RaiBlocks: Une crypto-monnaie en réseau distribuée sans frais

Colin LeMahieu
clemahieu@gmail.com

***Abstract*—Récemment, une forte demande et une modularité limitée ont fait augmenter les temps moyens de transactions et les frais des crypto-monnaies populaires, générant une expérience peu satisfaisante. Ici nous présentons Raiblocks, une crypto-monnaie avec une architecture nouvelle nommée "block-lattice" dans laquelle chaque compte a sa propre chaîne de blocs, permettant une vitesse de transaction quasi instantanée et une modularité illimité. Chaque utilisateur a sa propre chaîne de blocs, lui permettant de mettre à jour sa chaîne de façon asynchrone avec le reste du réseau, donnant lieu à des transactions rapides et des frais minimes. Les transactions gardent une trace du solde des comptes plutôt que des montants de transactions, ce qui permet de réduire agressivement la taille de la base de données sans compromettre la sécurité. Jusqu'à présent, le réseau Raiblocks a traité 4.2 millions de transactions avec un registre de seulement 1.7GB, sans réductions. Ces transactions sans frais et effectuées en une fraction de secondes font de RaiBlocks la crytpo-monnaie idéale pour les transactions faites par des consommateurs.**

***Index Terms*—crypto-monnaie, chaîne de blocs, raiblocks, registre distribué, digital, transactions**

I. Introduction

DEPUIS l'apparition du Bitcoin en 2009, nous assistons à un mouvement grandissant de personnes s'éloignant des monnaies traditionnelles soutenues par les gouvernements et les systèmes financiers pour se diriger vers des systèmes de paiement modernes basé sur la cryptographie, offrant la capacité de stocker et de transférer des fonds de manière sure et sans besoin pour des tiers de confiance [1]. Pour fonctionner correctement, une monnaie doit être facilement transférable, non réversible et doit avoir des frais limités ou pas de frais du tout. Les temps accrus de transactions, les frais importants et la modularité discutable de son réseau ont soulevé des questions quand à la fonctionnalité du Bitcoin en tant que monnaie courante.

Dans ce document, nous présentons RaiBlocks, une crypto-monnaie à faible latence bâtie sur une structure de données innovatrice de blocs-tréssés (block-lattice en anglais) offrant une modularité illimitée et pas de frais de transaction. RaiBlocks par nature est un protocole simple dont le seul but est d'être une crypto-monnaie à haute performance. Le protocole RaiBlocks peut tourner sur des logiciels peu gourmand en énergie lui permettant d'être une crypto-monnaie pratique et décentralisée faite pour un usage quotidien.

Les statistiques de crypto-monnaie rapportées dans ce document sont exactes à la date de publication.

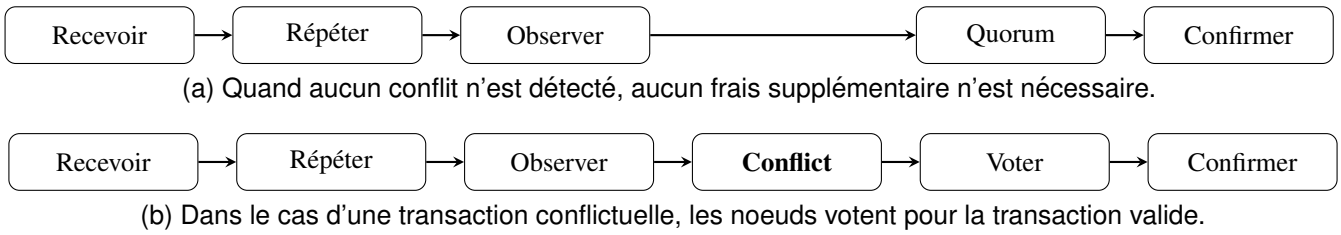
II. Contexte

En 2008, une personne anonyme sous le pseudonyme de Satoshi Nakamoto publie un document technique présentant la première crypto-monnaie décentralisée au monde, le Bitcoin [1]. Une des innovations clés amenée par le Bitcoin était la chaîne de blocs (blockchain en anglais), une structure de données publique, immutable, décentralisée qui est utilisée comme registre pour les transactions de la monnaie. Malheureusement, à mesure que le Bitcoin se développait, différents problèmes dans son protocole ont restreint l'utilisation du Bitcoin dans de nombreux domaines:

1) Modularité limitée: Chaque bloc dans la chaîne de blocs peut stocker une quantité limitée de données, cela signifie que le système ne peut traiter qu'un nombre limité de transactions par seconde, faisant de chaque espace dans un bloc une denrée. Actuellement le coût médian de transaction est de $10.38 [2].
2) Haute latence: Le temps moyen de confirmation est de 164 minutes [3].
3) Gourmand en énergie: L'estimation de la consommation électrique du réseau Bitcoin est de 27.28TWh par an, utilisant en moyenne 260KWh par transaction [4].

Bitcoin, et d'autres crypto-monnaies, fonctionnent en atteignant un consensus sur le registre global afin de vérifier les transactions légitimes tout en repoussant les acteurs malveillants. Le Bitcoin atteint un consensus via une mesure économique appelée preuve de travail (PoW en anglais). Dans un système PoW les participants rivalisent pour calculer un nombre, appelé un *nonce*, afin que le hash de la chaîne entière soit compris dans une fourchette prévue. Cette fourchette valable est inversement proportionnelle à la puissance de calcul cumulative de la totalité du réseau Bitcoin afin de maintenir un temps moyen constant pour trouver un nonce valide. Le découvreur d'un nonce valide est alors autorisé à ajouter le bloc à la chaîne de blocs ; donc ceux qui utilisent plus de ressources computationnelles pour calculer un nonce joue un rôle plus important dans la chaîne de blocs. La PoW permet une résistance accrue contre une attaque Sybil, dans laquelle une entité se comporte comme plusieurs entités pour gagner du pouvoir dans un système décentralisé, et la PoW réduit considérablement les conditions de course qui font partie inhérente du processus d'accession à une structure de données globale.

Un protocole alternatif de consensus, la preuve d'enjeu (PoS en anglais), a été d'abord introduit par Peercoin en 2012 [5]. Dans un système de PoS, les votes des participants sont



(a) Quand aucun conflit n'est détecté, aucun frais supplémentaire n'est nécessaire.

(b) Dans le cas d'une transaction conflictuelle, les noeuds votent pour la transaction valide.

Fig. 1. RaiBlocks ne nécessite aucun coût additionnel pour des transactions normales. Dans le cas d'une transaction conflictuelle , des noeuds doivent voter pour maintenir la transaction ou non.

pondérés en fonction des montants qu'ils possèdent pour une certaine crypto-monnaie. Avec cet arrangement, un pouvoir accru est donné à ceux qui ont un plus grand investissement financier et ces derniers sont incités à maintenir l'honnêteté du système afin de ne pas courir le risque de perdre leurs investissements. La PoS n'est pas gourmande en électricité comme les gaspilleuses compétitions de puissance de calcul et a seulement besoin de logiciels légers tournant sur du matériel à faible consommation énergétique.

Le document original de RailBlocks et la première implementation beta furent publié en Décembre 2014, ce qui en fait une des premières crypto-monnaies basée sur un graphe orienté acylique (DAG en anglais) [6]. Peu de temps après, d'autres crypto-monnaies DAG ont commencé à se développer en particulier DagCoin/Byteball et IOTA [7], [8]. Ces crypto-monnaies basées sur un DAG ont brisé le moule de la chaîne de blocs, améliorant la performance système et la sécurité. Byteball arrive à un consensus en comptant sur une chaîne principale composée de témoins honnêtes, réputés et approuvés par les utilisateurs, alors que IOTA arrive à un consensus via la PoW cumulée des transactions empilées. RaiBlocks arrive à un consensus sur les transactions conflictuelles via un vote pondéré en fonction du solde du compte. Ce type de consensus permet des transactions plus rapides et plus déterministes tout en maintenant un système fort et décentralisé. RaiBlocks continue ce développement et s'est positionné comme une des crypto-monnaies les plus performantes.

III. Composant de RaiBlocks

Avant de décrire l'architecture globale de RaiBlocks, nous allons définir les éléments individuels qui composent le système.

A. Compte

Un compte est la clé publique et elle est une des deux parties de la pair de clés composant la signature numérique. La clé publique, aussi appelée l'adresse, est partagée avec les autres participants du réseau alors que la clé privée est gardée secrète. Un paquet de données signé numériquement garantie que le contenu a été approuvé par le détenteur de la clé privée. Un utilisateur peut contrôler plusieurs comptes mais une seule adresse publique existe par compte.

B. Bloc/Transaction

Les termes "bloc" et "transaction" sont souvent utilisés indistinctement, quand un bloc contient une seule transaction. Le terme transaction renvoie spécifiquement à l'action alors que le terme bloc renvoie lui à l'encodage numérique de la transaction. Les transactions sont signées par la clé privée appartenant au compte sur lequel la transaction est effectuée.

C. Registre

Le registre est l'ensemble global de tous les comptes dans lequel chaque compte a sa propre chaîne de transaction (Figure 2). C'est un élément clé de la conception qui consiste à remplacer un accord run-time en accord design-time ; tout le monde se met d'accord par signature afin de confirmer que seul le propriétaire d'un compte peut modifier sa propre chaîne. Ceci converti un ensemble de données qui semble partagé, un registre partagé, en un ensemble de données non partagé.

D. Noeud

Un *noeud* est un logiciel tournant sur un ordinateur qui se conforme au protocole RaiBlocks et qui participe au réseau RaiBlocks. Le logiciel gère le registre et les comptes que ce noeud contrôle. Un noeud peut soit stocker le registre entier soit une version réduite de l'historique contenant seulement les quelques derniers blocs de la chaîne de blocs de chaque compte. Lors de l'installation d'un nouveau noeud il est recommandé de vérifier tout l'historique et de le réduire localement.

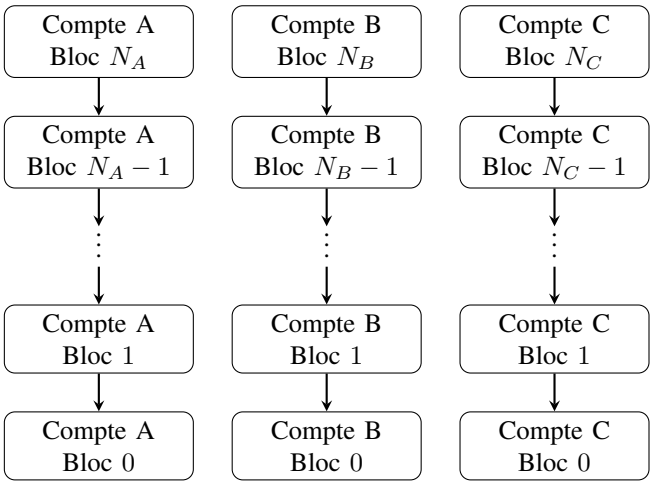


Fig. 2. Chaque compte à sa propre chaîne de blocs contenant l'historique de la balance du compte. Bloc 0 doit être une transaction ouverte(Section IV-B)

IV. Vue d'ensemble du système

Contrairement à bien d'autres crypto-monnaies Raiblocks utilise une structure de *blocs-tréssés* (block-lattice en anglais). Chaque compte possède sa propre chaîne de blocs, qui résume l'historique du solde et des transactions(Figure 2). Chaque compte-chaîne ne peut être mis à jour que par le propriétaire du compte; cela permet au compte-chaîne d'être immédiatement mis-à-jour et de manière asynchrone du reste des blocs tressés, résultant en des transactions rapides. Le protocole de Raiblocks est extrêmement léger; chaque transaction respecte la taille minimale des paquets UDP pour être transmis sur internet. L'équipement pré-requis pour les noeuds est minime lui aussi, puisque les noeuds ne doivent que répertorier et redistribuer les blocs pour la plupart des transactions. (Figure 1).

Le système est activé grâce à un *compte d'initialisation* contenant le *solde initial*. Le solde initial a une quantité fixe et ne pourra être augmenté. Le solde initial est divisé et envoyé vers d'autres comptes via des transactions d'envoi enregistrées sur le compte-chaîne d'initialisation. La somme des soldes de tous les comptes n'excédera jamais le solde initial ce qui donne au système une borne supérieure sur la quantité et aucune façon de l'augmenter.

Cette section expliquera comment les différents types de transactions sont construits et propagés sur le réseau.

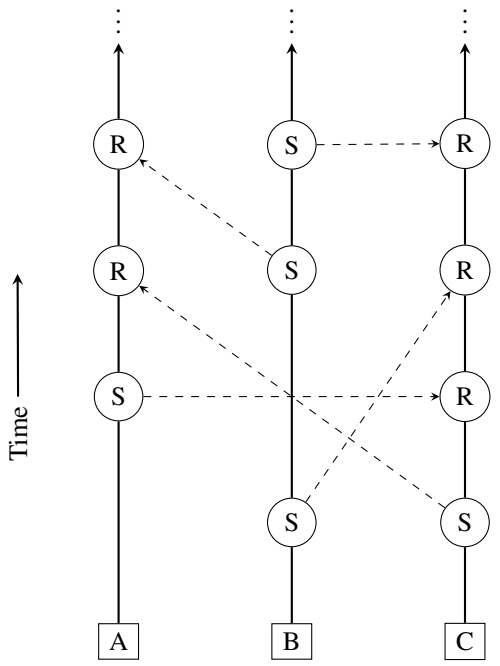


Fig. 3. Visualisation des blocs-tressés. Chaque transfert de fonds requiert un bloc d'envoi (S) et un bloc de réception (R) chacun signé par les propriétaires de compte-chaîne (A,B,C,)

A. Transactions

Le transfert de fonds d'un compte à un autre requiert deux transactions : un *envoi* déduisant le montant du solde du compte émetteur et un *récepteur* ajoutant le montant au solde du compte récepteur. (Figure 3).

Transférer des montants en tant que transactions séparées sur les soldes des émetteurs et des récepteurs a quelques objectifs importants :

1) Séquencer les transferts qui sont inhéremment asynchrones.
2) Garder les transactions adaptées aux paquets UDP.
3) Faciliter l'élagage (pruning en anglais) en minimisant les empreintes de données.
4) Isoler les transactions réglées des non réglées.

Une opération où plusieurs comptes transfèrent vers la même destination est une opération asynchrone; la latence du réseau et le fait que les comptes émetteurs ne communiquent pas nécessairement entre eux signifie qu'il n'y a pas de façon universelle de trouver un consensus pour savoir quelle transaction a eu lieu en premier. L'addition étant associative, la façon dont les entrées sont séquencées est sans importance, donc il ne nous faut en fait qu'un arrangement commun. C'est là un élément essentiel du design qui transforme un accord run-time (environnement d'exécution) en accord design-time (environnement de conception). Le compte récepteur décide de l'ordre d'arrivée des fonds en signant à son gré les blocs arrivants.

Dans le cas où un compte veut effectuer un transfert conséquent qui à été reçu en plusieurs séries de petits transferts, nous voulons présenter ça d'une façon qui corresponde au paquet UDP. Quand un compte récepteur séquence des transferts d'entrées, il garde disponible le total du solde de son compte afin qu'à n'importe quel moment il ait la capacité de transférer n'importe quelle somme grâce à une transaction à taille fixe. Ceci est différent du modèle de transaction entrée/sortie qu'utilisent Bitcoin et d'autres crypto-monnaies.

Certains noeuds ne désirent pas dépenser de ressources pour stocker l'historique entier des transactions d'un compte; ils ne sont intéressés que par le solde actuel de chaque compte. Quand un compte effectue une transaction, il encode son solde accumulé et ces noeuds ont seulement besoin de répertorier le dernier bloc, qui leur permet d'ignorer les données historiques tout en restant correct.

Même en se concentrant sur les accords design-time, il y a un laps de temps lorsque l'on valide les transactions dû à l'identification et la gestion des éléments malveillants du réseau. Puisque trouver un consensus sur Raiblocks se fait rapidement, de l'ordre de quelques millisecondes à quelques secondes, nous pouvons présenter à l'utilisateur deux catégories familières de transaction arrivantes : réglées et non réglées. Les transactions réglées sont des transactions où le compte a généré des blocs récepteurs. Les transactions non réglées ne sont pas encore incorporées dans le solde cumulé du récepteur. Ceci remplace les métriques de confirmation bien plus complexes et moins familières pratiquées par d'autres crypto-monnaies.

B. Création de compte

Pour créer un compte, il faut émettre une transaction d'*ouverture* (Figure 4). Une transaction d'ouverture est toujours la première transaction de chaque compte-chaîne et peut être créée dès la première réception de fonds. Le champ

compte stocke la clé publique (adresse) qui est utilisée pour signer. Le champ *source* contient le hash de la transaction qui a envoyé les fonds (Section IV-F). Le compte peut se déclarer comme étant son propre représentant.

```
open {
   account: DC04354B1...AE8FA2661B2,
   source: DC1E2B3F7C...182A0E26B4A,
   representative: xrb_1anr...posrs,
   work: 0000000000000000,
   type: open,
   signature: 83B0...006433265C7B204
}
```

Fig. 4. Anatomie d'une transaction d'ouverture.

C. Solde du compte

Le solde des comptes est enregistré sur le registre lui-même. Plutôt que d'enregistrer le montant d'une transaction, la vérification requiert de vérifier la différence entre le solde du bloc d'envoi et la balance du bloc précédent. (Section IV-I) . Le compte récepteur peut ensuite inclure le solde précédent tel que mesuré par le solde donné dans le nouveau bloc de réception. Ceci est fait pour améliorer la vitesse des opérations lors de téléchargements de blocs de gros volume. Lors de la requête d'historique, les montants sont déjà donnés.

D. Envoyer depuis un compte

Pour envoyer depuis une adresse, elle doit déjà avoir un bloc d'ouverture existant, et donc un solde. Le champ (Figure 5) *précédent* contient le hash du bloc précédent dans la chaîne compte. Le champ *destination* contient le compte vers lequel les fonds seront envoyés. Un bloc d'envoi est immuable une fois confirmé. Une fois émis sur le réseau, les fonds sont immédiatement déduits du solde de l'émetteur et sont en *attente* de confirmation jusqu'à ce que le parti récepteur signe un bloc pour accepter ces fonds. Les fonds en confirmation ne devraient pas être considérés comme en attente de confirmation pour l'expéditeur car ils sont déjà envoyés par l'émetteur et l'émetteur ne peut pas révoquer sa transaction.

```
send {
   previous: 1967EA355...F2F3E5BF801,
   balance: 010a8044a0...1d49289d88c,
   destination: xrb_3w...m37goeuufdp,
   work: 0000000000000000,
   type: send,
   signature: 83B0...006433265C7B204
}
```

Fig. 5. Anatomie d'une transaction d'envoi

E. Recevoir une Transaction

Pour finaliser une transaction, le bénéficiaire des fonds envoyés doit créer un bloc de réception sur son propre compte-chaîne (Figure 6). Le champ source référence le hash de la transaction associée qui a été émise. Quand ce bloc est crée et émis, le montant du compte est mis à jour et les fonds sont officiellement reçus sur le compte.

```
receive {
   previous: DC04354B1...AE8FA2661B2,
   source: DC1E2B3F7C6...182A0E26B4A,
   work: 0000000000000000,
   type: receive,
   signature: 83B0...006433265C7B204
}
```

Fig. 6. Anatomie d'une transaction de réception

F. Désigner un délégué

Les titulaires de comptes ayant la possibilité de choisir un délégué pour voter en leur noms, cela représente un outil de décentralisation puissant qui n'a pas de véritable équivalent dans les protocoles de preuve de travail (PoW) ou de preuve d'enjeu (PoS). Dans les systèmes conventionnels de preuve d'enjeu, le noeud du titulaire du compte doit être activé pour participer au vote. Faire tourner un noeud continuellement n'est pas pratique pour de nombreux utilisateurs; donner à un représentant le pouvoir de voter au nom d'un compte assoupli cette responsabilité. Les titulaires de comptes ont la possibilité de ré-attribuer leurs votes à n'importe quel compte et à n'importe quel moment. Une transaction *change* modifie le représentant d'un compte en soustrayant le poids du vote de l'ancien représentant et en ajoutant ce poids au nouveau représentant. (Figure 7). Aucun fond n'est mobilisé dans cette transaction et le représentant ne peux pas dépenser les fonds du compte qu'il représente.

```
change {
   previous: DC04354B1...AE8FA2661B2,
   representative: xrb_1anrz...posrs,
   work: 0000000000000000,
   type: change,
   signature: 83B0...006433265C7B204
}
```

Fig. 7. Anatomie d'une transaction change

G. Fourchettes et votes

Une fourchette a lieu lorsque les blocs $b_1, b_2, \ldots, b_j$ signés par j déclarent le même bloc comme étant leur prédécesseur (Figure 8). Ces blocs causent un conflit sur le statut d'un compte et doivent être résolus. Seulement le titulaire du compte à la possibilité de signer des blocs sur son compte-chaîne, donc une fourchette est forcément le résultat d'une

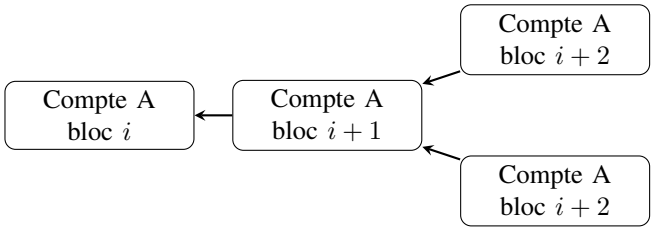


Fig. 8. Une fourchette à lieu quand deux blocs (ou plus) signés font référence au même bloc précédent. Les anciens blocs sont sur la gauche; les nouveaux blocs sur la droite.

mauvaise programmation ou d'une intention malveillante (double dépense) de la part du titulaire du compte.

Au moment de la détection, un représentant va créer un vote référant le bloc $\hat{b}_i$ dans son registre et va le diffuser au réseau. Le poids de vote d'un noeud, w_i, est la somme du solde de tous les comptes l'ayant nommé comme représentant. Le noeud observera les votes entrants provenant des autres M représentants en ligne et gardera un pointage cumulatif pour quatre périodes de votes, 1 minute au total, puis confirmera le bloc gagnant. (Équation 1).

$$v(b_j) = \sum_{i=1}^{M} w_i \mathbb{1}_{\hat{b}_i = b_j} \tag{1}$$

$$b^* = \arg\max_{b_j} v(b_j) \tag{2}$$

Le bloc le plus populaire b^* obtiendra la majorité des votes et sera conservé dans le registre du noeud (Équation 2). Les blocs perdants seront détruits. Si un représentant remplace un bloc dans son registre, il créera un nouveau vote avec un nombre de séquences plus haut et émettra le nouveau vote sur le réseau. C'est le **seul** scénario dans lequel les représentants votent.

Dans certaines circonstances, de brefs problèmes de connectivité peuvent amener un bloc émis à ne pas être accepté par ses homologues. Les blocs subséquents sur ce compte seront récupérés automatiquement. Une ré-émission de ce bloc sera acceptée par ses homologues. Même lorsqu'une fourchette se forme ou que l'on remarque un bloc manquant, seuls les comptes concernés dans la transaction sont affectés ; le reste du réseau continue d'opérer pour tous les autres comptes.

H. Preuve de travail (PoW)

Les quatre types de transactions ont un champ qui doit être correctement formulé. Le champ travail permet aux créateurs de transactions d'informatiser un nonce de façon à ce que le hash de ce nonce concaténé avec ses champs précédents réception/envoi/change ou que le champ compte dans une transaction d'ouverture soit en dessous d'une certaine valeur. Contrairement au Bitcoin, la PoW dans Raiblocks est simplement utilisée comme un outil anti-spam, similaire à Hashcash, et peut être calculé en quelques secondes [9]. Une fois que la transaction est envoyée, la preuve de travail pour le bloc subséquent peut être pré-calculée puisque le champ bloc précédent est connu; les transactions auront l'air instantanées pour les utilisateurs finaux tant que le temps entre les transactions est plus long que le temps qu'il faut pour calculer la preuve de travail.

I. Vérification des transaction

Pour qu'un bloc soit considéré comme valide, il faut qu'il ait les attributs suivants:

1) Le bloc ne doit pas déjà être dans le registre (transaction dupliquée)
2) Il doit être signé par le propriétaire du compte.
3) Le bloc précédent est le bloc de tête de la compte-chaîne. S'il existe mais n'est pas la tête, alors c'est une fourchette.
4) Le compte doit avoir un bloc d'ouverture.
5) Le hash calculé doit respecter le seuil pré-requis de PoW

Si c'est un bloc de réception, il faut vérifier si la source du bloc hash est en attente, voulant dire qu'il n'à pas encore été réclamé. Si c'est un bloc d'envoi, le solde doit être inférieur au solde précédent.

V. Types d'attaque

RaiBlocks, comme toutes les crypto-monnaies décentralisées, peut être attaqué par des entités malveillantes essayant d'obtenir un gain financier ou désirant la disparition du système. Dans cette section nous soulignons quelques uns des possibles scénarios d'attaques, les conséquences d'une telle attaque, et quelles sont les mesures préventives prisent par le protocole RaiBlocks.

A. Block Gap Synchronization

Dans la section IV-G,nous avons parlé du scénario dans lequel un bloc peut ne pas être correctement diffusé, causant le réseau à ignorer les blocs suivants. Si un nœud observe un bloc qui n'a pas de bloc précédent référencé, il a deux options:

1) Ignorer le bloc car il pourrait être malveillant.
2) Demander une resynchronisation avec un autre nœud.

Dans le cas d'une resynchronisation, une connexion TCP doit être établie avec un nœud d'amorçage afin de faciliter l'augmentation du trafic nécessité par la resynchronisation. Cependant, si le bloc était vraiment un bloc non valide, alors la resynchronisation ne serait pas nécessaire tout comme l'augmentation du trafic sur le réseau. Ceci est une attaque via une inondation de réseau et résulte en un déni de service.

Pour éviter une resynchronisation non nécessaire, les nœuds attendront jusqu'à ce qu'un certain seuil de votes soit observé pour un potentiel bloc malveillant avant de commencer une connexion avec un nœud d'amorçage pour se synchroniser. Si un bloc ne reçoit pas assez de votes, il peut considéré comme étant une donnée indésirable.

B. Surcharge de transaction

Une entité malveillante pourrait envoyer de nombreuses transactions inutiles entre des comptes étant sous son contrôle afin d'essayer de saturer le réseau. Sans la présence de frais de transaction, ils seront capables de continuer cette attaque

indéfiniment. Cependant, le PoW exigé pour chaque transaction limite le taux de transaction que l'entité malveillante pourrait générer sans investir massivement dans des ressources informatiques. Même avec ce type d'attaque dont le but est de gonfler le registre, les nœuds qui ne sont pas des nœuds historiques complets sont capables d'enlever les anciennes transactions de leur chaîne ; Cela limite l'utilisation du stockage de ce type d'attaque pour presque tous les utilisateurs.

C. Attaque Sybil

Une entité pourrait créer des centaines de nœuds RaiBlocks sur une seule machine. Cependant, puisque le système de vote est pondéré en fonction de la taille des soldes des comptes, ajouter des nœuds supplémentaires au réseau ne donnera pas à l'attaquant plus de votes. Donc il n'y a rien à gagner via une attaque Sybil.

D. Attaque Penny-spend

Une attaque de type penny-spend se produit lorsqu'un attaquant dépense des sommes infinitésimales sur un très grand nombre de comptes afin de gaspiller les ressources de stockages des nœuds. La publication des blocs se fait avec un débit limité en raison du système PoW. Dans une certaine mesure cela limite le nombre de créations de comptes et de transactions. Les nœuds qui ne sont pas des nœuds historiques complets peuvent supprimer des comptes dès lors qu'ils sont détectés comme n'étant probablement pas des comptes valides. Au final, Raiblocks est réglé pour utiliser en permanence le minimum d'espace de stockage. Ainsi l'espace nécessaire pour stocker un compte supplémentaire est proportionnel à la taille d'un bloc d'ouverture + indexage $= 96\text{B} + 32\text{B} = 128\text{B}$. Cela revient à dire qu'1GB est capable de stocker 8 millions de comptes de type Penny-spend. Si les nœuds veulent en supprimer de façon plus agressive, ils peuvent calculer une distribution basée sur les fréquences d'accès et déléguer de façon occasionnelle les comptes utilisés pour ralentir le stockage.

E. Attaque pré-calculée du PoW

Étant donné que le propriétaire d'un compte sera seul capable d'ajouter un bloc à sa chaîne, des blocs séquentiels peuvent être calculés, conjointement avec leur PoW, avant d'être ajouté au réseau. Ici, les attaquants génèrent des myriades de blocs séquentiels, chacun d'eux ayant une valeur minimale, pendant une période de temps élargie. A un certain moment, les attaquants réalisent un une attaque par déni de service (DoS attack) en submergeant le réseau avec un nombre de transactions valides très important, que les autres nœuds vont tenter de traiter en leur répondant aussi rapidement que possible. C'est une version avancée de l'attaque décrite dans le paragraphe V-B. Une telle attaque ne pourrait fonctionner que sur un temps très court mais pourrait être utilisée conjointement avec d'autres types d'attaques telles que celle du type >50% Attaque (paragraphe V-F) afin d'en augmenter l'efficacité. Des techniques de limitation du taux de transaction ainsi que d'autres techniques sont actuellement à l'étude pour limiter ce genre de risque d'attaques.

F. Attaque de type >50%

La règle qui fait consensus pour Raiblocks est un système de votes pondérés. Si des attaquants sont capables d'obtenir plus de 50% de la capacité de vote, ils peuvent empêcher le réseau de parvenir à un consensus ce qui le bloquerait. Un attaquant peut aussi faire en sorte que le poids qu'il doit avoir soit plus faible en faisant en sorte que les bons nœuds ne puissent voter en leur faisant subir une attaque de type DoS. Afin de se prémunir de telles attaques, Raiblocks a pris les mesures suivantes :

1) La défense principale contre ce type d'attaque est que la pondération du vote est liée à l'investissement dans le système. Une personne détenant un compte est motivée pour maintenir l'intégrité du système afin de protéger son investissement.
2) Le coût de cette attaque est proportionnel à la valeur de la capitalisation de Raiblocks. Dans un système PoW, des technologies peuvent être inventées afin de donner un contrôle disproportionné en regard de l'investissement monétaire et si l'attaque est victorieuse, cette technologie peut être réutilisée une fois que l'attaque est terminée. Avec Raiblocks le coût nécessaire pour attaquer le système est équivalent au coût du système lui-même et si une attaque est victorieuse l'investissement qui a été nécessaire pour la réaliser ne peut être récupéré.
3) Dans le but de maintenir le nombre le plus important de votants, la ligne de défense est un système de vote représentatif. Les détenteurs d'un compte qui ne peuvent participer à un vote de façon fiable pour des problèmes de connexion peuvent nommer un représentant qui peut voter avec leur poids. Maximiser le nombre et la diversité des représentants augmente la résilience du réseau.
4) Les fourchettes de Raiblocks ne sont jamais accidentelles, ainsi les nœuds peuvent interagir en toute connaissance de cause avec ce genre de blocs. Le seul moment où les comptes non-attaquants sont vulnérables à ce genre de blocs est lorsqu'ils reçoivent un solde d'un compte attaquant. Les comptes voulant être sécurisés contre ces blocs peuvent attendre un peu ou beaucoup plus longtemps avant de recevoir d'un compte qui a généré des fourchettes ou décider de ne jamais les recevoir. Les destinataires pourraient également générer des comptes séparés à utiliser lors de la réception de fonds provenant de comptes douteux afin de les isoler.
5) Une dernière ligne de défense qui n'a pas encore été mise en œuvre est le *cimentage de blocs*. RaiBlocks se donne beaucoup de mal pour briser rapidement les fourchettes de blocs en votant. Les nœuds pourraient être configurés pour cimenter les blocs, ce qui les empêcherait de revenir en arrière après un certain laps de temps. En se concentrant sur un temps de stabilisation rapide le réseau est suffisamment sécurisé éviter les fourchettes ambiguës.

Une version plus sophistiquée d'une attaque $> 50\%$ est détaillée dans la Figure 9. "Offline" représente le pourcentage de représentants qui ont été nommés mais ne sont pas en

ligne pour voter. ”Stake” est le montant de l’investissement avec lequel l’attaquant vote. ”Active” sont les représentants qui sont en ligne et votent selon le protocole. Un attaquant peut modifier le montant stake qu’il doit falsifier en faisant tomber les autres électeurs hors-ligne via une attaque du réseau de type DoS. Si cette attaque peut être maintenue, les représentants attaqués deviendront non synchronisés et ceci est démontré par ”Unsync”. Enfin, un attaquant peut obtenir un pic de force de vote en transférant son attaque DoS à un nouvel ensemble de représentants alors que les anciens re-synchronisent leur registre, ceci est démontré par ”Attack”.

Offline	Unsync	**Attack**	Active	Stake

Fig. 9. Un accord de vote potentiel qui pourrait réduire les exigences d’attaque de 51%.

Si un attaquant est capable de provoquer Stake>Active par une combinaison de ces circonstances, il pourrait réussir à renvoyer des votes sur le registre au détriment de leur stake. Il est possible d’estimer combien ce type d’attaque pourrait coûter en examinant la capitalisation d’autres systèmes. Si nous estimons que 33% des représentants sont hors-ligne ou attaqués via un DoS, un attaquant devrait acheter 33% de la capitalisation pour attaquer le système via le vote.

G. Empoisonnement bootstrap

Plus un attaquant est en mesure de détenir une ancienne clé privée avec un solde, plus la probabilité que les soldes qui existaient à ce moment-là n’aient pas de représentants participants est élevée parce que leurs soldes ou représentants ont été transférés vers des comptes plus récents. Cela signifie que si un nœud est amorcé à une ancienne représentation du réseau où l’attaquant a un quorum de vote par rapport aux représentants à ce moment-là, ils pourraient faire osciller les décisions de vote sur ce nœud. Si ce nouvel utilisateur souhaitait interagir avec n’importe qui d’autre que le nœud attaquant, toutes ses transactions seraient refusées car elles ont des blocs de tête différents. Le résultat est que les nœuds peuvent faire perdre du temps aux nouveaux nœuds du réseau en leur fournissant de mauvaises informations. Pour éviter cela, les nœuds peuvent être associés à une base de données initiale de comptes et à des têtes de blocs connues. Cela permet de télécharger toute la base de données depuis le bloc d’origine. Plus le téléchargement est près d’être réalisé, plus la probabilité de défendre avec précision contre cette attaque est élevée. En fin de compte, cette attaque n’est probablement pas pire que d’alimenter les nœuds avec des données indésirables lors de l’amorçage, car ils ne seraient pas en mesure de traiter avec quiconque possédant une base de données contemporaine.

VI. Implementation

Actuellement, l’implémentation de référence est le C ++ et des versions sont créées depuis 2014 sur Github. [10].

A. Caractéristiques de conception

La mise en œuvre de RaiBlocks adhère à la norme d’architecture décrite dans ce document. Des spécifications supplémentaires sont décrites ici.

1) Algorithme de signature: RaiBlocks utilise un algorithme de courbe elliptique ED25519 modifié avec le hachage Blake2b pour toutes les signatures numériques [11]. ED25519 a été choisi pour ces qualités de signature et de vérification rapide ainsi que pour son niveau de sécurité élevé.

2) Algorithme de hachage: Étant donné que l’algorithme de hachage est uniquement utilisé pour empêcher les spams sur le réseau, son choix est moins important par rapport aux crypto-monnaies qui sont minées. Notre implémentation utilise Blake2b comme algorithme d’assimilation contre le contenu des blocs [12].

3) Fonction de dérivation de clé: Dans le portefeuille de référence, les clés sont encryptées par un mot de passe et le mot de passe est protégé par une fonction de dérivation de clé pour se protéger contre les tentatives de piratage ASIC. Actuellement Argon2 [13] est le gagnant de la seule compétition publique visant à créer une fonction de dérivation de clé résiliente.

4) Intervalle des blocs: Étant donné que chaque compte a sa propre chaîne, les mises à jour peuvent se faire de façon asynchrone sur le réseau. Par conséquent, il n’y a pas de temps morts entre les blocs et les transactions peuvent être publiées instantanément.

5) Protocole de message UDP: Notre système est conçu pour fonctionner indéfiniment en utilisant le minimum de ressources informatiques. Tous les messages dans le système ont été conçus pour être sans état et s’intégrer dans un seul paquet UDP. Cela facilite également la participation intermittente des pairs dans le réseau sans rétablir les connexions TCP à court terme. Le TCP est utilisé uniquement pour les nouveaux pairs quand ils veulent bootstrap les chaînes de blocs de façon importante.

Les nœuds peuvent être sûrs que leur transaction a été reçue par le réseau en observant les transactions provenant d’autres nœuds. En effet, ils devraient pouvoir en voir plusieurs copies.

B. Multidiffusion IPv6

Construire le protocole UDP sans connexion permet aux futures implémentations d’utiliser la multidiffusion IPv6 en remplacement de l’afflux traditionnel de transactions et de la diffusion des votes. Cela permettra de réduire la consommation de bande passante du réseau et de donner plus de capacité d’adaptation pour les nœuds.

C. Performance

À ce jour, 4,2 millions de transactions ont été traitées par le réseau RaiBlocks, ce qui donne une taille de la chaîne de blocs de 1,7 Go. Les temps de transaction sont mesurés de l’ordre de quelques secondes. Une mise en œuvre de référence actuelle fonctionnant sur des disques SSD peut traiter plus de 10 000 transactions par seconde, principalement liées aux IO.

VII. Utilisation des ressources

Ceci est une vue d'ensemble des ressources utilisées par un nœud RaiBlocks. De plus, nous examinons des idées pour réduire l'utilisation des ressources pour des cas d'utilisation spécifiques. Les nœuds réduits sont généralement appelés nœuds de vérification de paiement (SPV) simplifiés, tronqués ou simplifiés.

A. Réseau

La quantité d'activité sur le réseau dépend de la contribution du réseau à sa santé.

1) Représentatif: Un nœud représentatif nécessite un maximum de ressources réseau car il observe le trafic de votes d'autres représentants et publie ses propres votes.

2) Sans Confiance: Un nœud sans confiance est similaire à un nœud représentatif à la différence qu'il n'est qu'un observateur. Il ne détient pas de clé privée de compte représentatif et ne publie pas de ses propres votes.

3) Confiance: Un nœud de confiance observe le trafic de votes d'un représentant auquel il fait confiance pour effectuer correctement le consensus. Cela réduit la quantité de trafic de votes entrants provenant des représentants se rendant sur ce nœud.

4) Léger: Un nœud léger est également un nœud de confiance mais qui n'observe le trafic que pour les comptes dans lesquels il est intéressé, ce qui permet une utilisation minimale du réseau.

5) Bootstrap: Un nœud d'amorçage sert une partie ou l'intégralité du registre pour les nœuds qui se mettent en ligne. Ceci est fait sur une connexion TCP plutôt qu'UDP car cela implique une grande quantité de données qui nécessite un contrôle de flux avancé.

B. Capacité du disque

En fonction des demandes de l'utilisateur, différentes configurations des nœuds nécessitent des exigences de stockage différentes.

1) Historique: Un nœud désirant conserver un historique complet de toutes les transactions aura besoin de la quantité maximale de stockage.

2) Actuel: En raison du choix fait de conserver les soldes accumulés avec les blocs, les nœuds n'ont besoin de conserver que le plus récent ou le bloc de tête pour chaque compte dans le but de participer au consensus. Si un nœud ne désire pas conserver l'historique complet il peut choisir de ne conserver que les blocs de têtes.

3) Léger: Un nœud léger ne conserve aucune donnée du registre local et ne participe au réseau que pour observer l'activité sur les comptes dans lesquels il a des intérêts ou, éventuellement, créer de nouvelles transactions avec des clés privées qu'il détient.

C. CPU

1) Génération de transaction: Un nœud désirant créer de nouvelles transactions doit produire un nonce de preuve de travail afin de réussir le mécanisme de limitation de RaiBlocks. Les performances de divers matériels sont référencés en annexe A.

2) Représentant: Un représentant doit vérifier les signatures pour les blocs, les votes, et aussi produire ses propres signatures pour participer au consensus. La quantité de ressources processeur pour un nœud représentatif est significativement inférieure à la génération d'une transaction et devrait fonctionner avec n'importe quel processeur d'ordinateur contemporain.

3) Observateur: Un nœud observateur ne génère pas ses propres votes. Étant donné que la surcharge engendrée par la génération d'une signature est minimale, les exigences du CPU sont presque identiques à l'exécution d'un nœud représentatif.

VIII. Conclusion

Dans cet article, nous présentons le cadre d'une cryptomonnaie sans frais, sans recours à un tiers de confiance, et à faible latence, qui utilise une nouvelle structure en blocs-tressés et un système de vote délégué de type PoS. Le réseau ne nécessite que des ressources minimales, aucun matériel de minage énergivore et peut traiter un débit de transactions élevé. Tout cela est réalisé en ayant des chaînes de blocs individuels pour chaque compte, éliminant les problèmes d'accès et les inefficacités d'une structure de données globale. Nous avons identifié des formes d'attaques possibles sur le système et présenté des arguments sur la résistance de RaiBlocks à ces formes d'attaques.

Appendix A
Performances PoW des matériels

Comme mentionné précédemment, le but du PoW dans RaiBlocks est de réduire les spams sur le réseau. Notre implémentation de nœud fournit une accélération qui peut tirer parti des GPU compatibles OpenCL. La table I fournit une comparaison de référence réelle de divers matériels. Actuellement, le seuil de PoW est fixe, mais un seuil adaptatif peut être mis en œuvre étant donné quela puissance de calcul moyenne progresse.

TABLE I
Performance de PoW des matériels

Matériel	Transactions par second
Nvidia Tesla V100 (AWS)	6.4
Nvidia Tesla P100 (Google,Cloud)	4.9
Nvidia Tesla K80 (Google,Cloud)	1.64
AMD RX 470 OC	1.59
Nvidia GTX 1060 3GB	1.25
Intel Core i7 4790K AVX2	0.33
Intel Core i7 4790K,WebAssembly (Firefox)	0.14
Google Cloud 4 vCores	0.14-0.16
ARM64 server 4 cores (Scaleway)	0.05-0.07

Remerciement

Nous aimerions remercier Brian Pugh d'avoir compilé et formaté ce document.

References

[1] S. Nakamoto, "Bitcoin: A peer-to-peer electronic cash system," 2008. [Online]. Available: http://bitcoin.org/bitcoin.pdf

[2] "Bitcoin median transaction fee historical chart." [Online]. Available: https://bitinfocharts.com/comparison/bitcoin-median_transaction_fee.html

[3] “Bitcoin average confirmation time.” [Online]. Available: https://blockchain.info/charts/avg-confirmation-time
[4] “Bitcoin energy consumption index.” [Online]. Available: https://digiconomist.net/bitcoin-energy-consumption
[5] S. King and S. Nadal, “Ppcoin: Peer-to-peer crypto-currency with proof-of-stake,” 2012. [Online]. Available: https://peercoin.net/assets/paper/peercoin-paper.pdf
[6] C. LeMahieu, “Raiblocks distributed ledger network,” 2014.
[7] Y. Ribero and D. Raissar, “Dagcoin whitepaper,” 2015.
[8] S. Popov, “The tangle,” 2016.
[9] A. Back, “Hashcash - a denial of service counter-measure,” 2002. [Online]. Available: http://www.hashcash.org/papers/hashcash.pdf
[10] C. LeMahieu, “Raiblocks,” 2014. [Online]. Available: https://github.com/clemahieu/raiblocks
[11] D. J. Bernstein, N. Duif, T. Lange, P. Shwabe, and B.-Y. Yang, “High-speed high-security signatures,” 2011. [Online]. Available: http://ed25519.cr.yp.to/ed25519-20110926.pdf
[12] J.-P. Aumasson, S. Neves, Z. Wilcox-O’Hearn, and C. Winnerlein, “Blake2: Simpler, smaller, fast as md5,” 2012. [Online]. Available: https://blake2.net/blake2.pdf
[13] A. Biryukov, D. Dinu, and D. Khovratovich, “Argon2: The memory-hard function for password hashing and other applications,” 2015. [Online]. Available: https://password-hashing.net/argon2-specs.pdf