HAL
open science

Sycomore : un registre de transactions distribué et public au débit auto-adaptatif

Emmanuelle Anceaume, Antoine Guellier, Romaric Ludinard, Bruno Sericola

To cite this version:

Emmanuelle Anceaume, Antoine Guellier, Romaric Ludinard, Bruno Sericola. Sycomore : un registre de transactions distribué et public au débit auto-adaptatif. ALGOTEL 2019 - 21èmes Rencontres Francophones sur les Aspects Algorithmiques des Télécommunications, Jun 2019, Saint Laurent de la Cabrerisse, France. pp.1-4. hal-02118385

HAL Id: hal-02118385

https://hal.science/hal-02118385

Submitted on 3 May 2019

HAL is a multi-disciplinary open access archive for the deposit and dissemination of scientific research documents, whether they are published or not. The documents may come from teaching and research institutions in France or abroad, or from public or private research centers.

L'archive ouverte pluridisciplinaire **HAL**, est destinée au dépôt et à la diffusion de documents scientifiques de niveau recherche, publiés ou non, émanant des établissements d'enseignement et de recherche français ou étrangers, des laboratoires publics ou privés.

Sycomore : un registre de transactions distribué et public au débit auto-adaptatif

E. Anceaume[1], A. Guellier, R. Ludinard[2] et B. Sericola[3]

[1]*CNRS, UMR 6074 - IRISA, emmanuelle.anceaume@irisa.fr*
[2]*IMT Atlantique, romaric.ludinard@imt-atlantique.fr*
[3]*INRIA Rennes - Bretagne Atlantique, bruno.sericola@inria.fr*

Bitcoin est un système pair-à-pair de crypto-devises considéré comme l'un des pionniers de ce type de système. La *blockchain*, une structure de donnée particulière, est la clé de voûte de ce système. Celle-ci est mise à jour régulièrement par l'ajout d'une nouvelle séquence de transactions. La fréquence des mises à jour constante, couplée à une borne maximale sur les transactions incluses dans celles-ci induit un débit d'enregistrement maximal borné. Cette limite, relativement basse, est l'un des freins à l'essor de Bitcoin. Cet article présente une structure de données alternative orientée graphe permettant de s'adapter dynamiquement au besoin d'enregistrement de transactions.

Mots-clefs : blockchain, scalabilité

1 Introduction

Bitcoin [Nak08] est un système de crypto-devises offrant à ces utilisateurs la possibilité de transférer des fonds (en bitcoins) de manière sécurisée sans tiers de confiance. Ce système s'appuie sur trois éléments distincts : les *transactions*, qui permettent la manipulation (création, authentification du possesseur, transfert) de bitcoins, un *historique* permettant l'enregistrement de chaque création et transfert de bitcoin, permettant ainsi d'assurer la non-répudiation d'un transfert, et enfin un *protocole de diffusion de l'information*, permettant la mise à jour de cet historique. Les utilisateurs interagissent avec le système au travers des transactions. Les *pairs*, qui constituent avec les *mineurs*, le système Bitcoin diffusent ces transactions dans le système tandis que les mineurs mettent à jour l'historique du système. La contre-mesure aux attaques par double dépense (existence de deux transaction cherchant à dépenser une même précédente transaction) proposée par Bitcoin réside dans la mise en place d'un *historique intègre* des transactions émises dans le système. Cet historique est mis en œuvre par une blockchain : une structure d'arbre dont une liste chaînée de *blocs* constitués d'une entête et d'une séquence de transactions valides [ALMLS16] est extraite.

Afin de mettre à jour cet historique, chaque mineur sélectionne un ensemble de nouvelles transactions valides, les ordonne et essaie de construire un bloc valide. L'entête d'un bloc comporte différentes informations dont : l'identifiant (hash) du bloc précédent dans l'arbre, un résumé des transactions incluses dans le bloc (racine de l'arbre de Merkle de ces transactions), une preuve de travail. Un bloc est considéré comme localement valide si *1)* sa taille est inférieure à un seuil défini par le protocole †, *2)* il contient une séquence de nouvelles transactions émises dans le système et considérées comme valides, *i.e.* ne crée pas de double dépense, *3)* le résumé fourni dans l'entête correspond aux transactions incluses dans le bloc (dans le même ordre), *4)* il satisfait la preuve de travail. Lorsqu'un mineur crée un bloc valide, il le diffuse dans le système et l'ajoute à sa blockchain locale avant de recommencer ce processus. À réception d'un bloc, un mineur vérifie également ces conditions. Si celles-ci sont valides, le mineur vérifie que l'historique de contient de pas bloc ayant le même prédécesseur : dans ce cas, le bloc est ajouté à la blockchain courante et le mineur essaie de créer un nouveau bloc successeur. Dans le cas contraire, le bloc est en conflit et un protocole d'arbitrage est exécuté qui extrait la liste chaînée la plus longue allant d'une feuille à la racine de l'arbre.

†. Avant le 24 août 2017, cette taille doit être inférieure à 1Mo, la définition de la taille d'un bloc est ensuite légèrement modifiée avec l'adoption de SegWit, mais n'a pas d'influence sur la suite de cet article.

Ce protocole de maintenance de l'historique, noté π_{btc}, offre malheureusement de faibles performances : le système est mis à jour à taux fixe (un bloc est généré en moyenne toutes les dix minutes) et chaque bloc n'embarque d'une quantité bornée d'information, menant à n'enregistrer que 3 à 7 transactions par seconde [CDE+16]. Des approches visant à améliorer ces performances proposent de modifier la structure interne de l'historique en une structure de graphe. Ainsi, [Bai16, Pop17, Chu17] proposent une structure de graphe orienté acyclique dont la mise à jour est effectuée par une entité centralisée de confiance, tandis que d'autres [SZ13, SLZ16] s'affranchissent de ce tiers de confiance mais ne structurent l'historique qu'en un graphe orienté mais pas acyclique. Un algorithme complexe doit alors être mis en œuvre afin d'en extraire l'historique des transactions.

Cet article présente le protocole π_{syc} ‡ permettant la mise à jour, sans tiers de confiance, d'un historique structuré en graphe orienté acyclique. Ce protocole est basé sur le constat suivant : la validité d'une transaction impose que les transactions consommées soient connues et non dépensées. L'implémentation de l'historique doit donc respecter la causalité des transactions. Tout historique satisfaisant cet ordre partiel permet de calculer le même ensemble de transactions dépensables. Le protocole π_{syc} construit, sans tiers de confiance, un graphe orienté acyclique équilibré, appelé SYC-DAG, sur lequel *1)* les transactions sont partitionnées sur les blocs, *2)* il est impossible de choisir le prédécesseur d'un bloc et *3)* le nombre de feuilles s'adapte dynamiquement à la charge du système, *i.e.* au transactions émises. Le protocole π_{syc} ainsi que la structure de données associées est présenté dans la section 2, les travaux futurs dans la section 3.

2 Sycomore

Le protocole π_{syc} vise à améliorer la scalabilité de Bitcoin, *i.e.* le nombre de transactions enregistrées par seconde, de manière dynamique, c'est-à-dire en adaptant la capacité d'enregistrement à la quantité de transactions soumises. Par conséquent, ce protocole s'appuie sur les transactions Bitcoin et sur le même protocole de communication mais modifie la structure interne de l'historique.

Dans les deux protocoles π_{btc} et π_{syc}, les blocs sont créés au moyen d'une preuve de travail. L'entête d'un bloc fourni par un mineur témoigne de sa vision locale de l'historique , ainsi qu'un engagement sur les données qui l'accompagne. Dans le cas de π_{btc}, le témoignage est fourni par le hash du bloc qu'il considère comme le plus haut dans la chaîne, tandis que l'engagement correspond à la racine de l'arbre de Merkle des transactions incluses dans le corps du bloc. Dans le cas de π_{syc}, les blocs sont organisés au sein d'un SYC-DAG. Les entêtes de ces blocs fournissent des informations propres à chaque feuille du graphe et fournissent, pour chacune d'elles, un engagement sous forme d'une racine d'arbre de Merkle. Dans les deux protocoles, la preuve de travail permet de sceller les informations du bloc de manière intègre.

À chaque bloc du SYC-DAG, on associe un bloc appelé prédécesseur. À l'ajout d'un bloc, le prédécesseur est une feuille du SYC-DAG. Ce prédécesseur n'est pas inclus de manière explicite dans le nouveau bloc, mais est déterminé à partir de l'entête de celui-ci. De cette manière, le prédécesseur effectif d'un bloc ne peut être ni choisi ni prédit à l'avance. Ce calcul est vérifiable par tout participant, ainsi que les transactions associées au bloc, assurant ainsi l'intégrité de la structure. On associe à chaque bloc du SYC-DAG une chaîne binaire appelée *label*. Par convention, on note ε le label vide. Ces labels sont utilisés pour partitionner les nouvelles transactions sur les feuilles du graphe. Ainsi, un bloc ayant pour label ℓ ne peut contenir que des transactions dont l'identifiant est préfixé lui-même par ℓ. L'ensemble des labels des feuilles du SYC-DAG forme une partition de l'espace de hachage $\{0,1\}^{256}$. L'entête d'un bloc contient l'ensemble des labels des feuilles du SYC-DAG au moment de sa création ainsi que le hash du bloc feuille correspondant à chaque label. Les propriétés des fonctions de hachage assurent l'équilibre de la structure.

La notion de chaîne, centrale dans le protocole π_{syc}, permet de contrôler le nombre de feuilles du SYC-DAG. La figure 1 illustre les différentes notions liées aux chaînes et à la construction d'un SYC-DAG. Chaque sommet du graphe de la figure 1 représente un bloc dont le label est situé au dessus du sommet. Étant donné un ensemble de blocs B, une chaîne $\mathcal{C} = (b_1 b_2 \dots b_c)$, avec $c \geq 2$ de B est une séquence de blocs commençant à b_1 et s'achevant à b_c telle que *1)* tout bloc $b_i, 2 \leq i \leq c$ a un unique prédécesseur et ce prédécesseur appartient à $\mathcal{C}$, *2)* tout bloc $b_i, 1 \leq i \leq c-1$ est le prédécesseur d'un unique bloc appartenant à $\mathcal{C}$, et *3)* le prédécesseur de b_1 n'appartient pas à $\mathcal{C}$ et aucun bloc de $\mathcal{C}$ n'a pour prédécesseur b_c. Dans la

‡. Une version longue de cet article est disponible [AGLS18].

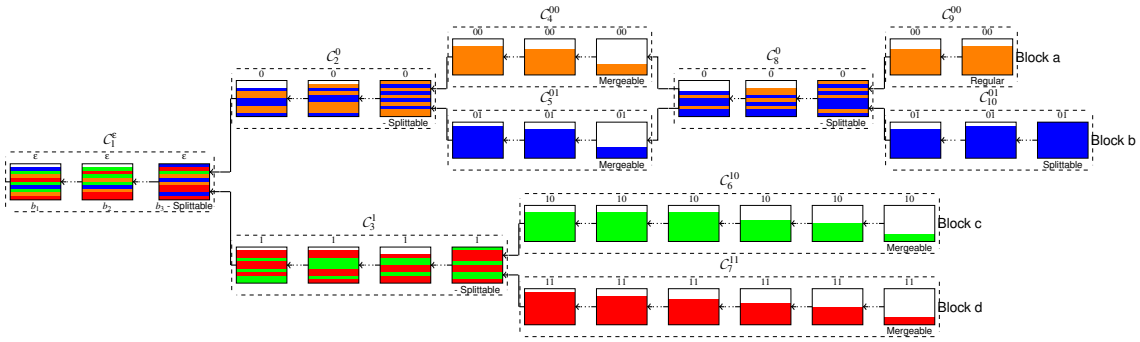

FIGURE 1: Un registre possible construit avec π_{syc}

figure 1, les blocks b_1, b_2, b_3 forment une chaîne. Les chaînes de la figure sont toutes représentées par des ensembles pointillés de blocs.

Étant donné $c_{min} > 2$ un paramètre du système, on définit pour toute chaîne de blocs $\mathcal{C} = (b_1 b_2 \dots b_c)$ sa *charge* comme la moyenne arithmétique des rapports entre la taille (en octets) des blocs qui la composent et la taille maximale autorisée par le système. Étant donnés $0 < \gamma < \Gamma < 1$, le bloc b_c de $\mathcal{C}$ est dit *splittable* si la charge de la chaîne composée des c_{min} derniers blocs est supérieure à Γ. À l'inverse, il est dit *mergeable* si la charge de cette chaîne est inférieure à γ. Une chaîne qui n'est ni splittable ni mergeable est dite *régulière*. La taille des blocs est représentée dans la figure 1 par la partie colorée du bloc, les couleurs représentent différents préfixes (*e.g.* 00, 01, 10, 11) des identifiants des transactions. Tous les blocs d'une chaîne $\mathcal{C}^\ell$ partagent le même label ℓ, préfixe des identifiants des transactions contenues dans ses blocs.

Deux chaînes $\mathcal{C}_1, \mathcal{C}_2$ sont dites *chaînes splits* si leurs labels respectifs ne diffèrent que sur le dernier bit et que leurs premiers blocs ont le même prédécesseur. Dans la figure 1, les chaînes $\mathcal{C}_4^{00}, \mathcal{C}_5^{01}$ sont deux chaînes splits dont le prédécesseur est le dernier bloc de $\mathcal{C}_2^0$. Lorsque deux chaînes splits sont sous-chargées, un bloc peut admettre comme prédécesseurs les derniers blocs de chacune de ces chaînes. Un tel bloc est le premier d'une *chaîne merge*. La chaîne $\mathcal{C}_8^0$ est la seule chaîne merge de la figure 1.

Un SYC-DAG est un graphe orienté acyclique ne contenant qu'un unique puits (le bloc *genesis*) et pour lequel on peut créer une partition $\mathcal{P} = \{\mathcal{C}^{\ell_1}, \dots, \mathcal{C}^{\ell_n}\}$ des blocs telle que :

i) pour tout élément de $\mathcal{P}$ de label ℓ, il existe, pour tout $0 \leq k \leq |\ell|$, un élément de $\mathcal{P}$ ayant un label de longueur k, préfixe de ℓ,

ii) pour toute chaîne merge $\mathcal{C}^{\ell_i}$ de $\mathcal{P}$, les chaînes $\mathcal{C}^{\ell_i.0}, \mathcal{C}^{\ell_i.1}$ appartiennent à $\mathcal{P}$ également, $\ell_i.b$ correspondant à la concaténation du label ℓ avec le bit $b \in \{0,1\}$,

iii) pour tout couple de chaîne $\mathcal{C}_1^\ell, \mathcal{C}_2^\ell$ de $\mathcal{P}$ de même label ℓ, leurs prédécesseurs sont différents.

Les sommets du graphe représenté dans la figure 1 peuvent être partitionnés ainsi :$\mathcal{P} = \{\mathcal{C}_1^\varepsilon, \mathcal{C}_2^0, \mathcal{C}_3^1, \mathcal{C}_4^{00}, \mathcal{C}_5^{01}, \mathcal{C}_6^{10}, \mathcal{C}_7^{11}, \mathcal{C}_8^0, \mathcal{C}_9^{00}, \mathcal{C}_{10}^{01}\}$. La structure de SYC-DAG généralise la notion de blockchain définie par Bitcoin : la liste des blocs issus de la blockchain Bitcoin constitue une unique chaîne dont le label est ε.

Le mécanisme de preuve de travail étant probabiliste par nature, il est possible que plusieurs mineurs créent un bloc dans un intervalle de temps très court. Dans le cas de π_{btc}, cette situation mène à deux mises à jours concurrentes nécessitant un arbitrage visant à choisir lequel de ces blocs doit étendre la blockchain Bitcoin. Dans le cas de π_{syc}, cette situation ne mène pas nécessairement à une incohérence : les deux blocs pouvant avoir un prédécesseur différent. Néanmoins, en cas d'incohérence, il suffit, de manière similaire à Bitcoin, de conserver le SYC-DAG ayant le plus fort niveau de confirmation du genesis bloc, *i.e.* contenant le plus long chemin d'une feuille au genesis bloc. Dans le cas d'un SYC-DAG ayant c feuilles et un taux de génération de blocs fixé à λ/c, la probabilité que deux blocs ayant le même prédécesseur dans un intervalle de temps $[0,t]$ est $1 - e^{-\lambda t/c}(1 + \lambda t/c)$. La figure 2 illustre la probabilité de fork dans le cas où un bloc est créé toutes les dix minutes en moyenne. Très naturellement, cette probabilité décroît avec le nombre de feuilles du SYC-DAG. La figure 3 illustre l'intervalle inter-blocs en fonction du nombre de feuilles à probabilité de fork constante. Très naturellement, à probabilité de fork constante, l'accroissement du nombre de feuilles permet de réduire le délai inter-blocs. On constate ainsi que le protocole π_{syc} permet d'augmenter le débit d'enregistrement sans pour autant remettre en cause la stabilité de l'historique. Cet

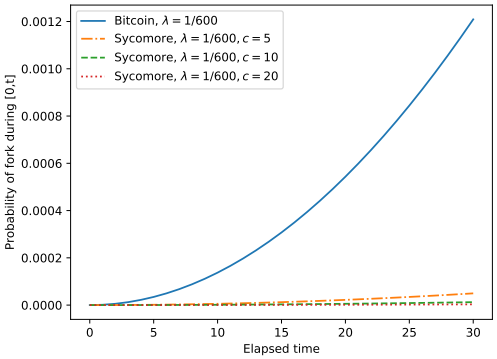

FIGURE 2: Probabilité de fork à taux de génération constant en fonction du nombre de feuilles du SYC-DAG

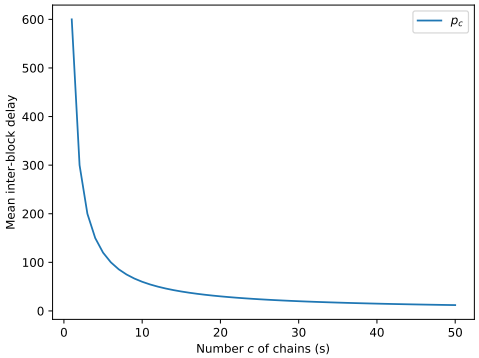

FIGURE 3: Délai inter-blocs à probabilité de fork constante en fonction du nombre de feuilles du SYC-DAG

accroissement est piloté par la charge des chaînes du SYC-DAG, permettant ainsi de réagir à des fluctuations ponctuelles (*e.g.* période de Noël, soldes, ...) ou de fond (*e.g.* accroissement de l'utilisation de Bitcoin).

3 Travaux en cours et futurs

Dans cet article nous avons présenté une structure alternative permettant de stocker l'historique des transactions émises dans Bitcoin sous forme d'un graphe orienté acyclique. En tirant parti de la nature distribuée de ce système, cette structure permet d'enregistrer de manière parallèle un nombre plus important de transactions sans augmenter les incohérences dans l'historique. Nous travaillons actuellement sur la validation théorique de la dynamique de cette structure. Une validation expérimentale par le déploiement d'un réseau Sycomore interfacé avec le réseau Bitcoin est en cours de réalisation.

Références

[AGLS18] E. Anceaume, A. Guellier, R. Ludinard, and B. Sericola. Sycomore : a Permissionless Distributed Ledger that self-adapts to Transactions Demand. In *17th IEEE International Symposium on Network Computing and Applications (NCA)*, 2018.

[ALMLS16] E. Anceaume, T. Lajoie-Mazenc, R. Ludinard, and B. Sericola. Safety Analysis of Bitcoin Improvement Proposals. In *15th IEEE International Symposium on Network Computing and Applications (NCA)*, 2016.

[Bai16] L. Baird. The SWIRLDS Hashgraph Consensus Algorithm : Fair, Fast, Byzantine Fault Tolerance. `http://www.swirlds.com/downloads/SWIRLDS-TR-2016-01.pdf`, 2016.

[CDE+16] K. Croman, C. Decker, I. Eyal, A. E. Gencer, A. Juels, A. E. Kosba, A. Miller, P. Saxena, E. Shi, E. G. Sirer, D. Song, and R. Wattenhofer. On scaling decentralized blockchains - (A position paper). In *Financial Cryptography and Data Security - FC International Workshops, BITCOIN, VOTING, and WAHC, Revised Selected Papers*, 2016.

[Chu17] A. Churyumov. ByteBall : A Decentralized System for Storage and Transfer of Value. `https://byteball.org/Byteball.pdf`, 2017.

[Nak08] S. Nakamoto. Bitcoin : A Peer-to-Peer Electronic Cash System. `https://bitcoin.org/bitcoin.pdf`, 2008.

[Pop17] S. Popov. The Tangle. `https://iota.org/IOTA_Whitepaper.pdf`, 2017.

[SLZ16] Y. Sompolinsky, Y. Lewenberg, and A. Zohar. SPECTRE : A fast and scalable cryptocurrency protocol. *IACR Cryptology ePrint Archive*, 2016, 2016.

[SZ13] Y. Sompolinsky and A. Zohar. Accelerating Bitcoin's Transaction Processing. Fast Money Grows on Trees, Not Chains. *IACR Cryptology ePrint Archive*, 2013, 2013.