Qu'est-ce qu'une cryptomonnaie ?	3
2.2 La blockchain	4
2.3 Les transactions	8
2.4 Réseau pair-à-pair distribué	8
2.5 Qu'est-ce qu'un protocole de consensus ?	8
3 État de l'art	9
3.1 Protocoles de consensus	9
3.1.1 Proof of work	9
3.1.2 Proof of stake, delegated proof of stake	10

3.1.3	Proof of authority, proof of reputation	11
3.1.4	Proof of space, proof of space-time	11
3.1.5	Proof of replication, catalytic space	12
3.1.6	Proof of weight	12
3.1.7	Proof of importance	12
3.1.8	Proof of burn	13
3.1.9	Proof of history	13
3.1.10	Byzantine Fault Tolerance	15
3.2	Analyse des protocoles	15
3.3	Attaques sur les blockchains	17
3.3.1	Attaque Sybil	17
3.3.2	Attaque des 51%	17
3.3.3	Grinding attacks	17
3.3.4	Reécriture de l'historique	17
3.3.5	Problème du "Nothing-to-stake"	17
3.4	Mécanismes de sécurisation	17
3.4.1	Zcash	18
3.4.2	Monero	18
3.5	Cryptomonnaies de 3ème génération	18
3.5.1	Performance des blockchains	18
3.5.2	Cardano	19
3.5.3	IOTA	19
3.5.4	Solana	19
4	Proof of space	21
4.1	Introduction	21
4.2	Résumé du protocole	21
4.3	Plotting	22
4.4	Farming	25

4.5	Vérification	25
4.6	Sécurité et attaques	26
5	Dossier de réalisation	27
5.1	Choix des langages et technologies utilisées	27
5.2	Cryptographie	27
5.3	Implémentation de la proof of space	28
5.4	Implémentation de la blockchain	28
5.4.1	Merkle tree	28
5.4.2	Transactions	28
5.4.2.1	Signature	28
5.4.3	Blocs	28
5.4.4	Registre	29
5.5	Réseau peer-to-peer et communication	29
5.6	Tests réalisés	29
6	Conclusion	31
6.1	Résultat final	31
6.2	Difficultés rencontrées	31
6.3	Objectifs non-réalisés	32
6.4	Améliorations possibles du projet	32
6.5	Conclusion personnelle	32
	Bibliographie	33

Chapitre 1

Introduction

1.1 Problématique

Le Bitcoin est l'une des cryptomonnaies les plus connues et une des plus utilisées aujourd'hui en 2021. Cependant, elle n'est pas parfaite et certains points sur sa conception et son fonctionnement peuvent poser problème. Un des points les plus critique est la vérification des transactions qui est très consommatrice d'énergie. En effet, les mineurs de Bitcoin doivent allouer une grande quantité de puissance de calcul pour sécuriser la blockchain. A titre d'exemple, la puissance totale de tous les mineurs de Bitcoin regroupés permettrait d'alimenter un pays de taille comparable aux Pays-Bas [**BTC_cons**].

Un autre problème notable est confidentialité des transactions. Il faut savoir que les transactions effectuées sur la blockchain Bitcoin (et bien d'autres) sont consultables publiquement. Il est possible de voir le montant de chaque transaction et de parcourir la blockchain pour trouver le solde d'un compte. Et ceci très facilement grâce à explorateur de blockchain comme par exemple `blockchain.com`. Cela n'est ainsi pas adapté à des transactions sensibles comme des versements de salaire par exemple.

1.2 Objectifs

L'objectif principal de ce travail de Bachelor est le développement d'une cryptomonnaie et de sa blockchain pour palier à certains problèmes existants sur le Bitcoin actuellement. Le développement d'une cryptomonnaie étant potentiellement long et complexe, ce travail se limitera à une blockchain simplifiée mais fonctionnelle. L'objectif n'étant pas de recréer une cryptomonnaie aussi complète que le Bitcoin mais de se concentrer sur les aspects techniques liés à l'écologie et à la confidentialité dans les blockchains.

Le but est de comprendre pourquoi le Bitcoin consomme-t-il tant d'énergie et de trouver et développer une solution plus écologique. Une analyse approfondie des protocoles de consensus sera effectuée dans ce travail afin de savoir lequel serait le plus adapté pour répondre à la problématique. Il faudra ensuite implémenter un de ses protocoles en Rust et, à partir de cette implémentation, créer une blockchain.

Résumé des objectifs :

- Analyse des protocoles de consensus
- Implémentation d'un protocole plus écologique que proof of work.
- Implémentation d'une blockchain utilisant le protocole réalisé

1.3 Pourquoi ce projet ?

En plus de résoudre la problématique décrite ci-dessus, ce travail a aussi pour but de permettre la compréhension du fonctionnement technique des cryptomonnaies à des personnes intéressées par ces technologies.

1.4 Organisation

Ce travail a commencé le 26 février 2021. Selon le planning de la HEIG-VD, un rendu intermédiaire est planifié le 20 mai 2021. La date de rendu finale est le 30 juillet 2021.

Le code source de ce travail est disponible sur GitHub au sein de l'organisation Spaceframe.

Chapitre 2

Fonctionnement des cryptomonnaies

2.1 Qu'est-ce qu'une cryptomonnaie ?

Dans un premier temps, il est nécessaire de comprendre comment fonctionnent une cryptomonnaie avant de pouvoir en créer une. Et il faut tout d'abord pouvoir définir *qu'est-ce qu'une cryptomonnaie*.

Une cryptomonnaie est un ensemble de mécanismes cryptographiques utilisés dans le but de sécuriser un registre distribué afin d'obtenir un système de paiement décentralisé. Plus simplement, une cryptomonnaie est un système de paiement qui n'est pas régi par une entité centrale, comme une banque par exemple, et qui permet à tout le monde d'émettre des transactions qui seront incluses dans un registre. Ce registre est publique et partagé avec tous les utilisateurs via un réseau pair-à-pair.

Mais comme partout, il y a des personnes honnêtes mais aussi des gens malhonnêtes qui vont essayer d'abuser des failles du système. Dans le cas d'un système de paiement décentralisé, il n'y a pas de banque pour vérifier la validité des transactions ce qui pose un réel problème de sécurité. Du coup, pour rendre le système sûr, on utilise de la cryptographie. La cryptographie, grâce à des preuves mathématiques, rend la validité (ou non) des transactions quasiment irréfutable et permet à n'importe qui de vérifier les transactions stockées dans le registre.

Chaque utilisateur possède une copie de ce registre de transactions sur son ordinateur en local et peut recevoir de nouvelles transactions d'autres utilisateurs et en envoyer lui-même. Chaque utilisateur vérifie les transactions entrantes et les ajoute à son registre si elles sont correctes.

2.2 La blockchain

Comme vu ci-dessus, une cryptomonnaie est en fait un registre dans lequel chacun peut ajouter des transactions. Imaginons maintenant les utilisateurs Alice, Bob et Charlie. Chacun d'entre eux peut ajouter une ligne au registre qui dit par exemple « Alice paie Bob 5 CHF », « Bob paie Charlie 8 CHF ». Et, à interval régulier, ils se regroupent pour se transmettre l'argent indiqué dans le registre.

Registre
Alice paie Bob 5 CHF
Bob paie Charlie 8 CHF

FIGURE 2.1 – Exemple de registre de transactions simplifié

Seulement rien n'empêche Bob de rajouter la première ligne « Alice paie Bob 5 CHF » autant de fois qu'il veut sans le consentement d'Alice.

Registre
Alice paie Bob 5 CHF
Bob paie Charlie 8 CHF
<i>Alice paie Bob 5 CHF</i>
<i>Alice paie Bob 5 CHF</i>
<i>Alice paie Bob 5 CHF</i>

FIGURE 2.2 – Bob inscrit de fausse transactions dans le registre

C'est là que la cryptographie rentre en jeu. Plus précisément les signatures numériques. Chaque transaction va être accompagnée d'une signature. Les signatures numériques reposent sur de la cryptographie asymétrique. C'est à dire que chaque utilisateur possède une pair de clé : une clé publique pk et une clé privée ou secrète sk . La clé publique est partagée

avec les autres utilisateurs du registre alors que la clé privée reste bien cachée chez son propriétaire et n'est jamais divulguée.

À la différence des signatures manuscrites sur papier qui sont à chaque fois presque identiques, une signature numérique change en fonction du message à signer et de la clé secrète. On peut imaginer la signature d'un message m avec la fonction suivante :

$$\text{Sign}(m, sk) = \text{signature}$$

Pour vérifier une signature numérique, on utilise la clé publique associée à la clé privée utilisée pour signer le message. Comme cette clé est connue de tous, tout le monde peut vérifier la signature du message :

$$\text{Verify}(m, \text{signature}, pk) = \{\text{Vrai}, \text{Faux}\}$$

Ainsi, au lieu de simplement ajouter une transaction au registre, les utilisateurs vont y joindre la signature de cette dernière ce qui assurent que l'émetteur de la transaction est d'accord car seulement lui possède la clé privée associée à la clé publique utilisée pour signer le message et qu'il est extrêmement difficile de forger une fausse signature sans posséder la clé privée. Par exemple, si Alice souhaite inscrire une nouvelle transaction dans le registre, elle le ferait de la manière suivante :

$$\begin{aligned} \text{transaction}_{\text{Alice}} &= \text{"Alice paie 10 CHF à Bob"} \\ \text{signature} &= \text{Sign}(\text{transaction}, sk_{\text{Alice}}) \end{aligned}$$


Registre	
Alice paie Bob 5 CHF	 Alice
Bob paie Charlie 8 CHF	 Bob

FIGURE 2.3 – Les transations du registre sont maintenant signées numériquement

De cette manière, Bob ne peut plus rajouter de lignes sans le consentement d'Alice car il ne possède par sa clé privée. A noté également que lors de la vérification de la signature, la clé publique doit appartenir à la personne qui envoie de l'argent, sinon la transaction est rejetée. Cela fait sens car Bob peut signer « Alice paie Bob 10 CHF » et la signature sera techniquement valide mais le contenu du message n'est pas juste. C'est uniquement la personne qui signe la transaction qui donne de l'argent.

Cependant, même si Bob ne connaît pas la clé privée d’Alice, il peut quand même ré-envoyer une ancienne transaction signée par Alice. Sa signature sera toujours valide.





Registre	
Alice paie Bob 5 CHF	 Alice
Bob paie Charlie 8 CHF	 Bob
Alice paie Bob 5 CHF	 Alice
Alice paie Bob 5 CHF	 Alice

FIGURE 2.4 – Bob inscrit deux anciennes transactions d’Alice

C’est pour cela qu’on inclut un identifiant unique, comme un nombre, à la transaction afin d’obtenir une signature différente à chaque transaction même si elles sont identiques. Cela force Alice à refaire une signature à chaque nouvelle transaction et cela empêche Bob de réutiliser une ancienne signature valide d’Alice.




Registre	
1 Alice paie Bob 5 CHF	 Alice
2 Bob paie Charlie 8 CHF	 Bob
3 Alice paie Bob 5 CHF	 Alice

FIGURE 2.5 – Un identifiant unique est signé avec la transaction

Maintenant, imaginons qu’un des utilisateurs, Charlie par exemple, doive beaucoup d’argent et ne vienne pas au rendez-vous pour payer les autres utilisateurs. Il peut écrire autant de transaction qu’il veut dans le registre et rien ne l’empêche de ne pas venir pour payer les autres. Il faut alors trouver un moyen où les utilisateurs n’ont pas besoin de se retrouver pour payer se qu’ils doivent.

On peut imaginer un système dans lequel les utilisateurs mettent une certaine somme d’argent dans un panier commun, disons 100 CHF et les premières lignes du registre indiqueraient combien ils ont mis.

Registre
Alice reçoit 100 CHF
Bob reçoit 100 CHF
Charlie reçoit 100 CHF

FIGURE 2.6 – Premières lignes indiquant la somme initiale donnée par chaque utilisateur

Comme cela, on peut empêcher les utilisateur de faire des transactions dont le montant dépasse ce qu'il possède dans le registre. Par exemple, si Charlie paie à Alice et Bob 50 CHF, il ne pourra pas payer 10 CHF à nouveau à Bob.

Registre	
Alice reçoit 100 CHF	
Bob reçoit 100 CHF	
Charlie reçoit 100 CHF	
<hr/>	
1. Charlie paie Alice 50 CHF	🏆
2. Charlie paie Bob 50 CHF	🏆
3. Charlie paie Bob 10 CHF	🏆

FIGURE 2.7 – Charlie n'as pas assez d'argent pour effectuer la troisième transaction

Cela implique de conserver l'historique complet des transactions pour pouvoir donner le montant restant d'un utilisateur.

Avec ce système, on a une très bonne séparation entre l'argent dans le registre et l'argent réel. On peut même utiliser une monnaie propre au registre, par exemple des « jetons ». Les utilisateurs sont libres d'échanger des jetons contre des francs suisse et inversement, cependant le taux de change entre les deux va dépendre de l'offre et de la demande puisqu'il y a un nombre limité de *jetons* dans le registre. Par exemple, Alice peut donner à Bob 20 CHF et Bob, en échange, écrit une transaction dans le registre de 20 *jetons* pour Alice.

2.3 Les transactions

...

2.4 Réseau pair-à-pair distribué

...

2.5 Qu'est-ce qu'un protocole de consensus ?

Dans une blockchain, un protocole de consensus est un algorithme permettant de mettre l'ensemble des noeuds du réseau d'accord sur une version de la blockchain, ceci en tenant compte du fait que certains noeuds peuvent être malveillants.

Dans une structure centralisée, comme une banque par exemple, les transactions sont vérifiées par la banque elle-même, il est donc difficile de forger de fausses transactions puisque ces dernières sont vérifiées par une entité centrale. Or, dans une structure décentralisée comme une blockchain, tout le monde peut se joindre au réseau et soumettre des blocs avec des transactions. Certains noeuds peuvent transmettre aux autres noeuds des blocs avec des transactions invalides et commettre des actes frauduleux comme de la double dépense.

Il nous faut donc un algorithme permettant de synchroniser tous les noeuds sur une version identique de la blockchain afin de garantir l'authenticité de tous les blocs qu'elle contient et empêcher qu'une même entité contrôle toute la chaîne de blocs. Ainsi, un protocole de consensus va permettre de déterminer quel noeud va pouvoir effectuer les calculs ou actions nécessaires afin d'ajouter un nouveau bloc à la chaîne. Dans le but de motiver les noeuds à agir de manière honnête, le réseau les récompense le plus souvent lors de la création d'un nouveau bloc avec une certaine quantité de cryptomonnaie.

Tous les autres noeuds doivent alors se synchroniser et travailler sur la chaîne de blocs la plus longue s'il y a plusieurs branches disponibles. Le but étant que la chaîne de blocs honnête grandisse plus rapidement que d'autres chaînes isolées ou frauduleuses.

Chapitre 3

État de l'art

3.1 Protocoles de consensus

TODO : Petite intro de section

3.1.1 Proof of work

Proof of work est un des tout premier protocole de consensus créé et est aujourd'hui un des plus utilisé. Il a au début été développé afin se prémunir des spams d'e-mail. Il a, en 2009, été adapté pour les blockchains par Satoshi Nakamoto en créant le **Bitcoin**.

Le protocole *proof of work* utilise des ordinateurs appelés mineurs pour vérifier les blocs en résolvant des puzzles mathématiques. La résolution de ses puzzles requiert une grande puissance de calcul et une grande quantité d'énergie.

Techniquement, les mineurs utilisent des fonctions de hachage cryptographiques. Pour simplifier le processus, ces derniers doivent hacher l'ensemble des données du dernier bloc ($Bloc_n$) et un nombre (p) qu'ils peuvent choisir. Ce nombre est appelé la **preuve de travail**. Ci-dessous, un exemple du calcul réalisé par un mineur :

$$H = \text{SHA256}(\text{Bloc}_n \| p)$$

L'objectif est que le hash final H commence par un certain nombre de 0. Ce nombre est fixé par le réseau. Comme les fonctions de hachage ne permettant pas de retrouver les données fournies en entrée à partir de la sortie, il n'y a pas d'autre moyen que de tester toutes les entrées possible jusqu'à ce que le hash commence par le bon nombre de 0. Ainsi le mineur va modifier la preuve de travail p jusqu'à ce que le hash H remplisse les conditions de la blockchain.

Le nombre de 0 demandé en sortie est ce que l'on appelle la **difficulté** de la preuve de travail car plus il y a de 0 à la suite, plus il faudra de temps au mineur pour trouver la preuve de travail correspondante. Ce nombre de 0 est automatiquement défini par le réseau et est adapté à la puissance de calcul globale. C'est-à-dire que si la puissance augmente parce que de nouveaux mineurs ont rejoint le réseau, alors la difficulté augmentera également. Ceci afin de maintenir le temps de création d'un bloc à environ 10 minutes. Si des mineurs viendraient à quitter le réseau, il y aurait moins de puissance de calcul disponible, la difficulté sera alors revue à la baisse.

3.1.2 Proof of stake, delegated proof of stake

Le protocole de *proof of stake* (preuve d'enjeu en français) est le deuxième protocole le plus utilisé dans les blockchains actuelles. Il fonctionne sur un principe totalement différent que PoW car il ne requiert pas de puissance de calcul particulière ce qui en fait une bonne alternative en terme d'énergie. La cryptomonnaie Ether du réseau Ethereum est en train d'effectuer une migration vers du *proof of stake* en 2021.

Proof of stake fonctionne sur le principe de staking. C'est-à-dire allouer une certaine quantité de cryptomonnaie au réseau. Cette monnaie bloquée appartient toujours à l'utilisateur mais ne peut plus être utilisée. Les nouveaux blocs seront créés par les utilisateurs qui mise le plus de jetons dans le réseau. Si ces personnes ne respectent pas leur engagement de contribuer au réseau de manière légitime, elle perdraient leur mise ce qui potentiellement ruinerait ces derniers.

Avec cette architecture, pour pouvoir effectuer un acte de double dépense, il faudrait posséder et bloquer plus de la moitié de tous les jetons misés sur le réseau pour le contrôler ce qui rend ces attaques difficiles. Mais le fait que *PoS* ne requiert pas de puissance de calcul amène d'autres problèmes qui n'existaient pas avec *PoW* comme par exemple le problème du Nothing-to-stake.

Un autre problème que l'on peut remarquer avec ce protocole est que ce sont toujours les utilisateurs qui mise le plus qui sont prioritaires pour ajouter des blocs à la blockchain ce qui peut rendre le protocole trop centralisé alors que l'on recherche plutôt l'inverse.

Pour palier à ce problème, on a créé le *delegated proof of stake*. Le principe est qu'on utilise cette fois-ci un système de vote dans lequel chaque utilisateur possède un nombre de voix proportionnel à la quantité de monnaie mise dans le réseau. Le système de vote varie en fonction des implémentations. Cela rend le protocole plus démocratique et ainsi ce ne sont pas toujours les mêmes entités qui ajoutent des blocs.

3.1.3 Proof of authority, proof of reputation

Proof of authority est un algorithme proposé par un des cofondateur d'Ethereum, Gavin Wood. Ce protocole se base lui sur la **réputation** des entités qui valident les blocs. A la différence du *proof of stake* qui se sert de la monnaie, *PoA* met en valeur l'identité des validateurs qui sont sélectionnés comme entités de confiance.

Il y a ainsi un nombre limité de validateurs ce qui rend le réseau plus évolutif et efficace qu'un système avec du *proof of work* ou *proof of stake* car le consensus peut être atteint plus rapidement.

Proof of authority est un protocole qui se porte particulièrement bien au **blockchains privées** permettant aux entreprise d'utiliser pleinement la technologie de la blockchain avec une architecture centralisée. En effet, l'aspect décentralisé du *PoW* et *PoS* peut ne pas convenir à certaines sociétés. D'un autre côté, ce protocole ne s'adapte pas très bien au blockchain publique du fait de sa centralisation. Centralisation que les utilisateurs des blockchains cherchent à éviter pour des raisons de confidentialité (politique) et de sécurité (pannes, attaques).

On peut voir le *PoA* comme un renoncement à la décentralisation dans un but d'efficacité mais ce mécanisme n'est pas vu de la même manière par tous. Notamment critiqué à cause des risques de corruption possibles si les identités des validateurs sont connus. En effet, un concurrent pourrait influencer les validateurs pour compromettre le réseau de l'intérieur.

En conclusion, *PoA* est une bonne alternative au *PoW* et *PoS* pour les **blockchains privées** d'entreprise souhaitant utiliser ses technologies.

3.1.4 Proof of space, proof of space-time

Proof of space est un protocole ressemblant à *proof of work* à la différence qu'au lieu de réaliser des puzzles mathématiques, les mineurs appelé farmers vont réalisés des preuves cryptographiques en allouant de l'**espace disque inutilisé** au réseau. Il est également appelé *proof of capacity*. Ce principe permet de créer des preuves et valider les blocs rapidement avec un coût énergétique beaucoup plus faible que *PoW*. Ainsi on utilise la capacité de stockage comme ressource au lieu de la puissance de calcul.

Cependant, comme les vérifications peuvent être faites très rapidement comparé au Bitcoin où il faut trouver la solution au puzzle qui prend obligatoirement du temps, des nouvelles attaques apparaissent. Par exemple, un attaquant peut valider une grande quantité de blocs à la suite et les soumettre au réseau d'un seul coup. Chose qui n'est pas possible avec *proof of work* puisque qu'il faut nécessairement trouver la preuve de travail avant de vérifier le suivant. Or trouver la preuve de travail prend du temps, beaucoup plus qu'avec *PoSpace*.

Pour éviter ce problème il existe plusieurs solutions. La première est de pénaliser les farmers

agissant de manière malicieuse en intégrant un type de transaction propre aux pénalités. Cette manière de faire a été décrite dans le document de Spacemint [DBLP:conf/fc/ParkKFGAP18]. Une autre solution est d'utiliser des preuve de temps (*proof of time*) grâce à des fonctions à délai vérifiable (VDF). Cette solution a été choisie par le réseau Chia. Elle met en relation *proof of space* et *proof of time* pour donner un protocole de *proof of space-time*. C'est-à-dire que les farmers prouvent au réseau qu'ils ont stocké une certaine quantité de données pendant un certain temps.

A noté que les données stockées sont inutiles dans le sens où elles ne représentent rien de particulier. C'est donc l'espace de stockage perdu au profit de la validation de blocs.

3.1.5 Proof of replication, catalytic space

Proof of replication est une adaptation de *proof of space* dans laquelle une majorité de l'espace de stockage peut être utilisé pour **stocker des données utiles**. Ici les farmers génèrent des preuves en prouvant qu'ils ont stocké des replicas de fichiers sur leurs disques. Ce principe est utilisé notamment dans la cryptomonnaie Filecoin. En revanche, l'infrastructure à mettre en place avec un tel protocole est beaucoup plus complexe qu'avec du *proof of space* simple.

3.1.6 Proof of weight

Le mécanisme de consensus par *proof of weight* est un algorithme basé sur le modèle Algorand. Ce modèle basé sur un protocole de Byzantine agreement permet de vérifier rapidement les transactions et peut gérer beaucoup d'utilisateurs.

Les blockchains utilisant *proof of weight* assignent aux utilisateurs un **poids relatif** à une ressource qu'ils mettent à disposition de la blockchain. *Proof of stake* est en quelque sorte un protocole de type *proof of weight* dans lequel la quantité monnaie mise représente un poids. Plus ce poids est élevé, plus l'utilisateur a de chance de créer le prochain bloc.

Mais *proof of weight* ne se limite pas à la quantité de monnaie mise sur le réseau comme *proof of stake*. Par exemple, avec Filecoin, le poids est défini par la quantité de données IPFS d'un utilisateur. On peut également adapter un protocole de type *proof of space* pour assigner aux utilisateurs un poids relatif à l'espace de stockage alloué au réseau.

3.1.7 Proof of importance

Proof of importance est un algorithme qui met l'accent sur les utilisateurs les plus importants sur le réseau, c'est-à-dire les utilisateurs qui effectuent le plus de transactions. Ainsi, plus un utilisateur aura fait de transactions, plus il aura de chance d'être sélectionné pour créer le prochain bloc. Cela a pour but de favoriser le transfert et le mouvement de la monnaie à

travers le réseau. En opposition avec *proof of stake* qui favorise les utilisateurs à garder et bloquer leur argent.

Proof of importance peut ainsi être utilisé en plus de *proof of stake* dans le but d'améliorer ce dernier. Cela résout une des principales critiques de *PoW* et *PoS* incite les utilisateurs à bloquer leur monnaie et se faisant centralise le système autour des personnes possédant le plus. *PoI* en plus de *PoS* permet d'éviter cette centralisation car les utilisateurs ne faisant pas de transaction seraient considéré comme moins important que les autres.

3.1.8 Proof of burn

Proof of burn (preuve de destruction en français) est proposé comme une alternative à *proof of work*. C'est un protocole qui permet aux utilisateurs de **brûler des coin** afin de prouver leur dévouement envers la blockchain. Ainsi plus un utilisateur brûle de coins, plus il aura de chance d'être sélectionné pour créer le prochain bloc. Ce principe utilise du coup moins d'énergie puisqu'il n'y a pas besoin de grande de puissance de calcul.

Le fonctionnement est le suivant : les utilisateurs souhaitant sécuriser le réseau vont envoyer des coins à une adresse d'incinération. Cette adresse rend les coins inutilisables ce qui crée une pénurie plus ou moins constante augmentant sa valeur potentielle. C'est un moyen d'investir dans la sécurité du réseau.

Il y a plusieurs moyen de mettre en oeuvre ce protocole. On peut sécuriser la blockchain en brûlant des Bitcoin ou bien certaines cryptomonnaie arrivent à le faire en brûlant leur propre monnaie.

En comparaison avec le *proof of stake*, les coins sont ici brûlés et donc inutilisables après alors qu'avec la preuve d'enjeu, l'utilisateur souhaitant se retirer peut débloquent son argent et l'utiliser à nouveau. Cela ne crée ainsi pas de pénurie permanente.

Le *proof of burn* a des avantages comme son aspect écologique ou encore le fait que les mineurs n'aient pas besoin de matériel particulier. Cependant il aussi des inconvénients comme le fait qu'il n'a jamais été mis en place à grande échelle. Le fait aussi de brûler des Bitcoin qui on été forgés avec du *PoW* rend le protocole tout de suite moins écologique.

3.1.9 Proof of history

Proof of history [**proof:poh**] est un protocole permettant de résoudre les problèmes de synchronisation au sein d'un système distribué grâce à des **fonctions à délai vérifiables (VDF)**. Ce n'est **pas** un protocole de consensus à lui seul cependant il est très intéressant et novateur c'est pourquoi il est inclu dans ce chapitre. Un des plus gros problème avec les blockchains et les systèmes décentralisés est la synchronisation des événements. S'assurer qu'un événement *B* à bien eu lieu après un événement *A* et avant un événement *C* et qu'il est

impossible d'en modifier l'ordre après coup. *Proof of history* permet d'accomplir cela grâce à une fonction de délai vérifiable sous forme de fonction de hachage itérative. C'est une fonction qui permet de prouver qu'un certain temps réel s'est bien écoulé et est facilement vérifiable par d'autres utilisateurs.

Le protocole est ainsi une fonction de hachage qui s'appelle en boucle comme illustré dans le schéma ci-dessous.

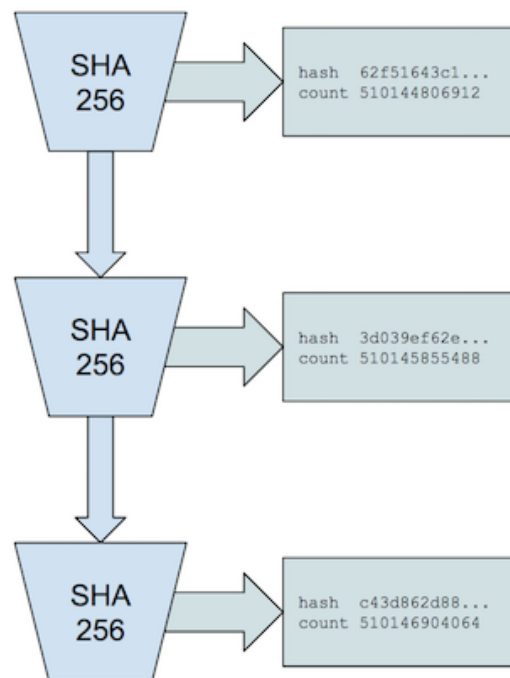


FIGURE 3.1 – Schéma simplifié de proof of history

On peut y injecter des entrées à tout moment. Cela va changer de manière imprédictible les données futures et ainsi ancrer les données dans l'historique de la chaîne de hachage.

Ce mécanisme a été inventé par Anatoly Yakovenko et est utilisé par la blockchain Solana. Ce protocole est utilisé avant un algorithme de consensus de type *Delegated Byzantine Fault Tolerance*. L'auteur appelle le *Proof of History* : **Clock before consensus**. C'est-à-dire que le *PoH* se charge de remettre dans l'ordre les événements avant faire le consensus.

Ce protocole permet d'obtenir une bande passante de transactions très élevée, jusqu'à 50'000 transactions par seconde d'après les créateurs rendant le protocole plus performant que Bitcoin qui atteint lui une douzaine de transactions par secondes.

3.1.10 Byzantine Fault Tolerance

Byzantine Fault Tolerance est un large groupe de protocoles permettant d'atteindre un consensus entre les noeuds du réseau en prenant en compte le fait que des noeuds peuvent être indisponibles, transmettre des informations erronées ou être malhonnêtes. Il existe différents types d'algorithme basés sur *BFT* mais la plus part d'entre eux ont un fonctionnement similaire.

Avec les protocoles *BFT*, pour simplifier, un noeud est choisi pour créer le prochain bloc. Cela peut se faire grâce à un système de vote ou aléatoirement sans tenir compte d'une ressource en particulier. Le bloc est ensuite transmis à travers le réseau à plusieurs autres noeuds. Ces autres noeuds vont faire certaines vérifications et transmettre le bloc à leur tour et le consensus est atteint lorsque la majorité des noeuds a reçu le bloc et est d'accord sur la version de la chaîne de bloc. Ensuite un autre noeud est sélectionné pour le bloc suivant.

Il est facile simple de détecter les fraudes car si les blocs sont invalides ils seront supprimés par les autres noeuds. Cependant les protocoles *BFT* sont pour la plupart vulnérables aux **attaques de Sybil** donc si une majorité des noeuds du réseau sont malhonnêtes, il sera impossible d'obtenir un consensus correct. C'est pour cela qu'ils sont souvent associer à d'autres protocoles comme *PoW* ou *PoS*.

3.2 Analyse des protocoles

Tous ces protocoles permettent d'une manière ou d'une autre de sécuriser la blockchain. Cependant ils ne respectent pas tous les contraintes écologiques de ce travail. Afin de pouvoir faire un choix, ces protocoles seront ci-dessous analysés afin de pouvoir trouver les mieux adaptés selon différents critères comme l'impact écologique, la facilité d'implémentation, les ressources disponibles et autre points importants.

- **Proof of work** : ce protocole est certes le plus populaire et possède le plus de ressources sur internet, il est cependant le moins écologique de tous. C'est principalement pour cette raison qu'il ne sera pas question d'implémenter un protocole comme *PoW* ou autres protocoles basés sur ce dernier. A noter que l'implémentation d'un algorithme de *proof of work* est plus simple que la majeure partie des autres protocoles.
- **Proof of stake** : écologiquement plus intéressant que PoW, la preuve d'enjeu peut se trouver être une bonne alternative en terme de consommation d'énergie. Cependant, comme elle fonctionne sur le principe de bloquer de l'argent pour la sécurité du réseau, cela implique qu'il faut nécessairement une certaine quantité d'argent dès le début, ce qui est problématique pour commencer une blockchain à partir de rien. *Ethereum* qui effectue une migration de PoW vers PoS possède déjà beaucoup d'argent en circulation pouvant être bloqué pour faire de la preuve d'enjeu convenablement. Mais à partir de rien, c'est conceptuellement difficile ce qui rend l'implémentation plus compliquée.

- **Proof of authority** : PoA est une bonne alternative pour les blockchains privées. Son implémentation est plutôt simple car il n'y a pas besoin d'utiliser des ressources particulières comme de la puissance de calcul, une somme d'argent ou de l'espace de stockage. Il suffit de simplement vérifier les noeuds validateurs qui sont explicitement autorisés par une entité. Il est du coup difficile pour n'importe qui de devenir validateur et contribuer à la sécurité du réseau. C'est pourquoi elle est adaptée à des blockchains privées or, dans ce travail, on souhaite implémenter une cryptomonnaie publique où tout le monde peut contribuer.
- **Proof of space** : ce protocole est intéressant du point de vue énergétique car il fonctionne comme *PoW* mais ne demande pas de puissance de calcul mais de l'espace de stockage à la place. Son empreinte écologique est du coup beaucoup plus faible. Son implémentation est cependant plus complexe car il faut utiliser des graphs permettant de prouver qu'un utilisateur a bien stocké tant de données et ces derniers ne sont pas forcément faciles à réaliser. Mais il existe des implémentations de *proof of space* comme Chia ou encore Spacemesh ce qui fournissent déjà une quantité de ressources acceptable. Ce protocole est un très bon candidat pour ce travail car bien adapté.
- **Proof of replication** : ce protocole possède les mêmes avantages que *proof of space* vue juste au dessus mais avec en plus l'avantage que les données stockées sont des données utiles. Mais en revanche l'implémentation est beaucoup plus compliquée. Prouver qu'un utilisateur a stocké des réplicas de fichiers est plus complexe mathématiquement que de prouver qu'il a stocké des données pseudo-aléatoires. Il y a des ressources comme Filecoin qui sont disponibles et peuvent aider à comprendre le fonctionnement mais cela reste trop de travail pour ce projet.
- **Proof of weight** : étant une famille de protocole, il n'existe pas un protocole mais plusieurs avec par exemple *proof of stake* qui peut être associé à du *proof of weight* d'une certaine manière. En tant que tel, *proof of weight* est assez jeune et très peu utilisé avec peu d'information disponible sur internet, c'est pourquoi il sera mis de côté mais reste un protocole intéressant sachant qu'on peut dériver du *PoSpace* en *PoWeight* en assignant un poids relatif à l'espace de stockage.
- **Proof of importance** : le principe de ce protocole est également intéressant mais est, à nouveau, très jeune avec peu de ressources disponibles. Se lancer dans l'implémentation d'un tel protocole est trop risqué.
- **Proof of burn** : *Proof of burn* a un concept assez spécial et peut être un protocole écologique si les coins brûlés sont les coins de la cryptomonnaie même. Cependant ce protocole n'a jamais été testé à large échelle et se trouve être assez théorique avec très peu d'implémentations existantes et de ressources disponibles le rendant peu attractif.
- **Proof of history** : probablement le protocole le plus intéressant avec *proof of space* car c'est également un protocole plus écologique que *PoW*. Son concept est innovant et utilisé dans la blockchain Solana donc déjà plus ou moins déployé. Cependant les concepts utilisés sont plus obscures que ceux utilisés avec *proof of space* ce qui rend son implémentation plus compliquée car plus difficile à comprendre. Il y a quelques ressources disponibles sur la documentation de Solana mais moins que sur le réseau

Chia par exemple.

- **Byzantine Fault Tolerance** : cette famille de protocole englobe tous les algorithmes de consensus vu ci-dessus, ce n'est pas un protocole à proprement dit. On utilise un ou plusieurs protocoles vu précédemment pour avoir de la *Byzantine Fault Tolerance* au sein d'un réseau d'ordinateurs distribués. Cela pour éviter principalement les attaques de Sybil.

3.3 Attaques sur les blockchains

3.3.1 Attaque Sybil

...

3.3.2 Attaque des 51%

...

3.3.3 Grinding attacks

...

3.3.4 Réécriture de l'historique

...

3.3.5 Problème du "Nothing-to-stake"

...

3.4 Mécanismes de sécurisation

La technologie de la blockchain apporté par Bitcoin est révolutionnaire. Pouvoir faire des paiements sans passer par un organisme centrale est génial mais le problème pour réaliser cela est qu'il faut conserver publiquement un registre de toutes les transactions. Les entrées et sorties des transactions sont identifiées par des adresses dérivées des clés publiques des utilisateurs. Les transactions sont ainsi pseudonymes. Satoshi Nakamoto l'a dit dans son

whitepaper : *The public can see that someone is sending an amount to someone else, but without information linking the transaction to anyone.*

Mais les transaction ne sont pas anonymes. On peut quand même voir le montant, de qui et vers qui va l'argent même si on ne sais pas forcément qui sont derrières les adresses. Il faut bien faire la distinction entre *anonyme* et *pseudonyme*.

Cependant, il existe des blockchains qui permettent d'anonymiser leurs transactions. Rendant illisible le montant et les adresses ainsi que tout autres données. Les deux cryptomonnaies les plus connue à ce sujet sont présentées ci-dessous.

3.4.1 Zcash

...

3.4.2 Monero

...

3.5 Cryptomonnaies de 3ème génération

On peut qualifier le Bitcoin de 1ère génération de cryptomonnaie. Il y a eu ensuite Ethereum et l'apparition des smart-contracts considéré comme cryptomonnaie de 2ème génération. Et maintenant il y a les cryptomonnaies de 3ème génération qui tentent de résoudre les problèmes présents avec le Bitcoin et Ethereum. Notamment les soucis de performance et de stockage.

3.5.1 Performance des blockchains

Un point important est la performance des blockchains. Or le Bitcoin ne peut traiter que 4 à 5 transactions par seconde ce qui le rend très peu performant comparé au réseau VISA qui traite environ 2000 transactions par seconde en moyenne et peut monter plus haut en cas de forte affluence. Le problème vient du fait qu'il faut 10 minutes pour générer un bloc et que chaque bloc fait au maximum 1 mégaoctet. Ces informations sont codées en dur dans le code source de Bitcoin ce qui veut dire qu'il est possible de les modifier pour améliorer les performances du réseau Bitcoin. Mais alors pourquoi les développeurs ne l'ont pas fait ?

Il y a deux possibilités : 1. réduire le temps de génération d'un bloc et 2. augmenter la taille maximum d'un bloc.

La première possibilité est compliquée à mettre en place car la propagation d'un nouveau bloc à travers le réseau prend du temps. Réduire le temps de création implique de réduire la difficulté de la preuve de travail ce qui fera que plus de mineurs généreront plus de blocs rendant le consensus plus compliqué à cause du nombre de mineurs présent sur le réseau et d'une propagation lente.

La deuxième possibilité a fait de grands débats au sein de la communauté. C'est facile d'augmenter la taille des blocs mais cela implique que la blockchain prendra plus d'espace de stockage à l'avenir. Certains développeurs étaient pour une augmentation et d'autres non ce qui est venu à créer *Bitcoin Cash*, un "hard fork" de Bitcoin. Les développeurs de Bitcoin Cash ont décidé d'augmenter la taille maximum des blocs. Ils ont choisi un bloc dans la blockchain Bitcoin et ont créé une nouvelles branches à partir depuis laquelle les nouveaux blocs de Bitcoin Cash seront créés. Par ailleurs, si vous aviez des Bitcoins avant la séparation vous avez du coup le même montant en Bitcoin Cash (BCH) mais aussi en Bitcoin. Comme les deux branches sont séparées l'une de l'autre, c'est comme si les coins s'étaient dupliqués.

3.5.2 Cardano

...

3.5.3 IOTA

...

3.5.4 Solana

...

Chapitre 4

Proof of space

4.1 Introduction

La *preuve d'espace* (ou *proof of space* en anglais) est un algorithme de consensus similaire à la preuve de travail à la différence qu'au lieu d'effectuer des calculs de manière continue pour trouver une preuve satisfaisant les conditions de la difficulté de la blockchain, les mineurs appelés farmers avec *PoSpace* vont trouver des preuves à partir d'un challenge grâce à une certaine quantité de données allouées sur le disque dur.

Il existe principalement deux constructions pour faire du proof of space : une basée sur l'inversion de fonctions de hachage en utilisant le compromis temps-mémoire de Hellman [DBLP:conf/asiacrypt/AbusalahACKPR17] et une basée sur des *hard-to-pebble graphs* [DBLP:conf/crypto/DziembowskiFKP15].

Pour ce travail, il a fallu faire un choix d'un algorithme à implémenter. Ce choix c'est porté sur la construction des compromis temps-mémoire car il y a plus de ressources disponibles. De plus, la construction des graphs de type *hard-to-pebble* s'avère être trop compliquée pour un travail comme celui-ci, c'est aussi pourquoi elle a été écartée.

Ce chapitre explique le fonctionnement général du protocole principalement tiré de la blockchain Chia. Les détails d'implémentation sont décrits dans le chapitre suivant [5].

4.2 Résumé du protocole

Dans un premier temps, les données sont générées par le farmer dans une étape appelée **plotting**. Ce sont des données pseudo-aléatoires qui ne représentent rien de particulier. D'autres algorithmes pour stocker des données utiles existent comme *proof of catalytic space* ou *proof of replication* utilisés dans la blockchain Filecoin par exemple.

En suite, un farmer peut prouver qu'il a bien les données qu'il dit avoir stockées en trouvant une ou plusieurs preuves répondant à un challenge donné. C'est l'étape du **farming**.

Lors de la **vérification**, le vérificateur récupère la preuve et à partir de celle-ci, recalcule le challenge et vérifie s'il correspond à celui envoyé précédemment.

Ainsi, on peut considérer *proof of space* comme est un protocole dans lequel nous avons :

1. un vérificateur qui envoie un challenge à un farmer et
2. un farmer qui peut prouver au vérificateur qu'il a bien alloué la quantité d'espace spécifiée à un instant t .

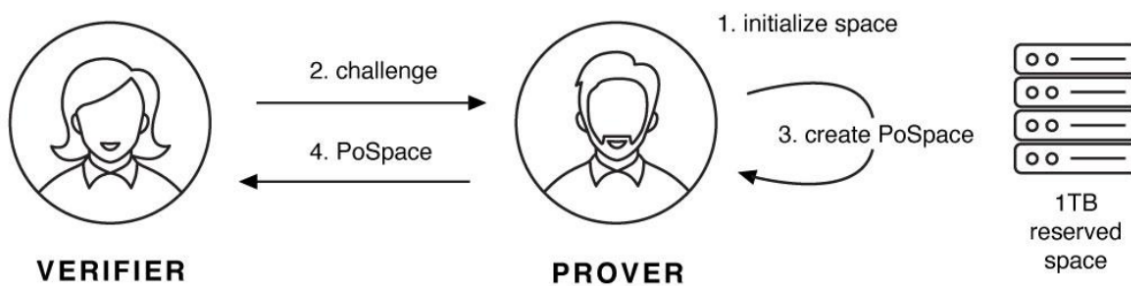


FIGURE 4.1 – Diagramme du protocole *proof of space* montrant les différentes étapes. Tiré de [chia:consensus]

4.3 Plotting

Le plotting est un procédé non-interactif dans le lequel le farmer va créer les données qui seront stockées et qui permettront de trouver les preuves d'espace liées aux challenges. Ce procédé peut être long, cela peut prendre de plusieurs heures à plusieurs jours suivant la quantité de données allouées.

La construction de cette preuve d'espace est basée sur la construction de Chia [chia:construction], elle-même basée sur l'inversion de fonctions de hachage [DBLP:conf/asiacrypt/AbusalahACKPR17]. Elle a cependant été simplifiée pour pouvoir être réalisée dans le temps imparti de ce travail.

Le principe est le suivant : soit une fonction modélisée comme un oracle aléatoire $f : [N] \rightarrow [N]$, où $N = 2^k$ et k étant le *space parameter*, le farmer doit être capable de l'inverser rapidement. Pour ce faire, on peut imaginer que le farmer précalcule la table de la fonction et la trie par rapport à la valeur de sortie. Ainsi, lorsqu'un challenge est donné au farmer, il a simplement à regarder dans la table des valeurs de sortie par une recherche dichotomique et renvoyer l'entrée associée comme avec une lookup-table. Par exemple, on peut modéliser f comme une fonction de hachage cryptographique et la table serait : $f(x) = H(n||x)$, avec n un nonce aléatoire. [chia:construction]

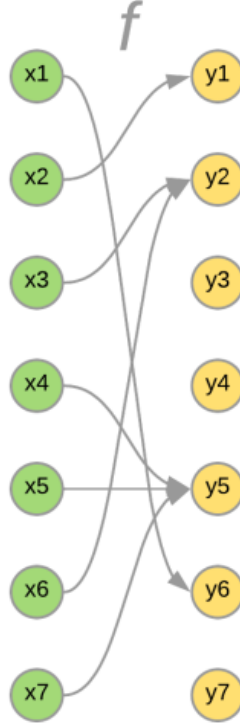


FIGURE 4.2 – Table simplifiée, tiré de [chia:construction]

Cependant, à cause d'un compromis temps-mémoire, il est possible de faire une attaque de *Hellman* avec laquelle il est possible de stocker uniquement $N^{1/2}$ valeurs et les autres $N^{1/2}$ valeurs seront calculées à la demande. Pour éviter un maximum ce problème, il faut rendre la fonction $[N] \rightarrow [N]$ difficile à calculer pour rendre l'initialisation plus longue et réduire le compromis temps-mémoire mais elle doit rester facilement inversible, ceci pour forcer les farmers à stocker l'entièreté de la table. [chia:construction]

Pour rendre plus difficile à calculer vers l'avant, on peut modéliser une nouvelle fonction $f(x_1) = f_2(x_1, x_2)$, où $f_2(x_1, x_2) = H(x_1 \| x_2)$ avec la condition que $f_1(x_1) = f_1(x_2) + 1$, avec f_1 une autre fonction de hachage. Ainsi, à partir d'un challenge z , le farmer doit trouver les valeurs x_1 et x_2 . Et comme la fonction f n'est pas facilement calculable, le farmer doit stocker toutes les tables pour pouvoir répondre rapidement. Cette construction, illustrée ci-dessous, est toujours proie aux attaques de Hellman mais il suffit de rajouter des tables pour la rendre encore plus lente à calculer. [chia:construction]

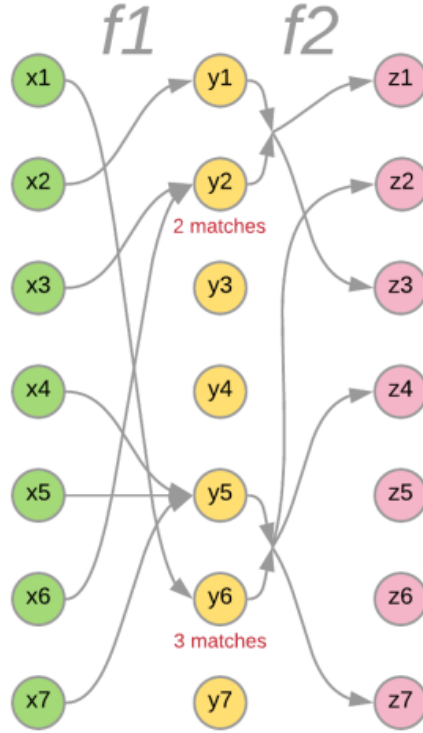


FIGURE 4.3 – Table améliorée, tiré de [chia:construction]

Ainsi, le farmer choisi un nombre k , le *space parameter* pour définir combien il souhaite allouer d'espace. La quantité de données finale sera exponentielle par rapport à ce nombre k . Le temps de la génération dépend de la puissance de l'ordinateur effectuant le plotting puisqu'il s'agit principalement d'exécution de fonctions de hachage comme vu précédemment. C'est le CPU qui sera beaucoup sollicité durant cette étape.

Pour ce travail, le nombre de tables sera de 4 pour simplifier l'implémentation contre 7 pour Chia. Chaque table contiendra 2^k entrées et chaque entrée d'une $table_i$ pointera vers 2 entrées dans la $table_{i-1}$. La première table quant à elle contiendra ce qu'on appellera les *x-values*. Ce sont simplement des nombres entiers de 0 à $2^k - 1$. Ainsi, une preuve d'espace correspond à 8 *x-values* ($2^{\text{nombre de table}-1} = 2^3 = 8$). A noter que ce processus à besoin de plus d'espace de stockage que souhaité pour le plot final car il générera des données qui seront ensuite nettoyées et compressées. La génération des tables occupera plus de place que le plot final. On supprimera ensuite les sorties des fonctions de hachages de chaque table intermédiaire pour ne stocker plus que les index dans la 1ère table pour chaque table.

Une fois les 4 tables générées, on peut commencer à faire du **farming**.

Les détails de l'implémentation et des choix réalisés sont décrits dans le chapitre suivant.

4.4 Farming

Le *farming* est le processus dans lequel le farmer reçoit un challenge et produit une preuve composé de 8 x-values pour démontrer qu'il a bien alloué l'espace indiqué. Le challenge est une valeur de 256 bits (un hash par exemple). Pour ce faire, le farmer parcourt la dernière table de son plot pour voir si un élément match avec le challenge tel que $\text{trunk}(\text{Chall}) = \text{trunk}_k(\text{table}_4[i])$, pour $i \in \{0, 1, \dots, 2^k - 1\}$. S'il trouve une correspondance, le farmer parcourt les tables dans le sens inverse, c'est-à-dire de la 4 à la 1, pour retrouver les x-values puisque $\text{table}_4[i] = f_4(x_1, x_2, \dots, x_8)$. Il renvoie ensuite les 8 valeur au vérificateur. Plus de détails sur l'algorithme sont données dans le chapitre suivant 5.

Dans le cas d'une blockchain, le système est non-interactif. La preuve doit pouvoir être vérifiée par n'importe qui à tout moment sans avoir besoin de demander quoi que ce soit au farmer en question. Pour ce faire le challenge doit être généré à partir d'un élément public, par exemple le hash du dernier bloc de la chaîne. Le nonce utilisé lors de l'initialisation est dans notre cas dérivé de la clé public du farmer et envoyé avec le preuve pour pouvoir être vérifiée par tout le monde.

4.5 Vérification

La vérification consiste à valider une preuve donnée par un farmer. Elle est simple puisqu'il suffit de recalculer les 4 tables avec uniquement les 8 x-values de la preuve et la clé publique du farmer. Ce qui peut être réalisé rapidement car ce n'est que environ $(8 \times 2) - 1 = 15$ exécutions de fonction de hachage qui peuvent être faite en parallèle.

La validateur va executer les fonctions de la manière suivante :

$$\begin{aligned} &f_1(x_1), f_1(x_2), f_1(x_3), f_1(x_4), f_1(x_5), f_1(x_6), f_1(x_7), f_1(x_8) \\ &f_2(x_1, x_2), f_2(x_3, x_4), f_2(x_5, x_6), f_2(x_7, x_8) \\ &f_3(x_1, x_2, x_3, x_4), f_3(x_5, x_6, x_7, x_8) \\ &f_4(x_1, x_2, x_3, x_4, x_5, x_6, x_7, x_8) \end{aligned}$$

Il va vérifier que les sorties correspondent bien, c'est-à-dire que toutes les sorties de la fonction M soient égales à **vrai** :

$$\begin{aligned} &M(f_1(x_1), f_1(x_2)), M(f_1(x_3), f_1(x_4)), \dots \\ &M(f_2(x_1, x_2), f_2(x_3, x_4)), M(f_2(x_5, x_6), f_2(x_7, x_8)) \\ &M(f_3(x_1, x_2, x_3, x_4), f_3(x_5, x_6, x_7, x_8)) \end{aligned}$$

Avec la fonction $M(y_1, y_2)$ qui retourne **vrai** si et seulement $y_1 = y_2 + 1$.

Le vérificateur regarde ensuite si la valeur calculée correspond avec le challenge initial.

4.6 Sécurité et attaques

Comme vu dans le chapitre sur les protocoles de consensus, *proof of space* a des avantages écologiques puisque le calcul n'est réalisé qu'une fois au début lors de la génération du plot. Cela signifie que la génération de preuve ne requiert que très peu de puissance calcul et que par conséquent elle peut être exécutée rapidement à la différence de *proof of work*. Cela implique qu'un attaquant peut forger des blocs en avance, générer leur preuve respective et les broadcaster sur le réseau. Si ces blocs sont valides ils pourraient être acceptés par le réseau qui se base automatiquement sur la plus longue chaîne de blocs. Il y aurait du coup la possibilité pour des attaquant d'effectuer des actes de double dépenses.

Pour palier à ce problème, la blockchain Chia utilise des preuves de temps (*proofs of time* en anglais) basées sur des *Verifiable Delay Functions* afin de vérifier qu'un certain c'est réellement écoulé et empêcher la création de bloc à l'avance. Ces fonctions particulières prennent un temps réel à s'exécuter et ne peuvent pas être parallélisées. Par contre la vérification de la preuve de temps est elle réalisable rapidement. Techniquement, la blockchain utilise des exponentiations répétées sur des groupes d'ordre inconnus [DBLP:conf/crypto/BonehBBF18] [DBLP:conf/innovations/Pietrzak19a] [DBLP:conf/eurocrypt/Wesolowski19].

Malheureusement pour des raisons de temps, il n'a pas été possible d'implémenter durant ce travail un système de *proof of time* comme l'a fait Chia.

Un autre problème de sécurité peut survenir avec cet algorithme si la valeur de k , la paramètre d'espace, est trop faible. Si c'est le cas, alors un attaquant pourrait régénérer le plot trop rapidement dû à sa petite taille. Il pourrait alors générer des plots et des preuves à la volée sans stocker de données sur disque. C'est pour cela qu'une valeur minimum est codée en dur dans le code source du protocole.

Chapitre 5

Dossier de réalisation

5.1 Choix des langages et technologies utilisées

Lorem ipsum dolor sit amet, consectetur adipiscing elit. Ut purus elit, vestibulum ut, placerat ac, adipiscing vitae, felis. Curabitur dictum gravida mauris. Nam arcu libero, nonummy eget, consectetur id, vulputate a, magna. Donec vehicula augue eu neque. Pellentesque habitant morbi tristique senectus et netus et malesuada fames ac turpis egestas. Mauris ut leo. Cras viverra metus rhoncus sem. Nulla et lectus vestibulum urna fringilla ultrices. Phasellus eu tellus sit amet tortor gravida placerat. Integer sapien est, iaculis in, pretium quis, viverra ac, nunc. Praesent eget sem vel leo ultrices bibendum. Aenean faucibus. Morbi dolor nulla, malesuada eu, pulvinar at, mollis ac, nulla. Curabitur auctor semper nulla. Donec varius orci eget risus. Duis nibh mi, congue eu, accumsan eleifend, sagittis quis, diam. Duis eget orci sit amet orci dignissim rutrum.

5.2 Cryptographie

Nam dui ligula, fringilla a, euismod sodales, sollicitudin vel, wisi. Morbi auctor lorem non justo. Nam lacus libero, pretium at, lobortis vitae, ultricies et, tellus. Donec aliquet, tortor sed accumsan bibendum, erat ligula aliquet magna, vitae ornare odio metus a mi. Morbi ac orci et nisl hendrerit mollis. Suspendisse ut massa. Cras nec ante. Pellentesque a nulla. Cum sociis natoque penatibus et magnis dis parturient montes, nascetur ridiculus mus. Aliquam tincidunt urna. Nulla ullamcorper vestibulum turpis. Pellentesque cursus luctus mauris.

5.3 Implémentation de la proof of space

Nulla malesuada porttitor diam. Donec felis erat, congue non, volutpat at, tincidunt tristique, libero. Vivamus viverra fermentum felis. Donec nonummy pellentesque ante. Phasellus adipiscing semper elit. Proin fermentum massa ac quam. Sed diam turpis, molestie vitae, placerat a, molestie nec, leo. Maecenas lacinia. Nam ipsum ligula, eleifend at, accumsan nec, suscipit a, ipsum. Morbi blandit ligula feugiat magna. Nunc eleifend consequat lorem. Sed lacinia nulla vitae enim. Pellentesque tincidunt purus vel magna. Integer non enim. Praesent euismod nunc eu purus. Donec bibendum quam in tellus. Nullam cursus pulvinar lectus. Donec et mi. Nam vulputate metus eu enim. Vestibulum pellentesque felis eu massa.

5.4 Implémentation de la blockchain

Quisque ullamcorper placerat ipsum. Cras nibh. Morbi vel justo vitae lacus tincidunt ultrices. Lorem ipsum dolor sit amet, consectetur adipiscing elit. In hac habitasse platea dictumst. Integer tempus convallis augue. Etiam facilisis. Nunc elementum fermentum wisi. Aenean placerat. Ut imperdiet, enim sed gravida sollicitudin, felis odio placerat quam, ac pulvinar elit purus eget enim. Nunc vitae tortor. Proin tempus nibh sit amet nisl. Vivamus quis tortor vitae risus porta vehicula.

5.4.1 Merkle tree

...

5.4.2 Transactions

...

5.4.2.1 Signature

...

5.4.3 Blocs

...

5.4.4 Registre

...

5.5 Réseau peer-to-peer et communication

Fusce mauris. Vestibulum luctus nibh at lectus. Sed bibendum, nulla a faucibus semper, leo velit ultricies tellus, ac venenatis arcu wisi vel nisl. Vestibulum diam. Aliquam pellentesque, augue quis sagittis posuere, turpis lacus congue quam, in hendrerit risus eros eget felis. Maecenas eget erat in sapien mattis porttitor. Vestibulum porttitor. Nulla facilisi. Sed a turpis eu lacus commodo facilisis. Morbi fringilla, wisi in dignissim interdum, justo lectus sagittis dui, et vehicula libero dui cursus dui. Mauris tempor ligula sed lacus. Duis cursus enim ut augue. Cras ac magna. Cras nulla. Nulla egestas. Curabitur a leo. Quisque egestas wisi eget nunc. Nam feugiat lacus vel est. Curabitur consectetur.

5.6 Tests réalisés

Suspendisse vel felis. Ut lorem lorem, interdum eu, tincidunt sit amet, laoreet vitae, arcu. Aenean faucibus pede eu ante. Praesent enim elit, rutrum at, molestie non, nonummy vel, nisl. Ut lectus eros, malesuada sit amet, fermentum eu, sodales cursus, magna. Donec eu purus. Quisque vehicula, urna sed ultricies auctor, pede lorem egestas dui, et convallis elit erat sed nulla. Donec luctus. Curabitur et nunc. Aliquam dolor odio, commodo pretium, ultricies non, pharetra in, velit. Integer arcu est, nonummy in, fermentum faucibus, egestas vel, odio.

Chapitre 6

Conclusion

6.1 Résultat final

Lorem ipsum dolor sit amet, consectetur adipiscing elit. Ut purus elit, vestibulum ut, placerat ac, adipiscing vitae, felis. Curabitur dictum gravida mauris. Nam arcu libero, nonummy eget, consectetur id, vulputate a, magna. Donec vehicula augue eu neque. Pellentesque habitant morbi tristique senectus et netus et malesuada fames ac turpis egestas. Mauris ut leo. Cras viverra metus rhoncus sem. Nulla et lectus vestibulum urna fringilla ultrices. Phasellus eu tellus sit amet tortor gravida placerat. Integer sapien est, iaculis in, pretium quis, viverra ac, nunc. Praesent eget sem vel leo ultrices bibendum. Aenean faucibus. Morbi dolor nulla, malesuada eu, pulvinar at, mollis ac, nulla. Curabitur auctor semper nulla. Donec varius orci eget risus. Duis nibh mi, congue eu, accumsan eleifend, sagittis quis, diam. Duis eget orci sit amet orci dignissim rutrum.

6.2 Difficultés rencontrées

Nam dui ligula, fringilla a, euismod sodales, sollicitudin vel, wisi. Morbi auctor lorem non justo. Nam lacus libero, pretium at, lobortis vitae, ultricies et, tellus. Donec aliquet, tortor sed accumsan bibendum, erat ligula aliquet magna, vitae ornare odio metus a mi. Morbi ac orci et nisl hendrerit mollis. Suspendisse ut massa. Cras nec ante. Pellentesque a nulla. Cum sociis natoque penatibus et magnis dis parturient montes, nascetur ridiculus mus. Aliquam tincidunt urna. Nulla ullamcorper vestibulum turpis. Pellentesque cursus luctus mauris.

6.3 Objectifs non-réalisés

Nulla malesuada porttitor diam. Donec felis erat, congue non, volutpat at, tincidunt tristique, libero. Vivamus viverra fermentum felis. Donec nonummy pellentesque ante. Phasellus adipiscing semper elit. Proin fermentum massa ac quam. Sed diam turpis, molestie vitae, placerat a, molestie nec, leo. Maecenas lacinia. Nam ipsum ligula, eleifend at, accumsan nec, suscipit a, ipsum. Morbi blandit ligula feugiat magna. Nunc eleifend consequat lorem. Sed lacinia nulla vitae enim. Pellentesque tincidunt purus vel magna. Integer non enim. Praesent euismod nunc eu purus. Donec bibendum quam in tellus. Nullam cursus pulvinar lectus. Donec et mi. Nam vulputate metus eu enim. Vestibulum pellentesque felis eu massa.

6.4 Améliorations possibles du projet

Quisque ullamcorper placerat ipsum. Cras nibh. Morbi vel justo vitae lacus tincidunt ultrices. Lorem ipsum dolor sit amet, consectetur adipiscing elit. In hac habitasse platea dictumst. Integer tempus convallis augue. Etiam facilisis. Nunc elementum fermentum wisi. Aenean placerat. Ut imperdiet, enim sed gravida sollicitudin, felis odio placerat quam, ac pulvinar elit purus eget enim. Nunc vitae tortor. Proin tempus nibh sit amet nisl. Vivamus quis tortor vitae risus porta vehicula.

6.5 Conclusion personnelle

Fusce mauris. Vestibulum luctus nibh at lectus. Sed bibendum, nulla a faucibus semper, leo velit ultricies tellus, ac venenatis arcu wisi vel nisl. Vestibulum diam. Aliquam pellentesque, augue quis sagittis posuere, turpis lacus congue quam, in hendrerit risus eros eget felis. Maecenas eget erat in sapien mattis porttitor. Vestibulum porttitor. Nulla facilisi. Sed a turpis eu lacus commodo facilisis. Morbi fringilla, wisi in dignissim interdum, justo lectus sagittis dui, et vehicula libero dui cursus dui. Mauris tempor ligula sed lacus. Duis cursus enim ut augue. Cras ac magna. Cras nulla. Nulla egestas. Curabitur a leo. Quisque egestas wisi eget nunc. Nam feugiat lacus vel est. Curabitur consectetur.

Table des figures

2.1	Exemple de registre de transactions simplifié	4
2.2	Bob inscrit de fausse transactions dans le registre	4
2.3	Les transations du registre sont maintenant signées numériquement	5
2.4	Bob inscrit deux anciennes transactions d'Alice	6
2.5	Un identifiant unique est signé avec la transaction	6
2.6	Premières lignes indiquant la somme initiale donnée par chaque utilisateur . .	7
2.7	Charlie n'as pas assez d'argent pour effectuer la troisième transaction	7
3.1	Schéma simplifié de proof of history	14
4.1	Diagramme du protocole <i>proof of space</i> montrant les différentes étapes. Tiré de [chia:consensus]	22
4.2	Table simplifiée, tiré de [chia:construction]	23
4.3	Table améliorée, tiré de [chia:construction]	24