Registre de comptabilité à résistance quantique (Quantum Resistant Ledger - QRL)

peterwaterland@gmail.com

Novembre 2016

Introduction à la traduction

Cette traduction a été faite par Claire Reid et commissionnée par l'investisseur de QRL, Justin Poirier, fondateur de Tokenized Capital. Veuillez vous référer aux documents originaux en langue anglaise et considérez ceux-ci comme officiels s'il existe des divergences entre les traductions. Un addendum a été ajouté de manière à informer les lecteurs au sujet des changements relatifs à l'émission de pièces qui sont entrés en vigueur après la publication de ce livre blanc. Une prévente concluante a été tenue et 4,16 millions de dollars US ont été recueillis en bitcoins et ether. Le contenu du livre blanc original n'a pas été modifié à titre d'information.

Résumé

Les monnaies numériques privées doivent être sécurisées contre les progrès informatiques pour une meilleure longévité. La conception et l'émission d'un registre de crypto-monnaie utilisant des signatures numériques basées sur le hachage et résistantes aux attaques informatiques classiques et quantiques sont présentées.

1 Introduction

Le concept d'un registre par internet comptabilisant les valeurs de pair-à-pair, sous forme de chaîne de blocs (blockchain) et sécurisé par une preuve de travail a été initialement signalé en 2008 [11]. Le Bitcoin reste la crypto-monnaie la plus répandue à ce jour. Des centaines de registres de crypto-monnaie similaires ont été créés ultérieurement, mais à quelques exceptions près, ils utilisent la même cryptographie à clé publique de courbe elliptique (ECDSA) pour générer des signatures numériques qui permettent de vérifier les transactions de manière sécurisée. Les schémas de signature les plus couramment utilisés aujourd'hui, tels que ECDSA, DSA et RSA, sont théoriquement vulnérables à l'attaque informatique quantique. Il serait utile d'explorer la conception et la construction d'un registre de blocs résistant au quantique afin de contrer l'avènement potentiel d'une avancée soudaine et non linéaire de l'informatique quantique.

2 Sécurité des transactions en bitcoins

Il est actuellement seulement possible de dépenser (sorties de transactions non dépensées) d'une adresse bitcoin en créant une transaction contenant une signature de courbe elliptique valide (secp256k1) depuis la clé privée ($x \in N \mid x < 2^{256}$) pour cette adresse bitcoin spécifique. Si la clé privée générée de manière véritablement aléatoire est maintenue secrète ou perdue, aucun fonds ne peut raisonnablement passer depuis cette adresse.

La possibilité d'une collision de spécifiques clés privées de bitcoin est de 1 sur 2^{256}. La probabilité d'une collision de clé d'adresse de bitcoin peut être estimée en utilisant le "Problème de l'anniversaire". Le nombre d'adresses bitcoin qui doivent être générées pour donner lieu à une probabilité de 0,1 % de chance de collision est $5{,}4 \times 10^{23}$ [14].

Toutefois, lorsqu'une transaction est signée, la clé publique ECDSA de l'adresse d'envoi est révélée et stockée dans la blockchain. La meilleure pratique recommande de ne pas réutiliser les adresses, mais depuis novembre 2016, 49,58 % du solde comptable en bitcoins est détenu dans des adresses avec des clés publiques exposées [1].

3 Vecteurs d'attaque informatique quantique

RSA, DSA et ECDSA restent sécurisés selon la difficulté computationnelle de factorisation des grands nombres entiers, le problème du logarithme discret et le problème du logarithme discret à courbe elliptique. L'algorithme quantique de Shor (1994) résout la factorisation de grands nombres entiers et de logarithmes discrets en temps polynomial. Par conséquent, un ordinateur quantique pourrait théoriquement reconstituer la clé privée si on lui fournissait une clé publique ECDH ECDSA. ECDSA a la réputation d'être plus vulnérable aux attaques quantiques que RSA en raison de l'utilisation de plus petites clés, étant donné qu'un ordinateur quantique de 1300 et 1600 qubit (2^{11}) est capable de résoudre 228 bit ECDSA.

Le développement d'ordinateurs quantiques publics n'a pas dépassé 2^5 qubits ou la factorisation de petits nombres (15 ou 21). Cependant, en août 2015, la NSA a déconseillé la cryptographie à courbe elliptique ostensiblement selon des préoccupations d'informatique quantique. On ne sait pas à quel degré d'avance se trouve l'informatique quantique actuellement, ou si des progrès dans ce domaine seront publiés pour autoriser des protocoles cryptographiques dans l'utilisation commune de l'internet afin d'apporter une sécurité post-quantique. Avec des origines quelque peu contestataires, bitcoin pourrait se trouver être la première cible adversaire de l'ordinateur quantique.

Si une avancée significative de l'informatique quantique devait se produire publiquement, les développeurs de nœud pourraient appliquer des systèmes de signature cryptographique résistant au quantum dans les bitcoins et encourager tous les utilisateurs à déplacer leur solde depuis des adresses ECDSA vers de nouvelles adresses protégées contre le quantum. Pour réduire la proportion d'adresses affectées, il serait raisonnable de désactiver le recyclage des clés publiques au niveau du protocole. Une telle mise à niveau prévue entraînerait aussi le mouvement possible des 1 million de pièces appartenant à Satoshi Nakamoto - avec la volatilité associée des prix.

Un scénario moins favorable serait une avancée silencieuse de l'informatique quantique non linéaire suivie d'une attaque informatique quantique nuancée sur les adresses de bitcoin avec des clés publiques exposées. Ces vols pourraient avoir un effet dévastateur sur le cours des bitcoins en raison de la nouvelle pression de vente accrue et une perte complète de confiance dans le système lorsque l'ampleur des vols sera connue. Le rôle des bitcoins en tant que réserve de valeur (« l'or numérique ») pourrait être très gravement endommagé avec des conséquences extrêmes pour le monde entier. Dans ce contexte, les auteurs pensent qu'il est raisonnable d'expérimenter avec les signatures cryptographiques résistant au quantum dans un registre de crypto-monnaie et potentiellement créer une réserve de valeurs de sauvegarde en cas de « cygne noir ».

4 Signatures résistant au quantique

Il existe plusieurs systèmes cryptographiques importants qui sont censés être résistant au quantique : la cryptographie basée sur le hachage, la cryptographie basée sur le code, la cryptographie basée sur le réseau, la cryptographie basée sur les équations quadratiques multivariées et la cryptographie à clé secrète. Tous ces programmes sont supposés résister aux attaques informatiques, à la fois classique et quantiques, à partir du moment où les clés sont suffisamment longues.

Des programmes hautement sécurisés de signature numérique basés sur le hachage existent avec les exigences de sécurité minimales qui s'appuient uniquement sur la résistance aux collisions d'une fonction de hachage cryptographique. Changer la fonction de hachage sélectionnée produit un nouveau schéma de signature numérique basé sur le hachage. Les signatures numériques basées sur le hachage sont bien étudiées et représentent le principal candidat pour les signatures post-quantique à l'avenir. Comme telles, elles sont la classe choisie de signatures post-quantique pour le QRL.

5 Signatures numériques basées sur le hachage

Les signatures numériques basées sur le hachage et résistant au quantique s'appuient sur la sécurité d'une fonction de hachage cryptographique unidirectionnelle qui prend un message m et sort un condensé de hachage h de longueur fixe n, donc SHA-256, SHA-512. En utilisant une fonction de hachage cryptographique, il devrait être mathématiquement impossible de forcer m hors de h (résistance de pré-image), ou de forcer h hors de h_2, où $h_2 = hash(h)$ (résistance deuxième pré-image), alors qu'il devrait être très difficile de trouver deux messages ($m1 \neq m2$) qui produisent le même h (résistance de collision).

L'algorithme quantique de Groverpeut être utilisé pour tenter de trouver une collision de hachage ou effectuer une attaque pré-image pour trouver m, nécessitant des opérations $O(2^{n/2})$. Ainsi, pour maintenir une sécurité de 128 bits, il faut sélectionner une longueur de séparation de hachage n d'au moins 256 bits, en supposant une fonction de hachage cryptographique parfaite.

Les signatures numériques basées sur le hachage nécessitent une clé publique pk pour la vérification et une clé privée sk pour signer un message. Diverses signatures ponctuelles basées sur le hachage (OTS) seront discutées en ce qui concerne leur pertinence d'inclusion dans le cadre d'un registre de blocs de chaines.

5.1 Le schéma de signature OTS de Lamport-Diffie

En 1979, Lamport a décrit une signature unique basée sur le hachage pour un message de longueur m bits (généralement la sortie d'une fonction de hachage résistant aux collisions). La génération de paires de clés génère m paires de clés secrètes aléatoires, $sk_j^m \in \{0,1\}^n$ où $j \in \{0,1\}$ donc la clé privée est: $((sk = sk_0^1, sk_1^1), \ldots, (sk_0^m, sk_1^m))$. Soit f une fonction de hachage à sens unique $\{0,1\}^n \rightarrow \{0,1\}^n$, avec m paires de clés publiques générées $pk_j^m = f(sk_j^m)$, cela signifie que la clé publique est: $((pk = pk_0^1, pk_1^1), \ldots, (pk_0^m, pk_1^m))$. La signature implique une inspection par bit du hachage du message pour sélectionner sk_j (c'est-à-dire si $bit = 0$, $sk_j = sk_0$, $bit = 1$, $sk_j = sk_1$) créant la signature:

$s = (sk_j^1, \ldots, sk_j^m)$ qui révèle la moitié de la clé privée. Pour vérifier une signature, l'inspection par bit ($j \in \{0,1\}$) du hachage du message vérifie que $(pk_j = f(sk_j))^m$

En supposant que, d'après l'algorithme de Grover, une sécurité de 128 bits est souhaitée, où la longueur du message est une sortie de hachage fixe de SHA256, m = 256 et n = 256, ce qui donne $pk = sk$ = 16kb et une signature de 8kb pour chaque OTS utilisé. Une signature de Lamport ne doit être utilisée qu'une seule fois, peut être générée très rapidement, mais elle pâtit de la grande taille de clé, pour la signature et, par extension, pour la taille des transactions, ce qui la rend peu pratique pour un registre public de blockchains.

5.2 Le schéma de signatures OTS de Winternitz

Pour un résumé de message *M* de longueur m bits, avec des clés secrètes et publiques de longueur, n bits, une fonction à sens unique, $f: \{0,1\}^n \rightarrow \{0,1\}^n$ et un paramètre de Winternitz, $w \in N \mid w > 1$, l'idée générale de la signature unique de Winternitz est d'appliquer une fonction de hachage itératif sur une liste de clés secrètes aléatoires, $sk \in \{0,1\}^n, sk = (sk_1, \ldots, sk_{\underline{w}})$, créant des chaînes de hachages de longueur, $w - 1$ se terminant par des clés publiques, $(pk \in N \mid \{0,1\}^n), pk_x = f^{2^w-1}(sk_x), pk = (pk_1, \ldots, pk_{\underline{w}})$.

Contrairement à l'inspection par bit du résumé du message dans la signature de Lamport, le message est analysé w bits à la fois pour extraire un nombre, $i \in N, i < 2^w - 1$, à partir duquel la signature est générée, $s_x = f^i(sk_x), s = (s_1 \ldots s_{\underline{x}})$. Avec un w croissant fournissant un compromis entre des clés et des signatures plus petites pour un plus grand effort computationnel. [10].

La vérification implique simplement de générer $pk_x = f^{2^w-1-i}(s_x)$ depuis M, s et de configurer la correspondance des clés publiques.

En utilisant SHA-2 (SHA-256) comme fonction de hachage cryptographique à sens unique, f: m = 256 et n = 256, avec w = 8 donne une taille de pk = sk = s de $\frac{(\underline{x})n}{8}$ octets = 1ko.

Pour générer pk, il faut des itérations de hachage f^i, où $i = \frac{m}{w}2^{w-1} = 8160$ par génération de paire de clés OTS. À w = 16 les clés et les signatures ont diminué de moitié, mais i = 1048560 devient impraticable.

5.3 Le schéma OTS+ de Winternitz (W-OTS+)

Buchmann a introduit une variante du schéma OTS de Winternitz d'origine en modifiant la fonction itérative à sens unique pour être appliquée à un nombre aléatoire, x, à plusieurs reprises, mais cette fois paramétrée par une clé k générée à partir de l'itération précédente de $f_k(x)$. Ceci est hautement infalsifiable dans les attaques de messagerie adaptées choisies lors de l'utilisation d'une fonction pseudo-aléatoire (PRF) et une épreuve de sécurité peut être calculée pour des paramètres donnés [3]. Il élimine la nécessité d'une famille de fonctions de hachage résistantes aux collisions en effectuant une navigation aléatoire à travers la fonction au lieu d'une simple itération. Huelsing a introduit une autre variante W-

OTS+ permettant de créer des signatures plus petites pour une sécurité de bit équivalente grâce à l'ajout d'un bitmask XOR dans la fonction de chaînage itératif. [6] Une autre différence entre W-OTS (variante 2011) / W-OTS et W-OTS+ est que le message est analysé $\log_2(w)$ bits à la fois plutôt que w, en diminuant les itérations de la fonction de hachage mais en augmentant les tailles de clés et de signatures.

Nous décrirons maintenant W-OTS+. Avec le paramètre de sécurité, $n \in \mathbb{N}$, correspondant à la longueur du message (n), des clés et de la signature en bits, étant déterminé par la fonction de hachage cryptographique choisie et le paramètre Winternitz, $w \in \mathbb{N} \mid w \geq 1$ (habituellement {4, 16}) l est calculé. l est le nombre d'éléments de chaîne de *n* bits dans une clé ou une signature WOTS , où $l = l_1 + l_2$:

$$l_1 = \left\lceil \frac{n}{\log_2(w)} \right\rceil, l_2 = \left\lfloor \frac{\log_2(l_1(w-1))}{\log_2(w)} \right\rfloor + 1$$

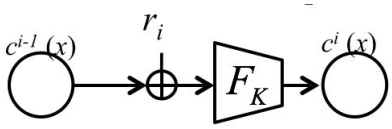
une fonction de hachage à clé est utilisée, $f_k : \{0,1\}^n \rightarrow \{0,1\}^n \mid k \in \{0,1\}^n$. En pseudo-code :

$$f_k(M) \quad Hash(Pad(K) \| Pad(M)$$

Où $Pad(x) \quad (x \| 10^{b-|x|-1})$ pour $| x | < b$.

La fonction de chaînage, $c_k^i(x, r)$: sur l'entrée de $x \in \{0,1\}^n$, compteur d'itération i, clé $k \in K$ et éléments de randomisation $r = (r_1, \ldots, r_j) \in \{0,1\}^{n \times j}$ où $j \geq$ i est défini comme suit :

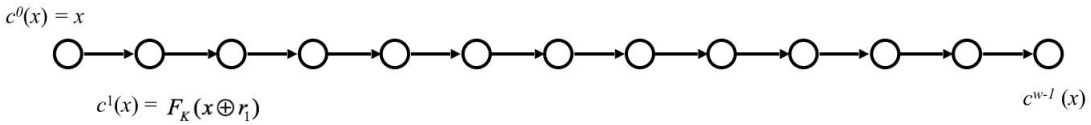
Figure 1: Fonction de chaînage W-OTS+



où:

$$c^i(x, r) = \begin{cases} x & si\ i = 0 \\ f_k(c_k^{i-1}(x, r) \oplus r_i) & si\ i > 0 \end{cases}$$

Figure 2: Exemple de génération de chaîne de hachage



C'est une opération de bits xor de l'itération précédente de c_k et l'élément de randomisation suivi de f_k sur le résultat, qui est ensuite introduit dans la prochaine itération de c_k.

5.3.1 Clé de signature

Pour créer la clé secrète, les chaînes sk, $l + w$ - 1 n bit sont uniformément choisies de manière aléatoire (avec FDP), où le premier l forme la clé secrète, $sk = (sk_1 \ldots sk_l)$ et le reste des chaînes w - 1 n bit deviennent $r = (r_1, \ldots, r_{w-1})$. Une clé de fonction k est choisie uniformément de manière aléatoire.

5.3.2 Clé de vérification

La clé publique est :

$$pk = (pk_0, pk_1, \ldots, pk_l) = ((r, k), c_k^{w-1}(sk_1, r), c_k^{w-1}(sk_2, r), \ldots, c_k^{w-1}(sk_l, r),$$

Remarquons que pk_0 contient r et k.

5.3.3 Signer

Pour effectuer une signature : message M de longueur m est analysé, tel que $M = M_1, \ldots M_{l_1}), M_i \in \{0, w-1\}$ (créant une représentation de base-w de M).

Ensuite la somme de contrôle *C* de longueur l_2 est calculée et ajoutée :

$$C = \sum_{i=1}^{l_1}(w - 1 - M_i$$

Tel que: $M + C = b = (b_0 \ldots b_l)$

La signature est :

$$s = (s_1, \ldots, s_l) = (c_k^{b_1}(sk_1, r), \ldots, c_k^{b_l}(sk_l, r))$$

5.3.4 Vérification

Pour vérifier une signature $b = (b_1, \ldots, b_l)$ est reconstruite à partir de M.

Si $pk = (c_k^{w-1-b_1}(s_1), \ldots, c_k^{w-l-b_l}(s_l))$, la signature est valide.

W-OTS + offre un niveau de sécurité d'au moins $n - w - 1 - 2\log(lw)$bits [3]. Une signature typique où w = 16 utilisant SHA-256 (n = m = 256) est in bits ou 2:1 Ko.

6 Schémas de signature de l'arbre de Merkle

Tandis que les signatures uniques assurent une sécurité cryptographique satisfaisante pour signer et vérifier les opérations, elles présentent l'inconvénient majeur de ne pouvoir être utilisées qu'une seule fois en toute sécurité. Si une adresse de registre repose sur une transformation de la clé publique d'une paire de clés unique OTS, cela conduirait à un registre de blockchains extrêmement restrictif où tous les fonds provenant d'une adresse d'envoi devraient être déplacés avec chaque transaction unique effectuée - sans quoi ces fonds seraient exposés au vol.

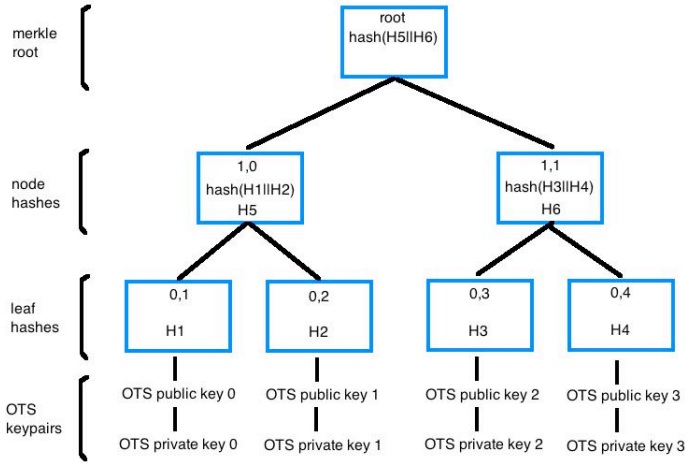
Une solution possible est d'étendre le schéma de signature afin d'incorporer plus d'une signature valide d'OTS pour chaque adresse de registre permettant autant de signatures que de paires de clés OTS sont prégénérées. Un arbre binaire de hachage connu sous le nom d'arbre de Merkle est un moyen logique d'atteindre cet objectif.

6.1 Arbre binaire de hachage

L'idée générale de l'arbre de Merkle est un arbre inversé composé de nœuds parents calculés en hachant la concaténation des nœuds frères enfant vers le haut en couches jusqu'à la racine. L'existence d'un nœud ou d'une feuille peut être prouvée par chiffrement en calculant la racine.

Un arbre de Merkle est formé de feuilles de base n et a une hauteur de racine Merkle, $h(n = 2^h)$- à partir des hachages de feuille (couche 0) et comptant vers le haut avec chaque couche de nœuds. Chaque nœud de feuille est créé dans notre cas d'utilisation hypothétique du registre en hachant une clé publique OTS prégénérée de manière aléatoire. Dans l'arbre ci-dessous, on peut voir que le nœud au-dessus de chaque paire de hachages de feuille est lui-même formé en hachant une concaténation des hachages de l'enfant.

Figure 3: Exemple du schéma de signature de l'arbre de Merkle



Cela se poursuit vers le haut à travers les couches de l'arbre jusqu'à la confluence dans le hachage de la racine de l'arbre, connu sous le nom de racine de Merkle.

D'après l'exemple de l'arbre du diagramme, prendre la racine de Merkle comme clé publique, quatre paires de clés OTS précaculées peuvent être utilisées pour générer quatre signatures cryptographiquement sécurisées à usage unique valides. La racine de Merkle de l'arbre binaire de hachage peut être transformée en adresse de registre (éventuellement par hachage itératif avec une somme de contrôle jointe). Une signature complète S d'un message M pour une paire de clés d'OTS donnée comprend : la signature s, le numéro de clé d'OTS n et le chemin d'accès d'authentification de Merkle; c'est-à-dire pour OTS, la paire de clé 0 (donc n = 0) :

S = s ; n ; clé publique d'OTS 0 ; H1 ; H2 ; H5 ; H6 ; racine

Compte tenu que la clé publique d'OTS et le hachage de feuilles peuvent être déduits de *s*, et que les nœuds parents peuvent être calculés de leurs enfants, on peut en fait résumer comme ceci :

S = s ; n ; H2 ; H6 ; racine

Où *S* est valide en vérifiant la clé publique d'OTS de *s* et *M*, puis en vérifiant que les hachages depuis le chemin d'accès d'authentification de Merkle recréent une racine de Merkle correspondant (clé publique).

6.2 État

L'utilisation du schéma de signature de Merkle (MSS) ci-dessus de manière sécurisée repose sur le fait de ne pas réutiliser les clés d'OTS. Par conséquent, il dépend de l'état des signatures ou que les opérations signées soient enregistrées. Généralement, dans l'usage du monde réel, ce serait

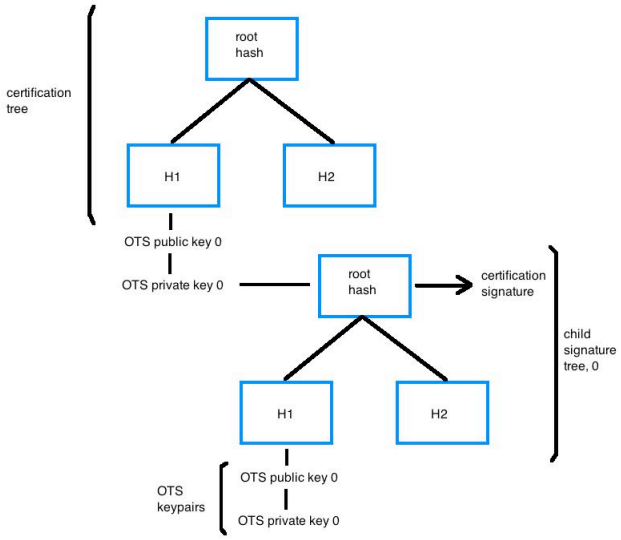
potentiellement un problème, mais un registre de blockchains public immuable est le support de stockage idéal pour un schéma de signature cryptographique dynamique. Un schéma plus récent de signature cryptographique basée sur le hachage appelé SPHINCS offrant signatures sans état pratiques avec une sécurité de 2^{128} a été signalé en 2015 [2].

6.3 Hyperarbres

Un des problèmes du MSS de base est que le nombre de signatures possibles est limité et que toutes les paires de clés OTS doivent être préalablement générées avant le calcul de l'arbre de Merkle. Le temps de génération de clés et de signature s'accroit exponentiellement avec la hauteur de l'arbre *h*, ce qui signifie que les arbres supérieurs à 256 paires de clés d'OTS deviennent temporellement et computationnellement chers à générer.

Une stratégie pour différer le calcul pendant la génération de la clé et d'arbre et étendre également le nombre de paires de clés d'OTS disponibles est d'utiliser un arbre qui est lui-même composé d'arbres de Merckle appelé "hyperarbre". L'idée générale est de signer la racine de Merkle d'un arbre enfant avec une clé d'OTS à partir du hachage de la feuille d'un arbre parent connu sous le nom d'un arbre de certification.

Figure 4. Liens entre les arbres Merkle



Dans la forme la plus simple (hauteur h = 2), un arbre de certification est précalculé avec 2^1 paires de clés d'OTS et lorsque la première signature est requise, un nouvel arbre de signature (arbre de signature 0) est calculé et signé par une des paires de clés d'OTS de l'arbre de certification. L'arbre de signature est composé de n hachages de feuilles avec les paires de clés d'OTS correspondantes et celles-ci servent à signer des messages au besoin. Lorsque chaque paire de clés d'OTS dans l'arbre de signature a été utilisée, l'arbre de signature suivant (arbre de signature 1) est signé par la deuxième paire de clés d'OTS de l'arbre de certification et le prochain lot de signatures est possible.

Une signature S d'une telle construction d'hyperarbre devient légèrement plus compliquée et comprendrait:

1. De l'arbre de signature: *s, n, chemin de Merkle, racine*
2. De chaque arbre de certification: *s* (de la racine de l'arbre enfant), *n, chemin merkle, racine*

Il est théoriquement possible de nicher des couches d'arbres vers le bas depuis l'arbre de certification pour étendre le MSS original à l'infini. La taille de la signature augmente linéairement pour chaque arbre supplémentaire qui est signé, tandis que la capacité de signature de l'hyperarbre augmente de façon exponentielle.

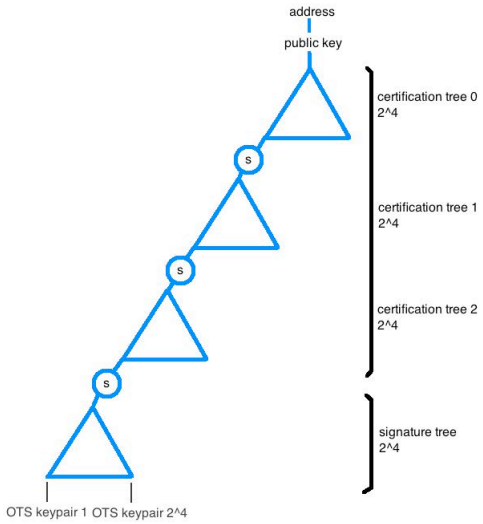
6.3.1 Exemples d'hyperarbres

Pour démontrer la facilité avec laquelle le MSS peut être étendu avec une construction d'hyperarbre, considérons un arbre de certification initial de hauteur h_1 = 5, avec 2^5 hachages de feuilles et les paires de clés d'OTS associées. La racine de Merkle de cet arbre est transformée pour générer une adresse du registre. Un autre arbre de Merkle, un arbre de signature de taille identique (h_2 = 5; 2^5 feuilles et paires de clés d'OTS) est instancié. 32 signatures sont possibles avant que l'arbre de signature suivant ne soit créé. Le nombre total de signatures possibles est de $2^{h_1+h_2}$ qui, dans ce cas, est 2^{10} = 1024.

Sur un MacBook pro de 2.7Ghz i5, 8gb ram créant des paires de clés d'OTS et un arbre de certification de Merkle pour différentes tailles ont donné les résultats suivants (code python non optimisé, Winternitz OTS): $2^4 = 0{,}5s, 2^5 = 1{,}2s, 2^6 = 3{,}5s, 2^9 = 15{,}5s$. Un hyperarbre consistant en la génération initiale de deux 2^4 arbres prend environ 1*s* par rapport à 15,5*s* pour l'arbre standard 2^8 MSS pour la même capacité de signature.

L'augmentation de la profondeur (ou hauteur) d'un hyperarbre continue cette tendance. Un hyperarbre composé de quatre arbres de certification 2^4 enchaînés et un arbre de signature de taille 2^4 est capable de 2^{20} = 1.048.576 signatures avec un coût supplémentaire pour la taille de signature, mais un temps de création de seulement 2: 5*s*.

Figure 5: Construction d'hyperarbre



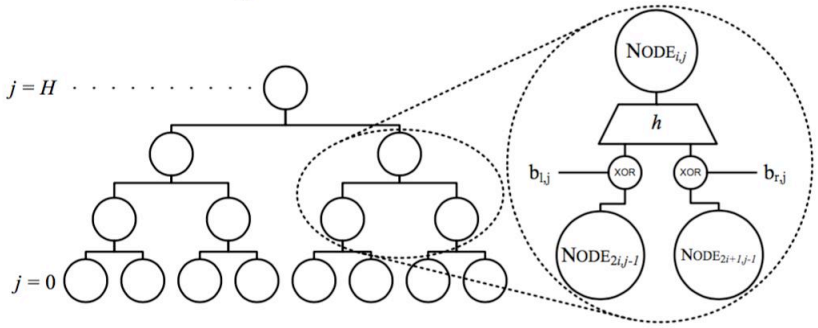
Il n'y a aucune exigence que l'hyperarbre soit symétrique et, donc, s'il est composé initialement de deux arbres, il pourrait ensuite être étendu plus tard en signant d'autres couches d'arbres. Les signatures d'une adresse de registre commenceraient donc petites et finiraient par augmenter à mesure que la profondeur

de l'hyperarbre augmente. L'utilisation d'un hyperarbre de Merkle pour créer et signer des transactions à partir d'une adresse de registre ne nécessiterait probablement jamais > 2^{12} transactions. Ainsi, la capacité de signer avec une facilité de calcul 2^{20} fois en toute sécurité à une profondeur d'hyperarbre de h = 5 est plus que suffisante.

6.4 XMSS

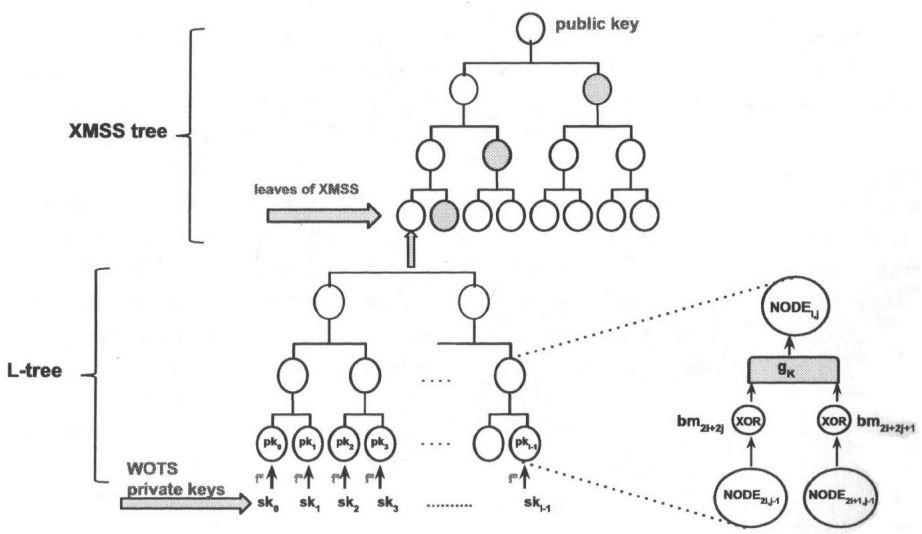
Le schéma de signature de Merkle étendu (XMSS) a été signalé pour la première fois par Buchmann et al. en 2011 et a été publié sous la forme d'un projet IETF l'année dernière [4] [7]. Il est sécurisé et existentiellement infalsifiable sous des attaques de message choisies avec des exigences de sécurité minimales: un PRF et une seconde fonction de hachage résistante à la pré-image. Le schéma permet l'extension de signatures à usage unique via un arbre de Merkle, où la différence majeure est l'utilisation de bitmask XOR des nœuds enfants avant la concaténation des hachages dans le nœud parent. L'utilisation du bitmask XOR permet de remplacer la famille de fonction de hachage résistante aux collisions.

Fig. 1. The XMSS tree construction



Les feuilles de l'arbre ne sont pas non plus des hachages de paires de clés d'OTS, mais la racine des arbres enfants L qui détiennent les clés publiques d'OTS avec des morceaux formant les feuilles de base. Winternitz OTS+ est utilisé pour les signatures uniques (même si la variante de 2011 a été décrite en premier).

Figure 7: Construction de XMSS [8]



La longueur en bits de la clé publique XMSS est $(2(H + \lceil \log l \rceil) + 1)n$, une signature XMSS a la longueur $(l + H)n$ et la longueur de la clé secrète de signature XMSS est < 2n.

Buchmann rapporte la performance avec un gigahertz d'Intel (R) *i*5 2,5 pour un arbre XMSS de hauteur, h = 20, où w = 16 et la fonction de hachage cryptographique utilisée est SHA-256 (m = 256) jusqu'à environ un million de signatures. Avec les mêmes paramètres et matériel, la signature a pris 7ms, la vérification 0,52ms de et la génération de clé 466 secondes. Le niveau de sécurité réalisé avec de tels paramètres était de 196 bits pour une taille de clé publique de 1,7 Ko, une clé privée de 280 bits et une signature de 2,8 Ko. XMSS est un modèle attrayant, son inconvénient principal étant le temps extrêmement long pour la génération des clés.

6.5 Performance de l'arbre XMSS

L'utilisation d'une bibliothèque Python non-optimisée construite pour une formation de nœuds de tests QRL d'un arbre XMSS de 4096 feuilles (h = 12) avec toutes les clés et les bitmasks générés à partir d'un PRF basé sur le hachage a pris 32s sur le même matériel hors décrit ci-dessus (un Macbook pro 2.7 Ghz i5, 8 Go de ram). Ce chiffre comprend la génération via PRF de plus de 8000 bitmasks et plus de 300 000 fragments de sk. Un algorithme de parcours en arbre de Merkle plus efficace et la nécessité d'effectuer uniquement des hachages w-1 par fonction de chaîne de clé secrète en WOTS + plutôt que 2^{w-1}avec WOTS contribue à un amélioration considérable de la performance sur un MSS classique.

Un format de signature complet d'environ 5,75 Ko a été atteint dans cette construction (codage de chaîne hexadécimale de 11,75 Ko), comprenant les paires de clés d'OTS, la signature, l'accès d'authentification XMSS, la clé publique OTS et l'arbre de clés publiques XMSS (y compris les graines de clés publiques PRF et la racine de l'arbre XMSS).

Pour les arbres de différentes capacités de signatures générées par PRF et une graine aléatoire, la performance suivante a été obtenue: (h = 9) 512 4,2s, (h = 10) 1024 8,2s, (h = 11) 2048 16,02s.

7 Schéma de signature proposé

7.1 Exigences en matière de sécurité

Dans la conception du QRL, il est important que la sécurité cryptographique du système de signature soit sécurisée contre l'attaque informatique classique et quantique, tant à l'heure actuelle qu'à l'avenir. XMSS utilisant SHA-256, où w = 16, offre une sécurité de 196 bits avec une sécurité prédite contre une attaque informatique en force jusqu'à l'année 2164 [9].

7.2 Signatures du QRL

Un schéma de signature d'hyperarbre asymétrique extensible et extensif composé d'arbres XMSS enchaînés est proposé. Cela a le double avantage d'utiliser un schéma de signature validé et de permettre la génération d'adresses de registres avec la capacité de signer des transactions en évitant un long délai de pré-calcul vu avec des constructions XMSS géantes. W-OTS+ est la signature ponctuelle par hachage choisie dans le schéma, tant pour des raisons de sécurité que de performance.

7.3 Construction d'hyperarbre

7.3.1 Tailles de clés et de signatures

Au fur et à mesure que le nombre d'arbres dans l'hyperarbre augmente, la taille des clés et des signatures augmente de façon linéaire, mais la capacité de signatures augmente de façon exponentielle.

Les tailles pour diverses clés et signatures publiques dérivées d'arbres XMSS (basées sur la description de 2011), où w = 16, m = 256, *h* est la hauteur de l'arbre et SHA-256 est choisi comme algorithme de hachage cryptographique:

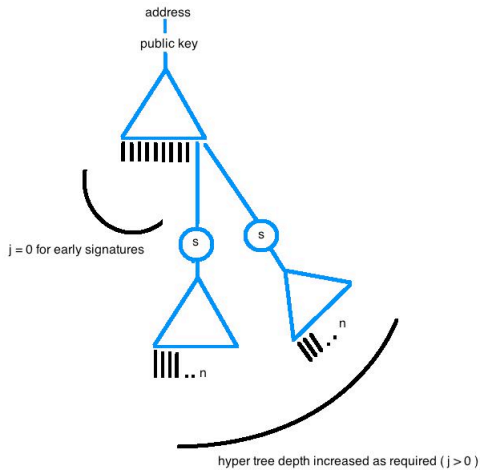
- k = 2, 2^2 signatures: clé publique 0,59ko, signature 2,12ko (0,4s)
- k = 5, 2^5 signatures: clé publique 0,78kb, signature 2,21ko (0,6s)
- k = 12, 2^{12} signatures: clé publique 1,23kb, signature 2,43ko (32s)
- k = 20, 2^{20} signatures: clé publique 1,7kb, signature 2,69ko (466s [3])

Le compromis pour la création d'un hyper-arbre XMSS (4 arbres, j = 3, k = 5) avec une capacité de signature finale de 2^{20} en moins de 3s par rapport à 466s, pour une signature de 8,84ko au lieu de 2,69ko peut être acceptable.

7.3.2 Asymétrie

La création d'un arbre asymétrique permet d'effectuer des signatures précoces avec une seule construction d'arbre XMSS, qui est étendue comme requis pour les signatures ultérieures à un coût pour la capacité de signature globale. La raison est que cela n'aura probablement aucune conséquence pour une application de registre de chaîne de blocs et que le portefeuille peut donner à l'utilisateur une option de capacité de signature par rapport à la signature / taille des clés. Une profondeur maximale d'arbre de j = 2 devrait suffire pour toutes les circonstances.

Figure 8: Hyperarbre asymétrique



7.4 Spécification de l'hyperabre de QRL

Les paramètres par défaut suivants doivent être adoptées pour une construction standard d'hyperarbre :

- $j = 0\ (j \in [0 \leq x \leq 2]), k = 12\ (k \in [1 \leq x \leq 14])$, limite supérieure de signatures possibles : 2^{36}, taille minimale de signature : 2,21 Ko, taille maximale de signature : 7,65 Ko.

par ex. : Un arbre unique XMSS, h = 12 avec 4096 signatures disponibles, qui peut être prolongé avec des arbres supplémentaires jusqu'à h = 14 à la demande. Pour la plupart des utilisateurs, des arbres supplémentaires ne seront probablement pas nécessaires.

7.4.1 Exemple de signature QRL

En supposant que la construction d'hyperarbre la plus complexe où j = 2 et h = 14, une signature de message de transaction m où n est la position de paire de clés OTS pour chaque arbre XMSS, il faudrait :

- Arbre de signature j = 2 : signature d'OTS de m , *n*, preuve d'authentification de Merkle, racine de Merkle de l'arbre de signature
- Arbre de certification j = 1 : signature d'OTS de racine de Merkle depuis l'arbre de signature (j = 2), n, preuve authentification de Merkle, racine de Merkle
- Arbre XMSS original j = 0 : signature d'OTS de racine de Merkle (j = 1), n, preuve d'authentification de Merkle, racine de Merkle

La vérification consiste à générer la clé publique d'OTS de m et de la signature, puis confirmer la preuve d'authentification de Merkle fournie génère la racine de Merkle de l'arbre de signature. Cela devient le message pour la prochaine signature OTS et, de là, la prochaine clé publique d'OTS est générée, la preuve d'authentification de Merkle fournie utilisée pour recréer la racine de Merkle de l'arbre de certification, qui devient le message pour la prochaine signature d'OTS de l'arbre de certification, et ainsi de suite. Une signature n'est valide que si la racine de Merkle de l'arbre le plus haut, l'arbre XMSS original (j = 0) est généré correctement.

Remarquons que les clés publiques d'OTS ne sont pas requises pour la vérification de l'arbre de signature XMSS. En fait, la racine de Merkle pour chaque arbre peut également être déduite et donc omise avec la vérification de signature d'hyperarbre si l'adresse du registre d'envoi est connue (comme il s'agit d'un dérivé calculé de la racine de Merkle pour l'arbre de certification XMSS le plus haut (j = 0) dans la signature QRL - voir Comptes plus loin).

Comme le schéma de signature est avec état, la mise en œuvre du portefeuille doit retenir et mettre à jour *n* pour chaque arbre XMSS généré dans l'hyperarbre pour une adresse donnée.

7.5 PRF

PRF depuis la graine HMAC_DRBG.

7.6 Portefeuille déterministe

En utilisant une seule graine (SEED), il est possible de générer un très grand arbre XMSS qui devrait suffire pour la plupart des utilisateurs pendant une période prolongée. Une source sûre d'entropie est utilisée pour générer cette graine qui est passée à travers une fonction sécurisée de PRF pour générer un ensemble de clés pseudo-aléatoires qui génèrent l'arbre. Un inconvénient de l'utilisation du même arbre XMSS est que l'utilisateur est connecté à une adresse unique (bien que l'exposition de clés publique ne soit pas un problème avec un MSS).

Une adresse bitcoin ou ethereum est dérivée de la clé publique associée ; ainsi une seule clé privée ou publique ne peut créer qu'une adresse unique. Toutefois, une adresse XMSS est dérivée de la clé publique, PK, qui contient la racine de Merkle et une graine publique. Si la graine reste constante, mais le nombre de paires de clés d'OTS pour calculer l'arbre varie, alors la racine de Merkle va changer pour chaque variation. Ainsi, pour chaque simple addition ou soustraction d'une paire de clés d'OTS unique, l'adresse dérivée va changer.

Cette fonctionnalité peut être utilisée par le logiciel de portefeuille/nœud pour générer de nombreuses variantes de l'arbre XMSS (prolonger/contracter selon les besoins à l'aide de la même graine initiale) permettant autant d'adresses uniques que nécessaires. Enregistrer ces informations de manière sécurisée, à état et compacte est mathématiquement trivial.

8 Paramètres de conception de cryptomonnaie

Le reste du livre blanc définira les paramètres de conception proposés pour le registre QRL. L'objectif du registre est d'être une blockchain publique hautement sécurisée contre les vecteurs d'attaque classiques et d'informatique quantique. Ce document n'est qu'une ébauche et est sujet à des révisions.

8.1 Preuve de participation (ou preuve d'enjeu)

QRL doit être un registre ouvert de blockchains publiques sécurisées par un algorithme de preuve-de-participation. Une époque dure 10.000 blocs. Des validateurs de participation sont déterminés à partir des transactions de participation à des époques antérieures. L'idée générale est que chaque validateur de participation signe une transaction contenant le hachage final d'une chaîne itérative d'une longueur de 10.000 hachages (un bitmask XOR peut être appliqué pendant chaque itération pour réduire les conditions de sécurité de la fonction de hachage). Une fois la transaction de participation confirmée dans la chaîne, chaque nœud du réseau peut alors attacher l'identité cryptographique de l'adresse de participation à la chaîne de hachage pour l'époque suivante.

8.1.1 Conception et caractère aléatoire

Pour chaque bloc, chaque nœud de validation qui participe à l'époque actuelle révèle le hachage précédent consécutif suivant dans la chaîne pour prouver cryptographiquement la participation et le vote pour être le sélecteur de bloc gagnant.

HMAC_DRBG est utilisé pour générer une séquence de nombres pseudo-aléatoires de sorties de 32 octets depuis la graine de données tirées de la blockchain (le bloc de genèse au début, puis l'entropie ajoutée tirée des hachages des titres de blocs récents concaténés pour chaque époque ultérieure).

Ainsi, chaque bloc que le validateur de participation a choisi pour devenir le sélecteur de bloc est déterminé en étant le hachage de révélation le plus proche en nombre de la sortie PRF pour ce bloc. Il est difficile à briguer car le PRF est inconnu des participants au moment de la création de la chaîne de hachage. En outre, une chaîne de hachage itérative (à clés) est essentiellement une séquence de nombres aléatoire. Enfin, même si les validateurs de participation s'associent en quelque sorte, ils ne peuvent pas connaître le contenu des autres chaînes de hachage des validateurs de participation, comme ils ne sont pas encore révélés.

Pour empêcher qu'un bloc ne retienne une attaque, si un bloc valide n'est pas produit après la soumission d'un hachage valide, alors tout le bloc de rémunération depuis cette adresse est perdu et il lui est donc interdit de participer pendant une période de punition.

Pour atténuer une attaque de bloc vide des nœuds de Sybil ou des adresses de validateur de participation à faible solde, on utilise un seuil de liste de validateurs flexibles. La rémunération du bloc est payée de façon pondérée selon le solde de l'adresse de participation. Avec le message de hachage révélateur, chaque nœud divulgue également un hachage de racine d'arbre de Merkle d'une liste triée de 'txhashes' dans leur zone de transaction, ainsi que le nombre de transactions en attente pour un bloc. Chaque nœud retranche un pourcentage du haut et du bas pour voir le nombre de transactions attendues dans le bloc suivant. Si le bloc est vide ou a moins que le nombre prévu de validateurs autorisés pour miser les contrats à la hausse (à l'exclusion des validateurs de pauvre à riche) chaque bloc. Si les nœuds du bloc sélecteur agissent honnêtement, l'inverse est alors vrai et le nombre de validateurs participants augmente. Les fonds ne peuvent pas être déplacés pendant l'époque d'enjeu - ceci empêche les tentatives de truquer la sélection de bloc par la création de Sybil de nombreuses adresses de validateurs.

8.2 Frais

Les transactions de plus grande taille par rapport aux autres registres nécessitent que des frais de transaction soient payés à chaque transaction. L'auteur est d'avis que les marchés de frais artificiels (voir le bitcoin) sont inutiles et sont à l'encontre de l'idéal d'un registre ouvert de blockchains ouvertes. Chaque transaction, si elle paie des frais minimums, devrait être aussi valable que n'importe quelle autre. Les frais minimum que les mineurs sont disposés à accepter devrait flotter et être fixés par le marché; c'est-à-dire que les nœuds/mineurs fixent entre eux les limites de frais de manière compétitive. Une valeur minimale absolue se fera au niveau du protocole. Ainsi, les mineurs ordonneront des transactions à partir de la mempool pour l'inclusion dans un bloc à leur discrétion.

8.3 Les blocs

8.3.1 Temps de latence

Bitcoin a un temps de latence d'environ 10 minutes, mais avec l'écart naturel, cela peut conduire parfois à d'assez longues périodes avant que le prochain bloc ne soit miné. Des concepts de registres plus récents comme Ethereum ont amélioré ceci et bénéficient d'un temps de bloc beaucoup plus court (15 secondes) sans aucune perte de sécurité ou centralisation de mineur des taux élevés de blocs orphelins/périmés. Ethereum utilise une version modifiée du protocole Greedy Heaviest Observed Subtree qui permet aux blocs périmés/orphelins d'être inclus dans la blockchain et être rémunérés [13, 5].

Comme le QRL projette d'employer un algorithme de preuve-de-participation dès le début, nous comptons employer sans risque un temps de latence d'environ 15 à 30 secondes.

8.3.2 Rémunérations de bloc

Chaque nouveau bloc créé inclura une première transaction « coinbase » contenant une adresse de minage dans laquelle une rémunération égale à la somme de la rémunération de coinbase et à la somme combinée de frais de transaction dans le bloc. La rémunération de bloc est pesée selon le solde de l'adresse de validateur sélectionnée comme sélecteur de bloc.

La rémunération du bloc est recalculée par le nœud de minage à chaque bloc et suit le schéma d'émission de monnaie.

8.3.3 Taille de bloc

Pour éviter toute controverse, une solution adaptative prête-à-l'emploi calquée sur la proposition de Bitpay d'augmenter la taille de bloc d'après un multiple *x* de la taille médiane *y* des derniers blocs *z* serait employée [12]. L'utilisation de la médiane empêche le jeu par des mineurs d'inclure des blocs vides ou surpeuplés pour modifier la taille de bloc moyenne. *x* et *z* seraient alors des règles de consensus que le réseau aurait du mal à respecter.

Ainsi, une taille de bloc maximale *b* pourrait être simplement calculée comme :

$$b = xy$$

8.4 Dénominations et unités de monnaie

Le QRL emploiera un symbole monétaire, le quantum (quanta au pluriel), comme unité monétaire de base. Chaque quantum est divisible en un plus petit élément comme suit :

- 1: Shor
- 10^3 :Nakamoto
- 10^6 : Buterin
- 10^{10} : Merkle
- 10^{13} : Lamport
- 10^{16} : Quantum

Ainsi, chaque transaction impliquant une fraction de quantum est en fait un nombre entier très grand d'unités de Shor. Les frais de transaction sont payés et calculés en unités de Shor.

8.5 Comptes

Les soldes de l'utilisateur sont tenus dans des comptes. Chaque compte est simplement une adresse réutilisable unique de registre dénotée par une chaîne commençant par Q .

Une adresse est créée en exécutant un SHA-256 sur la racine de Merkle du plus haut arbre de certification XMSS. Une somme de contrôle de quatre octets est ajoutée à cela (formée à partir des quatre premiers octets du double hachage SHA-256 de la racine de Merkle) et la lettre « Q » préfixée, soit en pseudocode pythonique :

Q + sha256 (racine_de Merkle) + sha256 (sha256 (racine de merkle))[: 4]

Une adresse de compte typique :

Qcea29b1402248d53469e352de662923986f3a94cf0f51522bedd08f b5e64948af479

Chaque compte a un solde dénommé en quanta divisibles en une unité simple de Shor.

Les adresses sont avec état avec chaque transaction en utilisant une nouvelle paire de clés d'OTS et le stockage QRL que chaque clé publique a utilisés (ceci pourrait être taillé puisqu'elle peut être régénérée à la demande à partir de la signature et du message de transaction mais serait intensif au point de vue opérationnel) pour chaque compte. Un compteur de transaction appelé "nonce" est incrémenté avec chaque transaction envoyée d'un compte. Cela permet aux portefeuilles qui ne stockent pas toute la blockchain de garder une trace de leur emplacement dans le schéma de signature d'hyperarbre de Merkle avec état.

8.6 Émission des pièces

8.6.1 Considérations historiques

Bitcoin a été la première cryptomonnaie décentralisée et initialement expérimentale sans valeur monétaire, il était donc approprié de distribuer la monnaie entièrement à partir du minage. Plus récemment, Zcash a choisi le même procédé avec un % de la rémunération de minage de monnaie dans la période préliminaire de l'émission passant au projet open source - avec l'énorme volatilité des prix qui en a résulté.
Par contre, d'autres registres tels qu'ethereum ont vendu un gros % de l'approvisionnement final en pièces dans le cadre d'une offre initiale de pièces (ICO). Cela présente l'avantage que ceux qui l'adoptent de manière précoce gagnent encore potentiellement en soutenant le projet, mais en outre le projet lui-même peut générer des fonds pour poursuivre le développement et les données d'amorçage et développer le projet depuis ses débuts. L'approche ICO permet également à un marché de se développer facilement tandis qu'un fond de caisse plus important est disponible à l'achat et à la vente pour les investisseurs à partir du bloc de genèse.
Auroracoin (2014) a adopté une approche différente en offrant à tout le monde en Islande une part égale de l'OIC, tandis que les développeurs se sont gardé 50 % de l'approvisionnement en monnaie.
D'autres cryptomonnaies ont soit simplement cloné le bitcoin de part en part, ou démarré une nouvelle chaîne avec des codebases différents.

8.6.2 Transfert de solde entre chaînes

Il est possible d'émettre le QRL d'après une capture de l'état de l'actuel registre de bitcoin inséré dans le bloc de genèse QRL. L'idée générale serait de permettre aux utilisateurs de créer une transaction à usage unique d' « importation » contenant un message et une signature uniques (c.-à-d. une adresse de portefeuille QRL aléatoirement produite signée avec une clé privée de bitcoin depuis une adresse avec un solde de bitcoins au moment de la capture). Cette fonction pourrait demeurer active jusqu'à une certaine hauteur de bloc, puis le reste de l'approvisionnement de monnaie serait miné en tant que normal. Le gonflement initial dans le bloc de genèse serait taillé à la même hauteur de bloc. Un inconvénient de

ceci est que, bien que juste, il pénalise les détenteurs de cryptomonnaies autres que le bitcoin et est techniquement potentiellement difficile à exécuter pour les nouveaux utilisateurs. Un problème technique serait qu'il est possible de récupérer une clé publique ECDH ECDSA depuis une simple signature et le message. Ceci exposerait de manière permanente les clés publiques aux adresses de bitcoin utilisées dans le processus et il serait important de déplacer les fonds après vers une nouvelle adresse de bitcoin générée aléatoirement pour atténuer ceci.

(? permission possible de la même fonction pour les détenteurs d'ethereum)

8.6.3 Émission proposée - projet

La délivrance initiale de QRL sera la suivante :

- ICO de 1 million de quanta (4,7 % de l'approvisionnement final en monnaie) avant le lancement.
- Une capture d'état de tous les soldes d'adresse de bitcoin au-dessus de 0,01 btc utilisé pour former le bloc initial de genèse de QRL. Toute personne souhaitant transférer directement des pièces à un rapport de 1:1 à partir du registre de bitcoin vers le registre de QRL pourra le faire ainsi jusqu'à une hauteur de bloc de 518400 (3 - 6 mois) par l'intermédiaire du portefeuille de nœud.
- 1 million de quanta supplémentaires seront détenus dans une adresse de bloc de genèse pour l'usage de la fondation
- L'offre restante sera minée (21.000.000 - (2.000.000 + btc soldes importés par hauteur de bloc 518400))

8.7 Calendrier d'émission de monnaie

L'une des caractéristiques principales du bitcoin est sa rareté et la limite supérieure fixe à l'émission de la monnaie sous-jacente. QRL suivra bitcoin à cet égard avec une limite supérieure fixée à l'émission de pièces de 21 x 10^6 quanta. Une décroissance exponentielle en douceur dans la rémunération de bloc est favorisée jusqu'au plafond dur d'approvisionnement de la pièce. Cela éliminera la volatilité associée au phénomène de réduction de moitié des bitcoins.

L'approvisionnement total en pièces, x = 21x10^6, moins les pièces créées lors du bloc de genèse y se réduira de façon exponentielle de Z_0 à la baisse de façon permanente. La courbe de désintégration est calculée pour distribuer les rémunérations de minage pour environ 200 ans (jusqu'à 2217AD, 420480000 blocs à 15s temps de latence) jusqu'à ce qu'un seul quanta soit laissé sans déminage (même si le minage pourrait continuer par la suite).

L'approvisionnement en monnaie restante au bloc t, Z_t, peut être calculé avec:

$$Z_t = Z_0 \cdot$$

Le coefficient λ, est calculé à partir de: $\lambda = \frac{\ln Z_0}{t}$. Où t, est le nombre total de blocs dans le programme d'émission jusqu'au quanta final. Jusqu'au bloc 518400, λ = 3,98590885111x10^{-08}. La rémunération du bloc b est calculée pour chaque bloc avec:

$$b = Z_{t-1} - Z_t$$

Entre le bloc de genèse et le bloc numéro 518400, les soldes en bitcoins peuvent être transférés sur le registre via des transactions d'importation. Au bloc 518401, le calendrier des émissions re-ciblera le verrouillage dans les nouveaux soldes importés, réduisant Z_t et ajustant le bloc-rémunération en

conséquence.

Références

[1] http://oxt.me/charts.

[2] D. Bernstein. Sphincs: practical stateless hash-based signatures. 2015.

[3] J Buchmann. On the security of the winternitz one-time signature scheme.

[4] J Buchmann. Xmss - a practical forward secure signature scheme based on minimal security assump-tions. 2011.

[5] V Buterin. Ethereum whitepaper. 2013.

[6] A. Hulsing. W-ots+ - shorter signatures for hash-based signature schemes. 2013.

[7] A. Hulsing. XMSS Extended hash-based signatures. 2015.

[8] A Karina. An e cient software implementation of the hash-based signature scheme mss and its variants. 2015.

[9] A. Lenstra. Selecting cryptographic key sizes. 2001.

[10] R. Merkle. A certi ed digital signature. CRYPTO, 435, 1989.

[11] S. Nakamoto. Bitcoin A peer-to-peer electronic cash system. 2008.

[12] S Pair. A simple, adaptive blocksize limit. 2016.

[13] Yonatan Sompolinsky. Accelerating bitcoin's transaction processing fast money grows on trees, not chains. 2014.

[14] A. Toshi. The birthday paradox. 2013.

Addendum

Mise à jour relative à l'émission de pièces :

Ce qui suit est une mise à jour fournie au moment de la traduction du livre blanc et n'était pas présente dans le document original. Cet addendum actualise les informations relatives aux chiffres d'émission de prévente qui ont changé par rapport au document original. Il doit être considéré comme remplaçant la section 8.6 : Emission des pièces et 8.7 : Calendrier d'émission des pièces. Il provient de cet article de blog par Peter

Waterland https://medium.com/the-quantum-resistant-ledger/seed-investment-strategy-pos-algorithm-updates-3b3854e83a4a

Emission des pièces

—la distribution initiale sera de 65 millions de pièces (quanta).

— la distribution finale dans 200 ans sera de 105 millions de pièces (courbe lissée de désintégration exponentielle vers un plafond dur).

— 20% de prévente (13 million de quanta) sera tenu par l'équipe, dont 65% seront investis par la fondation QRL.

— 100% du financement de la prévente seront détenus par la fondation QRL avec un programme d'investissement échelonné lié à des étapes de développement et des objectifs de recherche.

— les adresses à multisignatures de bitcoin et ethereum seront acceptées pour les dépôts de financement de la prévente (multisignature 2-de-3 m-de-n avec clés détenues séparément par 3 membres fondateurs de QRL).

— lancement mainnet proposé (si possible avec listing en bourse synchronisé) prévu pour le trimestre 4 de 2017.