IMT Atlantique
Bretagne-Pays de la Loire
École Mines-Télécom

UFR de mathématique et d'informatique
Université de Strasbourg

Adaptation de l'algorithme de compression de structures de données Blockchain MLS à un système à population variable

Mémoire de stage

Master Science et Ingénierie des Réseaux, de l'Internet et des Systèmes

Nathanaël Derousseaux–Lebert
Tuteur : LUDINARD Romaric
Entreprise : IMT Atlantique
Février 2024 - Août 2024

Table des matières

Chapitre 1

Introduction

Dans le cadre de mon stage de fin d'études, j'ai travaillé au sein d'IMT Atlantique, une école de l'Institut Mines-Télécom, sur un projet de recherche visant à compresser les structures de données de type Blockchain. Le stage, d'une durée de 6 mois, a commencé le 5 février 2024 et se terminera le 2 août 2024. Il s'est déroulé au sein du département d'enseignement et de recherche "Systèmes Réseaux, Cybersécurité et Droit du numérique" (SRCD) d'IMT Atlantique, situé à Rennes.

Le projet de recherche sur lequel j'ai travaillé avait pour objectif de proposer une solution pour compresser les structures de données de type Blockchain, et ainsi faire face aux problématiques de scalabilité que ces structures posent. Une telle solution existait déjà, mais elle fonctionnait dans un cadre très particulier : le cas où la population du système reste constante. L'objectif était de généraliser cette solution pour qu'elle fonctionne dans un cadre plus large, utilisable dans le monde réel, où la population du système est susceptible varier. J'ai intégré une équipe de 3 chercheurs qui travaillaient déjà sur ce projet depuis quelques mois. J'ai apporté ma contribution à la solution proposée.

Mon mémoire de stage se décompose en trois chapitres. Le premier est la mise en contexte avec une présentation de mon environnement de travail, et du projet plus global dans lequel s'est inscrit mon stage. Le deuxième chapitre est une revue de l'état de l'art sur les systèmes Blockchain et sur l'algorithme de compression que nous avons utilisé. Enfin, le troisième chapitre est une présentation de mon travail, des problématiques que j'ai rencontrées, et des solutions que j'ai apportées.

Chapitre 2

Contexte

Mon stage de fin d'études s'est déroulé au sein d'IMT Atlantique, sur le campus de Rennes. Ce chapitre détaille le contexte global de mon stage en suivant un plan en entonnoir, allant de l'organisation générale d'IMT Atlantique jusqu'à mes missions spécifiques au sein du projet ANR *Towards Public Blockchain for Self-Sovereign Identities* [1], dans le cadre duquel mon stage s'est déroulé.

2.1 IMT Atlantique

IMT Atlantique, de l'Institut Mines-Télécom, est une école d'ingénieurs française issue de la fusion, en janvier 2017, de l'École nationale supérieure des mines de Nantes et de Télécom Bretagne. L'école possède trois campus situés à Brest, Nantes et Rennes. Elle propose des formations d'ingénieurs pluridisciplinaires couvrant les domaines des sciences, des technologies de l'information, de la communication, et des mathématiques. En termes de recherche, IMT Atlantique est structurée en douze départements d'enseignement et de recherche, regroupant des enseignants-chercheurs affiliés à différentes Unités Mixtes de Recherche (UMR) dont le GEPEA [1], l'IRISA [2], et le Lab-STICC [3]. Parmi ces douze départements, on trouve le département de Systèmes Réseaux, Cybersécurité et Droit du numérique (SRCD) sur le campus de Rennes, dans lequel j'ai effectué mon stage.

Le département SRCD est composé à ce jour de 27 enseignants-chercheurs permanents, plus de 10 post-doctorants et ingénieurs contractuels, 31 doctorants et 4 stagiaires. Les activités de recherche du département sont articulées autour de trois grands axes : les systèmes réseaux, la cybersécurité, et le droit du numérique. Les enseignants-chercheurs du département SRCD participent, avec des collègues d'autres institutions (*e.g.,* Université de Rennes), à 6 équipes de l'IRISA, dont l'équipe SOTERN [4] qui a accueilli mon stage.

L'équipe SOTERN, pour Self-prOtecting The futurE inteRNet, s'articule autour du deuxième pilier de recherche du département SRCD : la cybersécurité. C'est une équipe qui a été créée en début d'année 2023 et qui est dirigée par Guillaume Doyen, professeur à IMT Atlantique. L'équipe compte aujourd'hui 9 membres permanents, 1 post-doctorant, 6 doctorants et 5 sta-

1. https://www.gepea.fr
2. https://www.irisa.fr
3. https://labsticc.fr
4. https://www.irisa.fr/sotern/

giaires. Les recherches de cette équipe se concentrent sur la conception, le développement et la validation de méthodes et d'outils pour l'auto-protection de l'Internet futur. Le projet ANR à l'origine de mon stage s'inscrit dans le cadre des recherches menées par l'équipe SOTERN.

2.2 Projet BC4SSI

Mon tuteur, Romaric Ludinard, est le coordinateur du projet ANR intitulé BC4SSI [1] (*Towards Public Blockchain for Self-Sovereign Identities*). Ce projet s'inscrit dans l'axe E.3 /CES 25 de l'ANR, qui se concentre sur les verrous scientifiques relatifs aux réseaux de communication, architectures de réseau, et technologies logicielles, avec une attention particulière aux technologies Blockchain, et aux systèmes distribués. Plus spécifiquement, le projet BC4SSI à pour objectif de chercher des solutions aux verrous scientifiques et technologiques en lien avec les systèmes Blockchain, tels que décrit dans le rapport d'avril 2021 de la Direction Générale des Entreprises détaillant les principaux défis et enjeux de tels systèmes en France [2]. Mon travail s'est concentré sur le verrou numéro 2, qui concerne la compression de structures de données de type Blockchain. J'ai rejoint le projet en février 2024, c'est-à-dire 12 mois après le début de celui-ci. Quand je suis arrivé, un article de recherche était en cours de rédaction, et j'ai contribué à l'écriture des sections restantes.

2.3 Mes missions

Les deux à trois premiers mois de mon stage au sein de l'équipe SOTERN ont été consacrés à la réalisation d'un état de l'art. Cette phase initiale était essentielle pour plusieurs raisons. La première était de me familiariser avec le sujet complexe de la Blockchain : cette période d'analyse des publications m'a aidé à identifier les solutions actuellement explorées dans le domaine. La seconde raison était de commencer à appréhender les outils et les concepts utilisés dans notre article. La fin de cette phase documentaire a donc été un mélange entre la lecture de la littérature et la reconstruction des preuves mathématiques détaillées dans les articles les plus proches de notre sujet, afin d'acquérir une compréhension fine des outils dont j'allais avoir besoin pour la seconde partie de mon stage.

La seconde partie de mon stage a été consacrée à la rédaction de l'article de recherche. L'objectif était de proposer une solution pour compresser les structures de données de type Blockchain. Une telle solution existait déjà, mais elle fonctionnait dans un cadre très particulier : le cas où la population du système reste constante. Nous avons donc cherché à généraliser cette solution pour qu'elle fonctionne dans le cadre plus large d'un système à la population variable. J'ai donc participé à la fabrication de plusieurs modèles pour décrire notre problème et aux travaux mathématiques nécessaires pour démontrer la validité de nos propositions. J'ai également créé un programme de calcul qui implémente nos modèles afin d'obtenir des résultats numériques.

2.4 Environnement de travail

Au quotidien, j'ai travaillé avec les 4 personnes qui constituaient le pôle Blockchain de l'équipe SOTERN. Romaric Ludinard, mon tuteur, enseignant chercheur à IMT Atlantique, Loïc Miller, post-doctorant à IMT Atlantique également, Dorian Pacaud, Stagiaire qui a commencé 1 mois

après moi, et Emmanuelle Anceaume, Directrice de Recherche au CNRS, membre de l'équipe PIRAT\');[5] de l'IRISA.

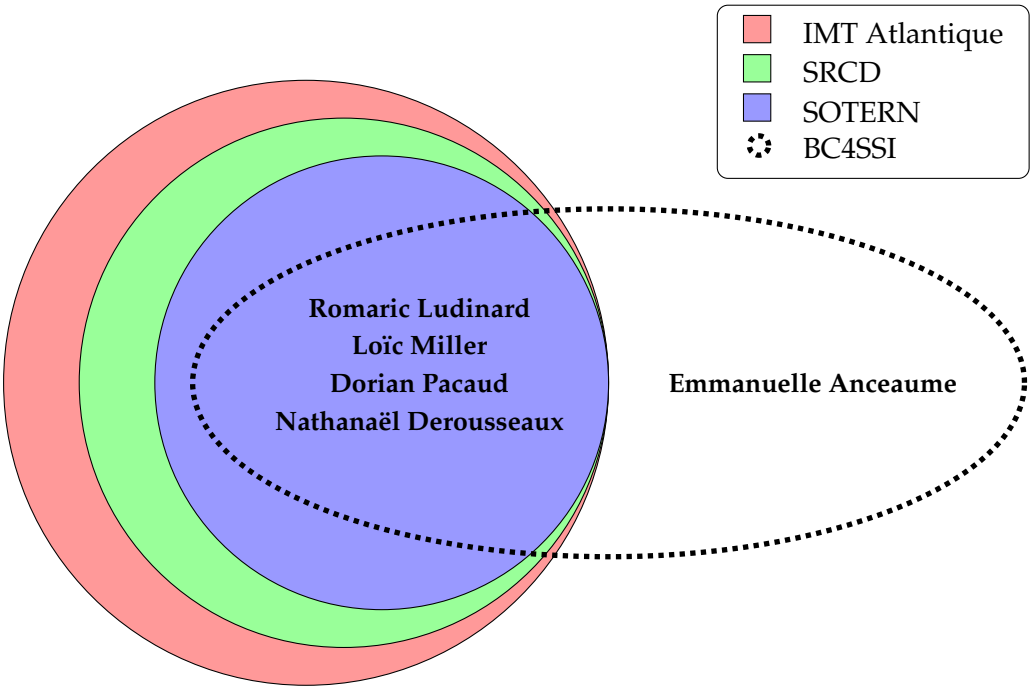

FIGURE 2.1 – Organigramme de mon environnement de travail

Pour garantir une bonne coordination, nous nous retrouvions deux fois par semaine pour discuter de nos avancées, des difficultés rencontrées, et travailler ensemble. Durant ma phase documentaire, ces réunions étaient surtout l'occasion pour moi d'obtenir des éclaircissements sur les articles que j'avais lus, et de me donner des pistes pour continuer mon travail. Ensuite, durant la phase de recherche, ces réunions ont permis d'avancer sur la rédaction de l'article, en réfléchissant ensemble à comment résoudre les problèmes que nous rencontrions. Le reste du temps, je travaillais en autonomie, partant des discussions de nos réunions pour avancer sur mes tâches.

5. https://www.inria.fr/pirat

Chapitre 3

État de l'art

La première partie de mon travail a consisté à réaliser un état de l'art sur les systèmes Blockchain. Le plan de ce chapitre suivra la chronologie de mes recherches. J'ai commencé par approfondir mes connaissances sur le fonctionnement général des Blockchains [3], avant de m'intéresser aux modèles utilisés pour décrire l'évolution de tels systèmes ([4] et [5]). J'ai ensuite étudié un algorithme de compression dans un modèle dit *statique* [6]. Enfin, j'ai étudié le travail déjà effectué par l'équipe avant mon arrivée. L'objectif de ce travail étant d'adapter l'algorithme de compression à un modèle *dynamique*, plus proche de la réalité.

En plus de la lecture approfondie des articles auxquels je fais référence, j'ai refait de nombreuses preuves et calculs des travaux présentés dans les sections 3.3 et 3.4. Cela m'a permis de me familiariser avec les outils et les concepts qui allaient m'être nécessaires pour la suite de mon travail de recherche.

J'ai pris la liberté de renommer certaines variables et notations présentes dans les articles, afin d'homogénéiser les notations entre elles, et de rendre ma présentation plus claire. Une liste des notations relatives à chaque section est disponible en annexe A.

3.1 Les systèmes Blockchain

Ma première lecture a été évidemment le livre blanc de Satoshi Nakamoto [3]. Ce document est la première publication sur le sujet et reste une référence incontournable. La Blockchain y est décrite comme une solution de paiement décentralisée, sans tiers de confiance. La sécurité du système y est assurée par un historique partagé de toutes les transactions. Il est ainsi auditable par tous les participants. La sécurité de l'historique est assurée par un mécanisme de preuve de travail (abrégée PoW pour Proof Of Work), qui garantit son intégrité.

Quand un utilisateur (Bob) de la Blockchain veut effectuer un paiement auprès d'un autre utilisateur (Alice), il crée une transaction. Bob signe cette transaction avec sa clé privée, puis la diffuse sur le réseau. Les autres participants peuvent ainsi vérifier que Bob est bien en capacité de dépenser les fonds qu'il veut envoyer, et qu'il ne se dédit pas. Pour cela, ils vérifient la signature de Bob, et qu'il n'a pas déjà dépensé ces fonds.

La transaction est ensuite ajoutée à l'historique des transactions. Pour savoir si Bob est en capacité d'utiliser son argent, on parcourt l'historique des transactions en partant de la création de

la Blockchain. On cherche alors toutes les transactions qui lui sont destinées, mais qu'il n'a pas encore dépensées. On appelle ces transactions des UTXOs. Bob peut alors présenter ces UTXOs pour prouver qu'il est en capacité de dépenser ces fonds.

D'une manière générale, une Blockchain est une structure de données. Au cours de mon travail, je n'ai pas eu besoin de m'intéresser aux données stockées, mais uniquement à la structure de la Blockchain. C'est pourquoi dans le reste de ce mémoire, je parlerai de données applicatives, sans préciser leur nature.

3.1.1 Structure de la chaîne

Chaque changement d'état des données applicatives est stocké dans des blocs. Chaque nœud de la Blockchain stocke une copie de la chaîne de blocs. Quand un nœud reçoit un changement des données applicatives, il stocke ce changement dans un bloc à la fin de sa chaîne. Il diffuse ensuite ce bloc sur le réseau. Les autres nœuds vérifient la validité du bloc, puis l'ajoutent à leur chaîne. La chaîne la plus difficile à créer est considérée comme la chaîne de référence. Chaque nœud du système peut posséder une chaîne légèrement différente des autres. En effet le réseau subit un temps de propagation. Un bloc en fin de chaîne peut donc se faire invalider par une autre chaîne ayant demandé plus de travail. Cette chaîne n'aurait pas encore été reçue par notre nœud initial. Les derniers blocs d'une chaîne sont donc instables. Malgré cette instabilité des derniers blocs, chaque chaîne du réseau possède un préfixe commun immuable.

Un bloc est une structure de données que l'on peut représenter par un tuple (h, d), où h est le header du bloc (ses méta-données) et d constitue les données applicatives. Le header contient plusieurs informations importantes, le timestamp de création du bloc, le numéro de version du protocole, etc. Mais surtout, il contient le hash ($\hbar$) du bloc précédent. Ainsi chaque bloc est lié au précédent, formant une chaîne de blocs. Un adversaire qui voudrait modifier un bloc, devrait recalculer tous les blocs suivants, puisque cela modifierait le hash du bloc modifié, qui modifierait celui du suivant, et ainsi de suite jusqu'à la fin de la chaîne. Pour s'assurer qu'un tel recalcul soit impossible, on va utiliser un mécanisme rendant la tâche de création d'un bloc difficile : la preuve de travail (PoW).

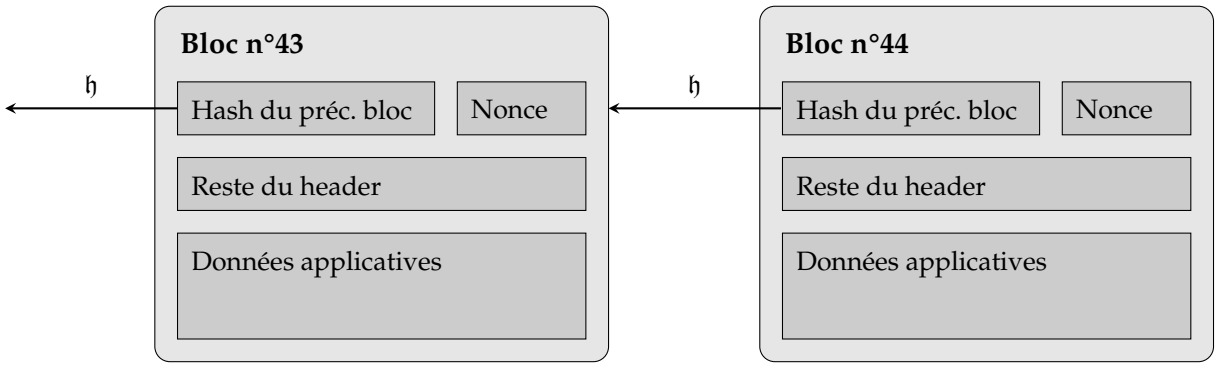

FIGURE 3.1 – Structure des blocs dans une Blockchain

3.1.2 Preuve de travail

Pour rendre la création d'un bloc difficile, il existe plusieurs mécanismes. Le plus connu est la preuve de travail (PoW), mais il en existe d'autres, comme la preuve d'enjeu (PoS) [7], ou la preuve de capacité (PoC) [8]. L'idée générale est de demander au créateur du bloc (appelé mineur) de résoudre un problème difficile. En revanche, la vérification de la solution doit être facile, pour que les autres participants puissent facilement vérifier la validité du bloc. Dans le cadre de mon stage, je me suis intéressé à la preuve de travail, qui est le mécanisme utilisé par Bitcoin.

La preuve de travail consiste à trouver à trouver un nonce (noté c), tel que :

$$\mathfrak{h}(c, h, d) < T \tag{3.1}$$

Avec $\mathfrak{h}$ la fonction de hashage, h le header du bloc (contenant le hash du bloc précédent), d les données applicatives, et T la target, définie par le protocole. Ainsi, il faudra tester tout les entiers c jusqu'à trouver un nonce qui vérifie cette équation. Plus T sera petit, plus il sera difficile de trouver un tel c, inversement, plus T sera grand, plus il sera facile de le trouver. On définit la difficulté comme l'inverse de la target $D = 1/T$.

Afin de garantir que la création d'un bloc soit difficile, la difficulté est ajustée régulièrement. En effet, si le réseau au global augmente sa puissance de calcul, la difficulté de création d'un bloc risque de devenir trop faible et de permettre à un attaquant de créer des blocs plus vite que le reste du réseau. Pour éviter cela, la difficulté est ajustée par le protocole tous les 2016 blocs. Cet ajustement est paramétré de manière à ce que le temps moyen de création d'un bloc reste d'environ 10 minutes.

Malgré ce mécanisme de preuve de travail, il est toujours possible que l'adversaire soit "chanceux" et trouve le nonce plus vite que les autres parties honnêtes du réseau. Cependant, statistiquement, il ne peut pas être "chanceux" à chaque fois. Le calcul de la probabilité de réussir à invalider un bloc à k profondeur a été fait dans l'article de Nakamoto [3], en modélisant le problème par un processus de Poisson :

$$1 - \sum_{i=0}^{k} \frac{\lambda^i e^{-\lambda}}{i!}(1 - (q/p)^{k-i}) \tag{3.2}$$

Avec respectivement q et p les proportions de puissances de calcul de l'adversaire et de l'honnête (telle que $q + p = 1$) et λ le taux de création de blocs ($\lambda = k(q/p)$). Ce calcul a permis d'estimer qu'avec $1/3$ de la puissance de calcul, l'adversaire a une probabilité de $P = 10^{-6}$ de réussir à invalider un bloc de profondeur 50.

3.2 Le modèle statique

Il existe un certain nombre de modèles pour décrire le fonctionnement des systèmes Blockchain. J'ai commencé par étudier un modèle statique [4] où la puissance de calcul de l'ensemble de nœud est fixée. Ce modèle est plus simple à étudier, mais reflétera moins la réalité qu'un modèle dynamique, où la puissance de calcul du réseau peut varier.

3.2.1 Présentation du modèle

Le modèle statique appréhende les systèmes Blockchain comme un ensemble de n nœuds dont la puissance de calcul est fixée. Parmi ces n nœuds, il y a $n - t$ nœuds honnêtes et t nœuds malveillants. Les nœuds honnêtes suivent le protocole à la lettre, tandis que les malveillants peuvent tenter de le violer. Ce modèle se place dans une hypothèse synchrone, où le temps s'écoule en rondes. Par ailleurs, les canaux de communications sont considérés comme fiables, c'est à dire qu'il n'y a ni perte, ni altération ni duplication des messages émis. Ainsi, ces hypothèses assurent que tous les messages émis au cours d'une ronde sont reçus par leurs destinataires avant la fin de celle-ci. Qu'il soit malveillant ou honnête, chaque nœud ne peut faire qu'un nombre fixé r requêtes par ronde à une fonction de hashage pour tenter de miner un bloc.

Probabilité de réussite On note P_T la probabilité d'un unique nœud honnête de réussir à trouver un bloc en une seule requête à la fonction de hashage. On note κ le nombre de bits de la sortie de la fonction de hashage. T est la target utilisée pour la preuve de travail. Étant donnée que la fonction de hashage est indistinguable d'un processus uniforme, la probabilité de trouver un bloc en une seule requête est $P_T = T/2^\kappa$. On note f la probabilité qu'au moins un nœud honnête trouve un bloc en une ronde. Cela revient à calculer l'espérance de la loi binomiale de paramètre r et P_T, soit :

$$f = 1 - (1 - P_T)^{r(n-t)} \tag{3.3}$$

À partir de f, les auteurs introduisent trois variables aléatoires X, Y et Z. $X_i = 1$ si au moins un nœud honnête trouve un bloc à la ronde i, $X_i = 0$ sinon. $Y_i = 1$ si exactement un nœud honnête trouve un bloc à la ronde i, $Y_i = 0$ sinon. $Z_i = 1$ si un nœud malveillant trouve un bloc à la ronde i, $Z_i = 0$ sinon. L'espérance et les variations de X, Y et Z sont facilement calculables puisqu'il s'agit de variables binomiales.

Exécution typique Le modèle introduit ensuite le concept d'exécution typique. Une exécution sera dite typique si les variables aléatoires X, Y et Z ne s'écartent pas trop de leur espérance, au plus d'un facteur $\varepsilon \in (0, 1)$. Une exécution peut être dite typique uniquement si le nombre rondes considérées est suffisamment grand. On note ce paramètre $\Lambda \geq 2/f$. Pour finir, on partira du principe que la fonction de hashage est inviolable, c'est à dire qu'on omettra les cas de collisions. Les auteurs prouvent que la probabilité d'une exécution typique vaut $1 - e^{O(\varepsilon^2 \Lambda f + \kappa - \log L)}$, avec L le nombre de rondes depuis le début du protocole.

3.2.2 Propriétés du modèle

Avec ces outils, les auteurs du modèle ont pu prouver deux propriétés fondamentales du protocole Bitcoin si on reste dans le cadre d'une exécution typique. Ces deux propriétés sont le préfixe commun et la qualité de la chaîne. Le préfixe commun assure que les chaînes des nœuds honnêtes ont un préfixe commun important, tandis que la qualité de la chaîne assure que la proportion de blocs malveillants dans la chaîne des nœuds honnêtes est limitée.

Préfixe commun Le préfixe commun est garanti si la proportion de nœuds malveillants est inférieure à celle des nœuds honnêtes. Dans une exécution typique, si $t < n - t$, alors les chaînes des nœuds honnêtes ont un nombre de blocs k à tronquer de la chaîne originale pour obtenir un préfixe commun qui est $k \geq 2\Lambda f$.

Qualité de la chaîne La qualité de la chaîne définit à quel point la chaîne des nœuds honnêtes est polluée par des blocs malveillants. La qualité de la chaîne sera idéale si la proportion de blocs appartenant aux malveillants est exactement égale à leur proportion de puissance de calcul. Les auteurs ont prouvé que dans une exécution typique, la qualité de la chaîne t/n est garantie avec une probabilité écrasante si $t/n < 1/3$.

3.3 Le modèle dynamique

Dans un autre article [5] les auteurs proposent une extension dynamique du modèle statique [4]. Ce modèle introduit la variation de la puissance de calcul du système au cours du temps. Il reprend les mêmes concepts que le modèle statique, mais rajoute un certain nombre d'outils mathématiques pour prendre en compte cette variation.

3.3.1 Présentation du modèle

La principale différence entre le modèle statique et le modèle dynamique est que le nombre de nœuds n n'est plus fixé. Or, pour maintenir un rythme de création de blocs constant, le protocole doit ajuster la difficulté de création de blocs. Ainsi, si le nombre de nœuds augmente, la difficulté doit augmenter, inversement si le nombre de nœuds diminue. Donc la difficulté devient variable.

Fonction de recalcul de la target Pour ajuster la difficulté, le protocole doit recalculer la target T tous les J blocs ($J = 2016$ blocs dans le cas de Bitcoin). On nomme cet intervalle entre deux recalculs une *epoch*. Les auteurs introduisent la fonction $D(\cdot)$ de recalcul de la difficulté. Cette fonction va être exécutée par chaque nœud à la fin de chaque epoch, pour définir la nouvelle target T'. La variation de la difficulté entre deux epochs est bornée par un facteur τ ($\tau = 4$ dans le cas de Bitcoin). C'est à dire que $T/\tau \leq T' \leq T \cdot \tau$. Cela permet d'éviter des variations trop brutales de la difficulté, ce qu'un attaquant pourrait exploiter.

Exécutions (η, θ)-good La fonction $D(\cdot)$ est exécutée par chaque nœud à la fin de chaque epoch. Sauf que chaque nœud ne dispose pas de la même information. En effet, le réseau subit un temps de propagation, et chaque nœud ne reçoit pas les mêmes blocs exactement au même moment. Il est donc légitime pour deux nœuds honnêtes de ne pas avoir exactement la même target T. Les auteurs introduisent donc le concept d'exécution (η, θ)-good, qui assure que le taux de production des blocs honnêtes ne s'écartent pas trop du taux de production théorique attendu par le protocole. Une exécution sera dite (η, θ)-good si :

$$\eta f \geq f(T, n) \geq \theta f \tag{3.4}$$

Avec f la nouvelle target calculée par le nœud, T l'ancienne target, et n le nombre de rondes considérées depuis la dernière epoch.

Environnement (γ, s)-respecting Un environnement sera dit (γ, s)-respecting si la variation du nombre de nœuds est bornée d'un facteur γ toutes les s rondes. Cela assure que la variation du nombre de nœuds n'est pas trop brutale. En effet, il est raisonnable de supposer que dans la réalité, le nombre de nœuds ne varie pas trop vite.

Variables aléatoires Les auteurs modifient ensuite la définition des variables aléatoires X, Y et Z introduites dans le modèle statique. Avec une target dynamique, il ne suffit plus de considérer un bloc crée ou non, mais aussi de prendre en compte la target utilisée pour le créer. En effet, deux blocs créés avec des targets différentes n'ont pas la même probabilité d'être créés, ni la même importance lors de la comparaison entre deux chaînes. Les auteurs modifient donc les variables X, Y et Z pour valoir 0 si aucun bloc est créé, et D si un bloc est créé avec une difficulté D.

3.3.2 Propriétés du modèle

À partir de ces nouveaux outils, les auteurs ont pu prouver deux autres propriétés importantes du protocole Bitcoin : la *persistance* et la *vivacité*. La *persistance* assure qu'une donnée une fois inscrite dans la chaîne est immuable, tandis que la *vivacité* assure qu'une donnée valide sera inscrite dans la chaîne.

Persistance La *persistance* assure qu'une donnée une fois inscrite dans la chaîne est immuable. Dans une exécution typique, et un environnement (γ, s)-respecting, la *persistance* est garantie quand la donnée est inscrite dans la chaîne à une profondeur k telle que :

$$k \geq \frac{\theta \gamma J}{4\tau} \tag{3.5}$$

Vivacité La *vivacité* assure qu'une donnée valide sera inscrite dans la chaîne. Les auteurs ont prouvé que dans une exécution typique, et un environnement (γ, s)-respecting, la *vivacité* est garantie avec une probabilité écrasante en un temps fini.

3.4 Algorithme de compression MLS

Une Blockchain peut vite devenir très volumineuse. En effet, c'est une structure de données en ajout uniquement. Ainsi, aujourd'hui, la chaîne Bitcoin pèse plus de 450 Go après 15 ans d'existence. La chaîne Ethereum pèse quant à elle plus de 1 To après seulement 8 ans d'existence. De plus la croissance de telles chaînes est linéaire. Cela pose un problème d'accès à un tel réseau : si un nœud doit stocker l'intégralité de la chaîne, il lui faudra un espace de stockage important, et tous les appareils ne disposent pas d'un tel espace de stockage. C'est pourquoi il est important de trouver des solutions pour compresser la chaîne.

Il existe aujourd'hui des solutions de compression et elles comportent toutes des compromis. J'ai étudié l'une d'entre elles, l'algorithme de compression *Mining in Logarithmic Space* (MLS) [6]. MLS est un algorithme de compression de la chaîne de bloc. L'article s'appuie sur les travaux du modèle statique [4], c'est pourquoi il était important pour moi de travailler sur ce modèle avant de m'intéresser à MLS.

3.4.1 Présentation de l'algorithme

MLS fait partie de la famille des algorithmes NIPoPoWs (Non-Interactive Proofs of Proof of Work). Ces algorithmes permettent de prouver que les preuves de travail ont été effectuées

sans avoir à effectivement posséder les blocs. L'algorithme MLS va échantillonner les blocs de la chaîne, et ne garder que les blocs les plus importants. Cependant, en échantillonnant les blocs, des données applicatives vont être perdues : on partira donc du principe que la chaîne que l'on va compresser effectue des snapshots de l'état applicatif à chaque bloc.

Deux algorithmes seront nécessaires pour utiliser MLS : un algorithme de compression et un algorithme de comparaison, permettant de comparer deux compressés. Les deux algorithmes sont disponibles en annexe B.1.

Niveau d'un bloc Les auteurs introduisent le concept de niveau d'un bloc. Le niveau ℓ d'un bloc est défini comme le nombre de fois ou la target a été dépassée. Plus formellement :

$$\text{Le bloc } b \text{ est de niveau } \ell \Leftrightarrow \mathfrak{h}(b) \leq \frac{T}{2^\ell} \tag{3.6}$$

Ainsi, un bloc de niveau ℓ est un bloc qui a dépassé la target de 2^ℓ Étant donnée que la sortie de $\mathfrak{h}$ est uniforme, la probabilité qu'un bloc soit de niveau ℓ est $2^{-\ell}$. Donc tous les blocs seront de niveau 0, la moitié de niveau 1, le quart de niveau 2, etc.

Algorithme de compression L'algorithme MLS va échantillonner les blocs de la chaîne. Pour cela, il va d'abord chercher le niveau dit *maximal* de la chaîne. Le niveau maximal est le niveau le plus haut qui possède au moins $2m$ bloc, m étant un paramètre du système. On garde tous les blocs du niveau maximal. Ensuite, l'algorithme garde les $2m$ derniers blocs de chaque niveau inférieur au niveau maximal, et les m derniers blocs du niveau supérieur.

Chainage des blocs Il est important de garder le bloc précédent dans la chaîne compressée pour pouvoir empêcher un adversaire de modifier un bloc au milieu de la chaîne. Cependant, comme l'algorithme ne garde pas tous les blocs, il est possible que le bloc précédent dans la chaîne initiale ne soit plus dans la chaîne compressée. Les auteurs propose de changer la structure de la Blockchain d'une liste simplement chaînée vers l'arrière à une liste à enjambement (*skiplist*). Ainsi, chaque bloc contiendra le hash du dernier bloc de chaque niveau. On s'assure ainsi qu'on a toujours au moins une référence à un bloc qui sera contenu dans la chaîne compressée.

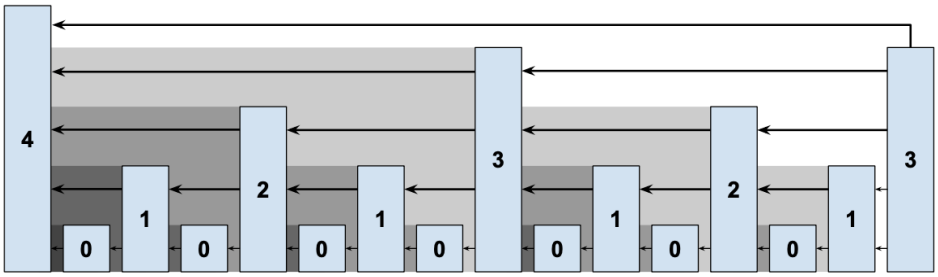

FIGURE 3.2 – Chainage des blocs dans MLS [6]

Algorithme de comparaison Les nœuds doivent être capables de comparer deux chaînes compressées pour choisir laquelle garder. L'algorithme de comparaison va chercher le dernier ancêtre commun de même niveau dans les deux compressés, et comparer les blocs à partir de ce point. Étant donné que l'algorithme de compression a gardé les blocs les plus *rares*, le compressé ayant le plus de blocs de haut niveau sera celui qui témoigne d'une chaîne ayant demandé plus de travail avant la compression.

3.4.2 Propriétés de l'algorithme

À l'aide des outils du modèle statique [4], les auteurs ont pu prouver que l'algorithme MLS respecte trois propriétés importantes : la *sécurité*, la *concision* et l'*idempotence*.

Sécurité La propriété de *sécurité* assure que lors de la comparaison de deux chaînes compressées, l'algorithme de compression choisira le compressé honnête. Cette propriété a été prouvée dans le cas où l'adversaire possède moins d'un tiers du réseau, et que l'on est dans le cadre d'une exécution typique.

Concision La propriété de *concision* assure que la taille de la chaîne compressée grandit de manière logarithmique par rapport à la chaîne originale $\mathcal{C}$. Il a été prouvé que le compressé est de taille $2m \log |\mathcal{C}| + k$ avec $|\mathcal{C}|$ le nombre de blocs de la chaîne originale, m le e paramètre de l'algorithme de compression, et k les derniers blocs de la chaîne originale, que l'on ne compresse pas car ils sont instables.

Idempotence La propriété d'*idempotence* assure que compresser une chaîne à laquelle on aurait préalablement ajouté un bloc est équivalent à compresser un compressé de cette chaîne à laquelle on aurait ajouté un bloc. Cette dernière propriété est importante pour garantir que l'on ne supprime pas des blocs dont on aurait besoin plus tard. La propriété d'*idempotence* assure également que l'on puisse continuer de miner des blocs sur la chaîne compressée.

3.5 MLS dans un modèle dynamique

L'algorithme de compression MLS [6] a été conçu dans un modèle statique. Cependant, dans la réalité, le nombre de nœuds n'est pas fixé, et la difficulté de création de blocs doit être ajustée régulièrement. C'est pourquoi l'équipe de BC4SSI a travaillé sur une adaptation de l'algorithme de compression MLS à un modèle dynamique, plus proche de la réalité.

La principale différence entre l'algorithme de compression MLS dans un modèle statique et un modèle dynamique est la variation de la difficulté. Cela a nécessité de modifier l'algorithme de compression et de comparaison pour prendre en compte cette variation (disponibles en annexe B.2). En effet, à présent, les difficultés des blocs sont prises en compte dans la comparaison des chaînes compressées.

Les preuves ont aussi du être adaptées pour prendre en compte ce modèle dynamique. Ainsi, l'équipe a pu prouver qu'il était toujours impossible de supprimer un bloc de la chaîne compressée et que les propriétés de *sécurité*, de *concision* et d'*idempotence* étaient toujours respectées dans la version dynamique de l'algorithme. Il a aussi été prouvé que la propriété du préfixe commun était toujours respectée en considérant les blocs du même niveau. C'est à dire que deux chaînes partagent un préfixe commun important de blocs de même niveau.

Valeur de m La valeur de m est un paramètre important de l'algorithme de compression MLS. Il définit le nombre de blocs de chaque niveau que l'on garde dans la chaîne compressée. Choisi trop petit, l'adversaire risque de pouvoir créer une chaîne compressée plus difficile que la chaîne honnête par "chance". Choisi trop grand, la chaîne compressée deviendra trop volumineuse. Les auteurs de l'algorithme du modèle statique n'ont pas exprimé de valeur de m. Il s'agit pourtant d'un paramètre important pour utiliser l'algorithme dans la réalité. C'est pourquoi l'objectif de notre article est de proposer une valeur de m pour l'algorithme de compres-

sion MLS dans un modèle dynamique. Ça sera l'objet de la suite de mon travail de recherche.

Chapitre 4

Réalisations

Dans ce chapitre, je détaille la seconde partie de mon travail de recherche : mes réalisations. En effet, ma mission était d'aider à finir l'article en cours d'écriture, dont j'ai présenté l'existant dans la section 3.5. Ce chapitre présentera donc les travaux que j'ai réalisés pour aider à la finition de cet article.

La première de mes réalisations a été de calculer la probabilité de créer un bloc pour un adversaire, sachant que celui-ci peut tricher sur la target T qu'il choisit. On a ensuite cherché à déterminer une valeur de m qui permettrait de garantir les propriétés souhaitées pour l'algorithme MLS. Pour cela, on a exploré deux méthodes : une méthode par marche aléatoire, et une méthode par la ruine du joueur. La méthode par marche aléatoire n'a rien donné, le problème n'étant pas résolu à l'heure actuelle par la communauté scientifique. En revanche, la méthode par la ruine du joueur a fourni des outils pour déterminer la valeur de m en fonction de la quantité de triche de l'adversaire. Enfin, dans la dernière section de ce chapitre, je parlerai des activités que j'ai réalisées durant mon stage, en marge de ce sujet.

4.1 Probabilité de créer un bloc

Mon premier travail a été de calculer la probabilité qu'un bloc appartienne à un adversaire. En effet, on aura besoin de cette probabilité pour calculer la valeur de m par la suite.

Dans une même ronde, on a quatre situations possibles : aucun bloc n'est créé, seul l'honnête crée un bloc, seul l'adversaire crée un bloc, ou les deux créent un bloc. On cherche la probabilité μ qu'un seul bloc soit créé, et que ce bloc appartienne à l'adversaire. On considère uniquement le cas où seul l'adversaire crée un bloc, car c'est le cas le plus intéressant pour l'adversaire. En effet, si les deux parties créent un bloc en même temps, on est dans une situation de fork et ce sont les blocs futurs qui vont décider de laquelle des deux versions de la chaîne sera acceptée. Il est donc plus intéressant pour l'adversaire de créer un bloc seul, pour être sûr que son bloc soit accepté.

Il est légitime que deux nœuds aient des targets différentes jusqu'à un facteur α tel que $\alpha T_m = T_b$. Ainsi, si $\alpha = 1/4$, la target de l'adversaire est 4 fois plus faible que celle de l'honnête, il est donc 4 fois plus difficile pour l'adversaire de créer un bloc. Inversement, si $\alpha = 4$, la target de l'adversaire est 4 fois plus grande que celle de l'honnête, il est donc 4 fois plus facile pour l'adversaire de créer un bloc.

On nomme N la variable aléatoire qui représente le nombre de blocs créés durant une ronde. On appelle B l'événement "un bloc de la ronde appartient à Bob" (Bob est l'honnête), M l'événement "un bloc de la ronde appartient à Mallory" (Mallory est l'adversaire). Calculer la probabilité $\mu(\alpha)$ qu'un bloc appartienne à Mallory revient à se demander : sachant qu'un seul bloc a été crée durant la ronde, quelle est la probabilité que ce bloc appartienne à Mallory ? Plus formellement, on cherche à calculer $\mu(\alpha) = P(M|N = 1)$. De manière assez intuitive, on peut dire que la probabilité que ce bloc appartienne à Bob vaut $1 - \mu(\alpha) = P(B|N = 1)$.

On note P_b la probabilité pour un unique honnête de créer un bloc en une seule requête à la fonction de hash, P_m la probabilité pour un unique adversaire de créer un bloc en une seule requête à la fonction de hash, r le nombre de requêtes à la fonction de hash par ronde, n le nombre de nœuds au total, et t le nombre de nœuds adversariaux. On a donc, à partir de l'équation 3.3 du modèle :

$$P(B) = 1 - (1 - P_b)^{r(n-t)} \tag{4.1}$$

$$P(M) = 1 - (1 - P_m)^{rt} \tag{4.2}$$

$$P(N = 1) = P(B \wedge \neg M) + P(\neg B \wedge M) \tag{4.3}$$

Avec la formule des probabilités conditionnelles :

$$P(M|N = 1) = \frac{P(M \wedge \neg B)}{P(N = 1)} = \frac{(1 - (1 - \frac{T_m}{2^\kappa})^{rt})(1 - \frac{T_b}{2^\kappa})^{r(n-t)}}{(1 - \frac{T_b}{2^\kappa})^{r(n-t)} + (1 - \frac{T_m}{2^\kappa})^{rt} - 2(1 - \frac{T_b}{2^\kappa})^{r(n-t)}(1 - \frac{T_m}{2^\kappa})^{rt}} \tag{4.4}$$

Avec T_b la target choisie par l'honnête, et T_m la target choisie par l'adversaire.

Simplification du calcul de $\mu(\alpha)$ Plus tard durant mon stage, je me suis rendu compte que l'on pouvait approcher $\mu(\alpha)$ de manière plus simple. En effet, si on ne prend pas en compte les collisions (c'est à dire les moments ou les deux joueurs trouvent un bloc à la même ronde), il suffit de considérer que si l'adversaire mine un bloc 4 fois plus difficile ($\alpha = 4$), il le fera 4 fois moins souvent. On a donc $\mu(\alpha) = \alpha t/((n - t) + \alpha t)$. Dans les fait, les modèles statique [4] et dynamique [5] nous apprennent que les collisions sont extrêmement rare, on peut donc utiliser cette approximation sans problème dans la plupart des cas. Malgré tout, dans le reste de ce mémoire, toutes les applications numériques ont été faites avec la formule complète de $\mu(\alpha)$, la formule 4.4.

Le protocole borne la dérive entre les targets des participants d'un facteur $\alpha \in [1/4, 4]$, ce qui implique l'adversaire ne peut pas tricher sur sa target plus que d'un facteur 4. En effet, si l'adversaire triche trop, il sera immédiatement repéré par les autres nœuds, et le bloc ne sera pas accepté. On a donc $T_m/4 \geq T_b \geq 4T_m$.

On a donc effectué l'application numérique pour les trois valeurs de α : $\alpha = 1/4$, $\alpha = 1$, et $\alpha = 4$, avec un adversaire à $1/3$ de la puissance de calcul ($n = t/3$), puisqu'il s'agit du cas le plus critique pour le réseau, tel que décrit dans l'article [5].

Les résultats obtenus sont décrits dans le tableau 4.1.

De manière assez intuitive, on peut voir que plus l'adversaire a une target grande (donc facile), plus il a de chance de créer un bloc. Quand il triche au maximum ($\alpha = 4$), $2/3$ des blocs créés seront des blocs adversariaux, contre seulement $1/3$ de blocs honnêtes. Cependant, comme les

α	$\mu(\alpha)$	$1-\mu(\alpha)$
1/4	1/9	8/9
1	1/3	2/3
4	2/3	1/3

TABLE 4.1 – $\mu(\alpha)$ en fonction de α avec $q = 1/3$

protocoles Bitcoin et MLS ne comparent pas le nombre de blocs entre deux chaînes, mais la somme de travail fourni, l'adversaire aura peut-être plus de blocs, mais ils auront moins de "valeur" que les blocs honnêtes. Il est donc important de se demander quelle serait la meilleure stratégie pour l'adversaire : plus de blocs de moins grande valeur, ou moins de blocs de plus grande valeur ? C'est une question qu'il faudra prendre en compte dans les calcul de la valeur de m, et fixer sa valeur de manière à ce que l'algorithme résiste à l'attaque de l'adversaire, même dans le pire des cas.

4.2 Recherche de m par marche aléatoire

Dans l'algorithme MLS, la valeur de m est un paramètre important. C'est le paramètre qui va définir le nombre de blocs de chaque niveau de la chaîne. Il doit être suffisamment grand pour être sur que l'adversaire ne puisse pas rattraper la chaîne honnête, mais pas trop grand pour ne pas créer un compressé trop grand. De plus, étant donné que l'adversaire peut tricher sur la target, on va trouver plusieurs valeurs de m fonction de la quantité de triche α de l'adversaire, on prendra donc le m maximal parmi ces valeurs.

On a donc cherché à déterminer les valeurs de m qui permettraient de garantir les propriétés souhaitées pour l'algorithme MLS. Pour commencer, on a essayé de modéliser la création de blocs par une marche aléatoire non isotrope en deux dimensions.

4.2.1 Construction du modèle

En prenant appui sur l'article [9], qui travaillait sur un problème similaire, on a travaillé sur la création d'un modèle permettant de simuler la création des blocs.

On considère une succession de rondes où l'honnête et l'adversaire jouent chacun de leur côté. On s'intéresse au nombre de victoires de chaque joueur (Bob et Mallory) après k victoires au total. On note $V_k = (A_b, A_m)$ l'état après k victoires, où A_b et A_m sont respectivement le nombre de victoires de Bob et de Mallory. A partir de l'état $V_k = (A_b, A_m)$, seules deux transitions sont possibles : soit Bob gagne la ronde et on a $V_{k+1} = (A_b + 1, A_m)$, soit Mallory gagne la ronde et on a $V_{k+1} = (A_b, A_m + 1)$. On note $\mu(\alpha)$ la probabilité que Mallory gagne la ronde, et $1 - \mu(\alpha)$ la probabilité que Bob gagne la ronde. On ne s'intéresse pas aux rondes où aucun bloc n'est créé, car elles n'ont pas d'impact sur la chaîne.

L'espace de V_k est donc un quart de plan : si Bob gagne la ronde, on va vers la droite, si Mallory gagne la ronde, on va vers le haut. On va tracer une droite délimitant l'espace de V_k en deux parties : l'espace $\mathcal{B}$ où la chaîne de Bob a plus de difficulté cumulée que la chaîne de Mallory, et l'espace $\mathcal{M}$ où la chaîne de Mallory a plus de difficultés que la chaîne de Bob. Cet espace est délimité par la droite $A_b - \alpha A_m = 0$, de telle manière que :

$$\mathcal{M} = \{(b,m) \in \mathbb{N}^2 \mid A_b - \alpha A_m >= 0\} \tag{4.5}$$

$$\mathcal{B} = \{(b,m) \in \mathbb{N}^2 \mid A_b - \alpha A_m < 0\} \tag{4.6}$$

Dans cette définition $V_k \in \mathcal{B}$ signifie que la chaîne de Bob a plus de difficulté que la chaîne de Mallory, et $V_k \in \mathcal{M}$ signifie que la chaîne de Mallory a plus de difficulté que la chaîne de Bob.

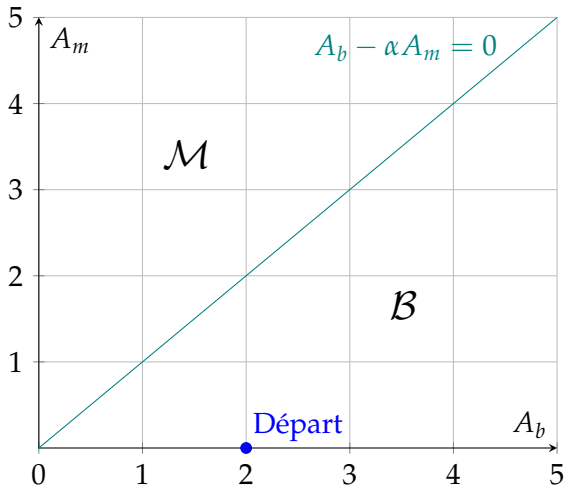

FIGURE 4.1 – Modélisation de la marche aléatoire pour $m = 2$ et $\alpha = 1$

On part du principe que Bob part avec une avance de m blocs, donc $V_0 = (m, 0)$. On cherche à déterminer la probabilité que Mallory rattrape Bob, c'est à dire la probabilité $P_{\text{gagne}}(m)$ que la marche aléatoire atteigne la droite $A_b - \alpha A_m = 0$, sachant que la marche aléatoire a commencé à la position $(m, 0)$. $P_{\text{gagne}}(m)$ est décroissante en fonction de m : plus Bob part avec une avance grande, moins Mallory a de chances de le rattraper. On cherche donc :

$$\min_{m \in \mathbb{N}} P_{\text{gagne}}(M) < \varepsilon \tag{4.7}$$

Avec ε notre paramètre de sécurité, la probabilité que Mallory rattrape Bob malgré une avance de m blocs.

4.2.2 Résultats

Pour calculer la probabilité $P_{\text{gagne}}(m)$, il fallait une fonction $f(x, y, x', y', \alpha)$ qui renvoie le nombre de chemins pour aller du point (x, y) au point (x', y') sans passer au dessus de la droite $A_b - \alpha A_m = 0$. On a assez vite trouvé une formule pouvant nous aider [10] :

$$f(x,y,x',y',\alpha) = \binom{(x'-x)+(y'-y)}{x'-x} - (\alpha+1)\binom{(x'-x)+(y'-y)}{y'-y} \forall \alpha \geq 1 \tag{4.8}$$

Malheureusement, cette expression n'est valable que pour $\alpha \geq 1$. On s'est assez vite rendu compte que, malgré l'apparente similarité entre les deux cas $\alpha \geq 1$ et $\alpha < 1$, les deux problèmes impliquaient des méthodes de résolution différentes. On a donc passé un certain temps à essayer de trouver une formule pour $\alpha < 1$. J'ai développé un programme permettant de tester

nos formules et comparer nos résultats avec ceux obtenus par simulation. On a fini par trouver une formule récursive pour $\alpha < 1$:

$$f(0,1,x,y) = \binom{x+y-1}{y} - g(y-1) - \sum_{i=0}^{y-2} g(i) \binom{x+y-i-\lfloor \frac{i+1}{4} \rfloor - 3}{y-i-1} \tag{4.9}$$

$$g(z) = \sum_{i=1}^{\lceil \frac{z+1}{4} \rceil} \left[\binom{z+\lceil \frac{z+1}{4} \rceil - i}{z} \sum_{j=0}^{z-2} g(j) \binom{z+\lfloor \frac{z}{4} \rfloor - j - \lfloor \frac{j+1}{4} \rfloor - i - 1}{z-j-1} \right] \tag{4.10}$$

Cette formule permettait de compter le nombre de chemins permettant de relier deux points tout en restant strictement en dessous de la droite $A_b - 1/4A_m = 0.$, mais uniquement pour un point de départ $(0,1)$. On a donc cherché à généraliser cette formule pour un point de départ quelconque (x,y). On a finalement trouvé une technique générale permettant de calculer le nombre de chemins pour relier deux points (x,y) et (x',y') pour n'importe quel $\alpha \leq 1$ dans un ouvrage de combinatoire [10].

Malheureusement, malgré nos efforts, une telle technique n'a pas abouti car elle impliquait des outils mathématiques complexes qu'aucun d'entre nous ne maîtrisait. On a donc essayé de trouver une autre méthode pour déterminer la valeur de m, que je décrirai dans la section suivante 4.3.

Malgré tout, nous avions une formule pour $\alpha \geq 1$, et j'ai pu calculer une valeur de m pour $\alpha = 1$ et $\alpha = 4$ comme suit :

$$P_{\text{gagne}}(m) = \sum_{A_b=m}^{\infty} f(m,0,A_b,\alpha A_m,)\mu(\alpha)^{\alpha A_m}(1-\mu(\alpha))^{A_b} \tag{4.11}$$

Avec f de l'équation 4.8 pour $\alpha \geq 1$. J'ai pu effectuer l'application numérique pour au moins deux valeurs de α : $\alpha = 1$ et $\alpha = 4$. Le tableau 4.2 donne les valeurs de m pour un $\varepsilon = 10^{-6}$ et un adversaire détenant $1/3$ de la puissance de calcul.

α	m
1	65
4	26

TABLE 4.2 – m en fonction de α, avec $q = 1/3$ et $\varepsilon = 10^{-6}$

On peut voir que la valeur de m est plus grande pour $\alpha = 1$ que pour $\alpha = 4$. On peut en déduire qu'il sera plus facile pour l'adversaire de rattraper l'honnête si l'adversaire mine des blocs à la target officielle, il faut donc augmenter la valeur de m pour empêcher l'honnête de se faire rattraper. Ces valeurs de m serviront donc pour valider les propriétés du second modèle, si on trouve des m dans le même ordre de grandeur, on pourra raisonnablement penser que les résultats sont corrects.

4.3 Recherche de m par la ruine du joueur

La recherche de m par la méthode de la marche aléatoire décrite dans la section 4.2 n'ayant pas abouti, on a cherché une autre méthode pour déterminer les valeurs de m qui permettrait de garantir les propriétés souhaitées pour l'algorithme MLS. On est donc parti de l'idée originale

de Nakamoto, qui a déterminé la probabilité pour l'adversaire de rattraper l'honnête en utilisant une loi de Poisson [3]. On a donc cherché à adapter cette méthode pour notre problème, à savoir déterminer la probabilité pour l'adversaire de rattraper l'honnête, sachant que la valeur relative entre les blocs honnête et les blocs adversariaux peut varier d'un facteur α.

On a utilisé les travaux présenté dans les articles [11] et [12] qui apportent des compléments et des corrections au calcul original de Nakamoto. On a ensuite utilisé l'article [13] pour calculer la probabilité de ruine du joueur avec des gains différents des pertes, contrairement à la ruine du joueur "classique".

4.3.1 Construction du modèle

Le modèle de recherche de m par ruine du joueur modélise la progression de la chaîne honnête et de la chaîne adversariale par une chaîne de Markov. Dans le contexte de la ruine du joueur, on peut dire que l'honnête joue avec un solde initial de m. À chaque jeu, l'honnête peut gagner avec une probabilité $1 - \mu(\alpha)$ et perdre avec une probabilité $\mu(\alpha)$. Mais la différence avec la ruine du joueur classique, c'est que l'honnête ne perd pas la même quantité que ce qu'il gagne. En effet, si l'adversaire mine à une target $\alpha = 4$ (donc 4 fois plus facile que l'honnête), l'honnête gagnera 4 à chaque partie gagnée (*i.e.* un bloc miné), mais l'adversaire gagnera 1 à chaque partie gagnée.

Calcul de la probabilité de perte On s'intéresse donc à la probabilité que l'adversaire rattrape l'honnête. En terme de ruine du joueur, cela revient à calculer la probabilité que l'honnête soit ruiné. Avec des gains différents des pertes, ce calcul est assez complexe. Posons a le gain de l'honnête, et b le gain de l'adversaire. Ainsi, si $\alpha = 4$, on a $a = 4$ et $b = 1$, si $\alpha = 1$, on a $a = 1$ et $b = 1$ et si $\alpha = 1/4$, on a $a = 1$ et $b = 4$. Avec les travaux présentés dans les articles [11] et [12], on a pu déterminer la probabilité que l'adversaire rattrape l'honnête malgré une avance de m blocs (ou de $m \cdot a$ difficulté) comme suit :

$$P_{\text{gagne}}(m) = 1 - \sum_{k=0}^{ma-1} \frac{\lambda^k e^{-\lambda}}{k!}(1 - P_{\text{ruin}}(ma - kb)) \tag{4.12}$$

$$\lambda = \frac{m\mu(\alpha)}{1 - \mu(\alpha)} \tag{4.13}$$

Il s'agit d'une loi de Poisson de paramètre λ représentant le taux de création de blocs de l'adversaire et $P_{\text{ruin}}(m)$ la probabilité que l'honnête soit ruiné avec un solde de difficulté $ma - kb$.

Calcul de la probabilité de ruine C'est donc la valeur de $P_{\text{ruin}}(M)$ qui nous reste à déterminer. Il s'agit du calcul de la probabilité de ruine du joueur avec des gains différents des pertes. Ce genre de calcul est assez complexe, mais l'article [13] nous à donné des pistes pour calculer la probabilité de ruine du joueur dans ce genre de situation. On pose une série de Laurent $p(z)$ représentant l'espérance de gain du joueur :

$$p(z) = (1 - \mu(\alpha))z^a + \mu(\alpha)z^{-b} + 1 \tag{4.14}$$

On cherche ensuite les ν racines η de $p(z)$ dans le disque unité $|z| < 1$. La probabilité de ruine du joueur est alors donnée par :

$$P_{\text{ruin}}(M) = \sum_{j=1}^{\nu} \eta_j^M \prod_{i \neq j} \frac{1-\eta_i}{\eta_j - \eta_i} \tag{4.15}$$

L'équation 4.12 nous donne donc la probabilité que l'adversaire rattrape l'honnête, sachant que l'honnête à une avance de M difficulté. Notre but sera de rendre cette probabilité inférieure à un certain ε pour garantir que l'honnête ne sera pas rattrapé par l'adversaire même si celui-ci triche sur la target.

Vérification du modèle À l'heure où j'écris ce mémoire, nous sommes en train de refaire les preuves des équations 4.14 et 4.15 pour être sûr de leur validité. En effet, bien que l'article [13] ai été publié et donc validé par des pairs, il est toujours bon de vérifier les résultats. C'est également l'occasion de comprendre en profondeur le fonctionnement de ces équations, et d'être convaincu que l'application que l'on en fait est cohérente avec ce que l'on cherche à modéliser.

4.3.2 Résultats

Pour trouver les valeurs de m qui permettraient de garantir les propriétés souhaitées pour l'algorithme MLS, j'ai développé un programme permettant de calculer par recherche dichotomique les valeurs de m pour différentes valeurs de $\alpha \in [1/4, 4]$. On cherche le minimum de m tel que la probabilité que l'adversaire rattrape l'honnête soit inférieure à ε (voir équation 4.7).

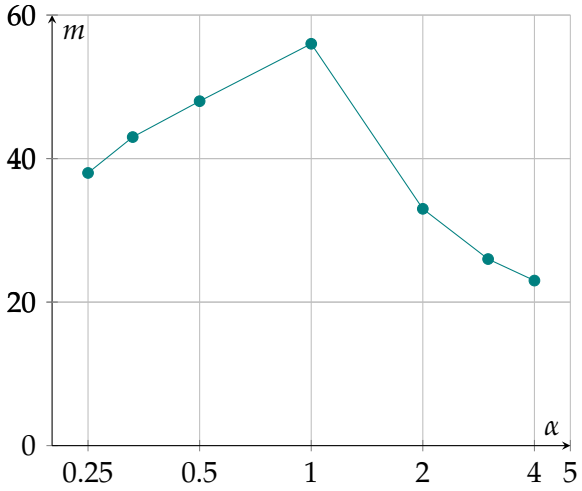

FIGURE 4.2 – Valeur de m en fonction de α, avec $q = 1/3$ et $\varepsilon = 10^{-6}$

On a décidé de garder $\varepsilon = 10^{-6}$ comme paramètre de sécurité, considérant que c'était une bonne valeur de sécurité. Évidemment, plus ε est petit, plus la sécurité est grande, mais la valeur de m sera plus grande, augmentant ainsi la taille du compressé.

Bien que l'application des modèles décrit dans les articles [4] et [5] aient prouvés que la propriété de qualité de la chaîne n'est respectée que si $q \leq 1/3$ (voir la sous-section 3.2.2), j'ai tout de même prévu de faire varier la puissance de calcul de l'adversaire $q \in [0.1, 0.45]$ pour voir l'impact de la puissance de calcul de l'adversaire sur la valeur de m. Prendre en compte cette variation demande encore quelques ajustements dans mon programme, c'est pourquoi je n'ai pas encore de résultats impliquant $q \neq 1/3$ à présenter. La dernière semaine de mon stage sera consacrée à l'étude de ces résultats avec un $q \in [0.1, 0.45]$.

α	m
1/4	38
1	56
4	23

TABLE 4.3 – m en fonction de α, avec $q = 1/3$ et $\varepsilon = 10^{-6}$

Le tableau 4.3 donne les valeurs de m pour un $\varepsilon = 10^{-6}$ et un adversaire détenant $1/3$ de la puissance de calcul. La valeur de m à choisir sera donc la plus grande des valeurs de m pour chaque α. La valeur optimale de m sera donc 56.

Les résultats sont édifiants : si ils sont validés, ces résultats nous permettront d'annoncer qu'il suffit d'attendre $m = 56$ blocs pour être sûr que notre bloc ne sera pas invalidé par un adversaire qui triche, peut importe la valeur de α. Ce résultat est intéressant pour MLS, mais aussi pour la chaîne Bitcoin : il est plus rentable pour l'adversaire de "jouer le jeu" du protocole et de ne pas faire baisser artificiellement sa target.

Malgré tout, ces résultats sont à relativiser : au moment où j'écris ce mémoire, nous sommes en train vérifier de manière bien plus formelle que notre méthodes et les outils que nous avons utilisés sont bien adaptés au problème que nous cherchons à résoudre. Il est donc toujours possible que ces résultats soient invalidés par la suite.

4.4 Autres travaux

En plus des travaux que j'ai décrit dans les sections précédentes, j'ai réalisé quelques travaux satellites qui n'ont pas demandé autant de temps, et ne méritaient pas une section à eux seuls.

Communication Ce stage a été l'occasion pour moi de travailler dans le monde de la recherche. J'ai donc eu l'occasion de participer à une activité essentielle dans la vie de chercheur : la communication des travaux, et le partage des connaissances. J'ai donc eu la chance d'assister aux conférences *AlgoTel* et *CoRes*[1], conférences nationales du GDR Réseaux et Systèmes Distribués[2] sur les algorithmes des télécommunications et les protocoles de communication. J'ai pu assister à plusieurs présentations, certaines directement liées aux problématiques Blockchain. J'ai aussi participé à un séminaire avec toute l'équipe de SOTERN, c'était l'occasion de présenter mes travaux, et de discuter des problèmes que j'ai rencontrés. J'ai donc fait une présentation de mes travaux, et je me suis essayé à l'exercice de la présentation face à un public de chercheurs.

Apprentissage du RUST Sous les conseils de mon tuteur, j'ai commencé à apprendre le langage de programmation RUST quand j'avais le temps. J'ai donc appris les bases du langage durant environ une vingtaines d'heures. L'apprentissage de ce langage m'aurait permis de travailler sur des démonstrateurs de concepts pour MLS ou d'autres protocoles ensuite. Cependant, la recherche de m a été bien plus longue que prévu, et ces compétences n'ont pas été sollicitées. Malgré tout, c'est un langage de plus en plus utilisé, et les concepts et outils qu'il propose sont très intéressants. C'est donc une compétence que je suis content d'avoir acquise.

1. https://algotelcores2024.sciencesconf.org/
2. https://gdr-rsd.fr/

Chapitre 5

Conclusion

Au terme de ce stage, j'ai acquis de sérieuses connaissances et compétences dans le domaine des technologies Blockchains. Ce stage demandait plus de compétences en mathématiques que prévu, et j'ai fait des progrès significatifs dans ce domaine. En choisissant ce stage, j'avais comme objectif de découvrir le monde de la recherche. J'ai pu travailler avec des chercheurs d'IMT Atlantique, et j'ai pu découvrir le fonctionnement d'un laboratoire de recherche. J'ai également participé à des réunions de travail et à des séminaires. En somme, ce stage a été une expérience très enrichissante.

Bien que l'article de recherche ne soit pas encore publié, on a pu obtenir des résultats, et il ne reste que quelques preuves et démonstrations à rédiger avant de pouvoir soumettre l'article à une conférence. Plusieurs pistes d'amélioration sont envisageables pour la suite. On pourrait par exemple essayer d'adapter l'algorithme de compression à d'autres types de structures de données, comme les structures DAG (Directed Acyclic Graph) comme dans l'article [14]. On pourrait aussi essayer d'affiner la valeur de m, en prenant en compte les effets du changement de target entre deux epoch. Pour conclure, je suis heureux d'avoir pu participer à ce projet.

Pour finir ce mémoire, je tiens à remercier toutes les personnes qui m'ont aidé et soutenu durant ce stage. Je tiens à remercier mon tuteur et toute l'équipe de recherche d'IMT Atlantique pour m'avoir accueilli et encadré durant ces 6 mois.

Annexe A

Rappel des notations

Notations	Signification
$\mathcal{C}$	Chaîne de blocs
h	Header d'un bloc.
d	Données applicatives d'un bloc.
c	Nonce d'un bloc.
$\mathfrak{h}(\cdot)$	Fonction de hashage.
κ	Nombre de bit en sortie de la fonction de hashage.
T	Target de la PoW (T_b la target honnête et T_m la target adversariale).
D	Difficulté de la PoW (telle que $D = 1/T$).
k	Nombre de blocs à tronquer à chaque chaîne honnête pour obtenir un préfixe commun.

TABLE A.1 – Tableau récapitulatif des notations relatives à la structure de la chaîne (*sec. 3.1*)

Notations	Signification
q	Proportion de puissance de calcul de l'adversaire.
p	Proportion de puissance de calcul de l'honnête.
n	Nombre total de nœuds du réseau.
t	Nombre de nœuds malveillants.
r	Nombre de requêtes à la fonction de hashage par ronde et par nœud.
P_T	Probabilité de trouver un bloc en une seule requête avec une target. T (P_b avec la target de l'honnête et P_m avec la target.
f	Probabilité qu'au moins un nœud honnête trouve un bloc en une ronde.
$D(\cdot)$	Fonction de recalcul de la difficulté.
X_i	Variable aléatoire, vaut 1 si au moins un nœud honnête à trouvé un bloc à la ronde i.
Y_i	Variable aléatoire, vaut 1 si un seul nœud honnête à trouvé un bloc à la ronde i .
Z_i	Variable aléatoire, vaut 1 si au moins un nœud malveillant à trouvé un bloc à la ronde i.
ε	Paramètre de sécurité.
Λ	Nombre de rondes minimum à considérer avant de pouvoir parler d'exécution typique.
τ	Facteur limitant la variation de T entre deux epochs.
(η, θ)	Bornes de variation de T entre deux nœuds à la même epoch.
(γ, s)	Paramètres limitant la variation de la population du système entre deux rondes.
L	Nombre de rondes depuis le début du protocole.
J	Nombre de blocs par epoch.

TABLE A.2 – Tableau récapitulatif des notations relatives aux modèles backbone (*sec. 3.2 et 3.3*)

Notations	Signification
l	Niveau d'un bloc.
m	Paramètre de l'algorithme MLS.
α	Rapport entre la target de l'adversaire et celle de l'honnête (tel que $T_b = \alpha T_m$).
$\mu(\alpha)$	Probabilité qu'un bloc trouvé appartienne à l'adversaire.
$P_{\text{gagne}}(m)$	Probabilité pour l'adversaire de rattraper son retard de m blocs.

TABLE A.3 – Tableau récapitulatif des notations relatives à MLS (*sec. 3.4, 3.5 et 4.1*)

Notations	Signification
V_k	Nombre de victoire à la k-ième partie.
A_b	Nombre de parties gagnées par l'honnête.
A_m	Nombre de parties gagnées par l'adversaire.
$\mathcal{M}$	Espace du quart de plan ou l'adversaire à gagné.
$\mathcal{B}$	Espace du quart de plan ou l'honnête à gagné.

TABLE A.4 – Tableau récapitulatif des notations relatives à la marche aléatoire (*sec. 4.2*)

Notations	Signification
a	Perte de l'adversaire.
b	Gain de l'adversaire.
$p(z)$	Espérance de gain du joueur.
η	Nombre de racines de $p(z)$.
ν_i	i-ème racine de $p(z)$.
$P_{\text{ruin}}(M)$	Probabilité pour le joueur d'être ruiné en possédant une fortune de M.

TABLE A.5 – Tableau récapitulatif des notations relatives à la ruine du joueur (*sec. 4.3*)

Annexe B

Algorithmes

B.1 Algorithmes MLS dans le cas statique

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

Algorithme 1 : Algorithme de compression MLS dans le cas statique

```
function maxvalid_{m,k}(Π, Π′)
    if Π is not valid then
        return Π′
    end
    if Π′ is not valid then
        return Π
    end
    (χ, ℓ, D) ← Dissolve_{m,k}(Π)
    (χ′, ℓ′, D′) ← Dissolve_{m,k}(Π′)
    M ← {μ ∈ ℕ : D[μ] ∩ D′[μ] ≠ ∅}
    if M = ∅ then
        if ℓ′ > ℓ then
            return Π′
        end
        return Π
    end
    μ ← min(M)
    b ← (D[μ] ∩ D′[μ])[−1]
    if |D′[μ]b :| > |D[μ]b :| then
        return Π′
    end
    return Π
```

Algorithme 2 : Algorithme de comparaison MLS dans le cas statique

B.2 Algorithmes MLS dans le cas dynamique

```
function Compress_{m,k}(C)
    D ← ∅
    χ ← C[−k :]
    C* ← C[: −k]
    if |C*| ≥ 2m then
        ℓ ← max{μ : |C* ↑^μ | ≥ 2m}
        D[ℓ] ← C* ↑^ℓ
        for μ ← ℓ − 1 to 0 do
            b* ← C* ↑^{μ+1} [−m]
            D[μ] ← C* ↑^μ [−2m :] ∪ C* ↑^μ {b* :}
        end
    else
        ℓ ← 0
        D[0] ← C*
    end
    return (D, ℓ, χ)
```

Algorithme 3 : Algorithme de compression MLS dans le cas dynamique

```
1  function maxvalid_{m,k}(Π_1, ..., Π_n)
2      candidates ← ∅
3      (D, χ, ℓ) ← ⊥
4      i ← 0
5      while (D, χ, ℓ) is ⊥ and i ≤ n do
6          if valid(Π_i) then
7              (D, χ, ℓ) ← Compress_{m,k}(Π_i)
8              candidates ← candidates ∪{(D, χ, ℓ)}
9          end
10         i ← i + 1
11     end
12     if ¬((D, χ, ℓ) is ⊥) then
13         for j ← i to n do
14             if valid(Π_j) then
15                 (D', χ', ℓ') ← Compress_{m,k}(Π_j)
16                 M ← {μ ∈ ℕ | D[μ] ∩ D'[μ] ≠ ∅}
17                 if M = ∅ then
18                     candidates ← candidates ∪{(D', χ', ℓ')}
19                 else
20                     μ ← min M
21                     b ← (D[μ] ∩ D'[μ])[−1]
22                     if score(D'{b :} · χ') ≥ score(D{b :} · χ) then
23                         (D, χ, ℓ) ← (D', χ', ℓ')
24                     end
25                 end
26             end
27         end
28     end
29     if |candidates| = 1 then
30         best ← (D, χ, ℓ)
31     else
32         best ← ⊥
33         beacon ← generate_beacon()
34         send_tx_beacon(beacon)
35     end
36     return best
```

Algorithme 4 : Algorithme de comparaison MLS dans le cas dynamique

Annexe C

Bibliographie

[1] Ludinard Romaric and Anceaume Emmanuelle. Bc4ssi. In *Projet ANR*, 2022. https://hub.imt-atlantique.fr/bc4ssi/ (consulté le 30/07/2024).

[2] DGE. Les verrous technologiques des blockchains. In *Rapport de la DGE*, 2021. https://www.entreprises.gouv.fr/files/files/etudes-et-statistiques/rapport-final-blockchain.pdf (consulté le 18/07/2024).

[3] Satoshi Nakamoto. Bitcoin : A peer-to-peer electronic cash system. *SSRN Electronic Journal*, 2008.

[4] Juan A. Garay, Aggelos Kiayias, and Nikos Leonardos. The bitcoin backbone protocol : Analysis and applications. *Journal of the ACM*, April 2024.

[5] Juan Garay, Aggelos Kiayias, and Nikos Leonardos. *The Bitcoin Backbone Protocol with Chains of Variable Difficulty*, page 291–323. Springer International Publishing, 2017.

[6] Aggelos Kiayias, Nikos Leonardos, and Dionysis Zindros. Mining in logarithmic space. In *Proceedings of the 2021 ACM SIGSAC Conference on Computer and Communications Security*, CCS '21. ACM, November 2021.

[7] Sunny King and Scott Nadal. Ppcoin : Peer-to-peer crypto-currency with proof-of-stake, 2012.

[8] Stefan Dziembowski, Sebastian Faust, Vladimir Kolmogorov, and Krzysztof Pietrzak. Proofs of space. Cryptology ePrint Archive, Paper 2013/796, 2013. https://eprint.iacr.org/2013/796.

[9] Emmanuelle Anceaume, Thibaut Lajoie-Mazenc, Romaric Ludinard, and Bruno Sericola. Safety analysis of bitcoin improvement proposals. In *2016 IEEE 15th International Symposium on Network Computing and Applications (NCA)*, pages 318–325, 2016.

[10] Miklos Bona. *Handbook of Enumerative Combinatorics*. Chapman and Hall/CRC, March 2015.

[11] Meni Rosenfeld. Analysis of hashrate-based double spending, 2014.

[12] A. Pinar Ozisik and Brian Neil Levine. An explanation of nakamoto's analysis of double-spend attacks, 2017.

[13] Guy Katriel. Gambler's ruin probability—a general formula. *Statistics & Probability Letters*, 83(10) :2205–2210, 2013.

[14] Emmanuelle Anceaume, Antoine Guellier, Romaric Ludinard, and Bruno Sericola. Sycomore : A permissionless distributed ledger that self-adapts to transactions demand. In *2018 IEEE 17th International Symposium on Network Computing and Applications (NCA)*. IEEE, November 2018.