Introduction à la *blockchain*

Par Ilarion PAVEL
Conseil général de l'Économie (CGE)

Cet article constitue une introduction technique à la *blockchain*, en partant d'un problème concret de paiement électronique dans un réseau décentralisé, à l'aide d'une cryptomonnaie de type *bitcoin*. La mise en place d'un tel système rencontre immédiatement des exigences en matière de protection de la vie privée, de sécurité du paiement, de structuration des transactions et de validation dans un réseau distribué. La résolution de ces exigences nous permet d'expliquer pas à pas la structure de la *blockchain*, en introduisant en préalable quelques outils importants, comme les fonctions de hachage, la cryptographie à clé asymétrique, ainsi que la signature électronique.

La naissance de la technologie date de la fin des années 2000, suite à la publication d'un article signé sous le pseudonyme de Satoshi Nakamoto, qui propose un système de paiement électronique, appelé *bitcoin,* sans faire appel à une autorité centrale financière.

Dans un système de paiement centralisé, le tiers de confiance est le rôle joué par les banques. Supposons qu'une personne, appelée Alice, paie 150 € à une autre personne, appelée Bob. C'est la banque qui débitera les 150 € du compte d'Alice pour créditer le compte de Bob et gardera la trace écrite de cette transaction dans ses registres. Dans un système de paiement décentralisé de type *bitcoin,* les transactions sont inscrites dans un registre, qui est ensuite partagé avec l'ensemble des nœuds d'un réseau point à point (*peer-to-peer*). Ce sont donc les nœuds de ce réseau qui font office de tiers de confiance, en lieu et place des banques (voir la Figure 1 ci-dessous).

Dans un tel schéma, plusieurs questions se posent naturellement :

1) comment assurer la protection de la vie privée ? Alice et Bob n'ont pas forcément envie de dévoiler leur identité à l'ensemble du réseau et les détails de leurs transactions financières ;
2) comment garantir l'authenticité et l'intégrité de la transaction ? Une personne mal intentionnée pourrait intercepter la transaction, la modifier en changeant le montant ou le destinataire ;
3) comment permettre à Bob de dépenser les 150 € reçus de la part d'Alice, et en même temps empêcher une personne mal intentionnée de le faire à sa place ?
4) comment grouper les transactions pour constituer le registre ?
5) qui, dans un réseau décentralisé, aura autorité pour valider les transactions et sur la base de quels critères ?

Avant de répondre à ces questions, il faut aborder quelques notions simples de cryptographie : fonctions de hachage, cryptographie à clé asymétrique et signature électronique.

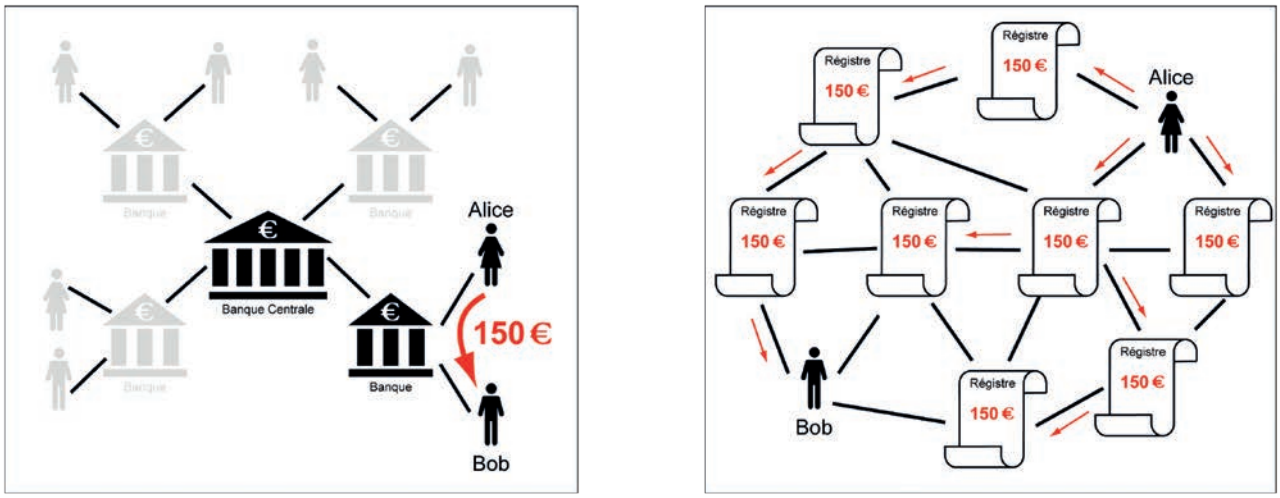

Figure 1 : Systèmes de paiement centralisé (à gauche) et décentralisé (à droite).

Rappel de quelques notions

Fonctions de hachage

Une fonction de hachage permet, à partir d'un fichier texte, d'extraire une empreinte numérique, appelée hash (voir la Figure 2 ci-après). Le hash d'un fichier est pratiquement unique : la probabilité d'avoir deux textes différents avec deux hash identiques est pratiquement nulle. Une fonction de hachage est très difficile à opérer en sens inverse : à partir d'un texte, on peut calculer facilement son hash, mais à partir d'un hash donné, il est pratiquement impossible de retrouver le texte. De plus, une petite variation du texte, par exemple le rajout d'un caractère, d'un espace ou d'une virgule, conduit à des changements drastiques du hash, toute modification devient ainsi facilement repérable.

Figure 2 : Fonction de hachage.

Les fonctions de hachage ont de multiples applications : par exemple, elles permettent de vérifier l'intégrité d'un fichier téléchargé [1] ou encore d'indexer plus facilement les fichiers sur une base de données. Elles sont aussi utilisées dans l'authentification login/mot de passe [2], ou encore dans la signature électronique ou la construction de la *blockchain,* comme nous le verrons par la suite.

Cryptographie à clé asymétrique

Imaginons en temps de guerre un réseau d'agents secrets opérant sur le territoire ennemi qui transmettent des informations vers leur pays. Bien évidemment, pour garantir la confidentialité, ces messages doivent être cryptés. Les crypter avec une clé symétrique [3] n'est pas une bonne idée [4]. En cas de capture d'un agent, l'ennemi pourrait intercepter non seulement le message mais également la clé, qu'il pourra utiliser par la suite pour décrypter tous les messages et compromettre ainsi l'ensemble du réseau.

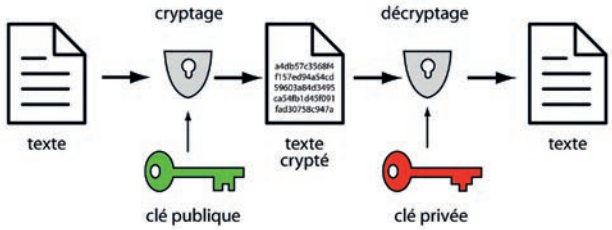

Figure 3 : Cryptographie à clé asymétrique.

L'idée est alors d'utiliser un système de cryptographie à clé asymétrique (voir la Figure 3 ci-dessus), qui fait intervenir deux clés différentes : l'une appelée clé publique, distribuée à l'ensemble des agents ; l'autre, appelée clé privée, gardée précieusement par le destinataire des messages. On crypte alors les messages avec la clé publique, puis on les décrypte avec la clé privée : on ne peut pas utiliser la même clé pour le cryptage et le décryptage. Ainsi, en cas de capture d'un agent, l'ennemi pourra intercepter la clé publique, mais ne pourra pas lire les messages cryptés faute de disposer de la clé privée [5].

Signature électronique

Une application utile des fonctions de hachage et de la cryptographie à clé asymétrique est la signature électronique, qui est utilisée pour authentifier des documents. À partir d'un document donné, on extrait d'abord son hash (son empreinte numérique) à l'aide d'une fonction de hachage. On crypte ensuite ce hash avec la clé privée, et l'on obtient ainsi la signature électronique. Le document sera alors envoyé aux destinataires accompagné de cette signature (voir la Figure 4 ci-dessous).

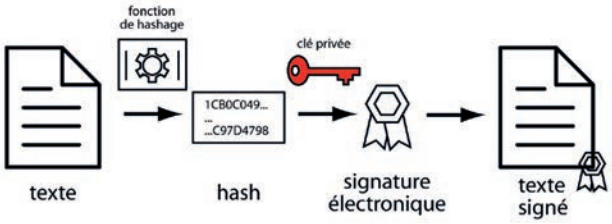

Figure 4 : Signature électronique d'un texte.

Pour vérifier l'authenticité du document, le destinataire effectue deux opérations (voir la Figure 5 de la page suivante). Il décrypte d'abord la signature électronique, à l'aide de la clé publique, il obtient donc le même hash que l'expéditeur avait extrait du document avant de l'avoir envoyé. Ensuite, le destinataire extrait le hash du document reçu, avec la même fonction de hachage que l'expéditeur a utilisée [6]. Si les deux hash coïncident, le document reçu est bien identique au document envoyé, sinon soit le document, soit la signature électronique ont subi des modifications pendant la transmission.

(1) On calcule le hash du fichier téléchargé, puis on le compare avec celui du fichier original, disponible sur le serveur. S'ils coïncident, le téléchargement a été correctement exécuté, le fichier est considéré comme intègre. Dans le cas contraire, le fichier est corrompu, il faudra alors le télécharger à nouveau.
(2) Pour se protéger contre le piratage, on ne stocke pas sur le serveur le mot de passe, mais son hash obtenu en appliquant de manière itérative plusieurs fois la fonction de hachage. Lorsqu'un utilisateur saisit son mot de passe pour se connecter, on recalcule ce hash, puis on le compare avec le hash stocké. S'ils coïncident, on autorise la connexion. Le nombre d'itérations est choisi pour que ce processus puisse durer quelques secondes, ce qui n'est pas gênant pour l'utilisateur, mais peut décourager un hacker qui essaie successivement diverses combinaisons de caractères dans l'espoir de tomber par hasard sur le mot de passe.
(3) Clé symétrique : on utilise pour décrypter la même clé que celle utilisée pour le cryptage.
(4) Cela ne signifie pas que la cryptographie à clé symétrique n'est pas utilisée. Elle nécessite des calculs bien plus simples que la cryptographie à clé asymétrique, qui la rend bien plus pratique pour le décryptage en temps réel, par exemple dans le cas des gros fichiers ou du *streaming*.

(5) En pratique, on peut utiliser un système à cryptographie asymétrique pour communiquer d'abord la clé symétrique, puis utiliser cette clé pour crypter les messages. On prendra soin d'employer une nouvelle clé symétrique pour chaque message : donc, même si l'ennemi l'intercepte, il ne pourra pas l'utiliser pour décrypter les autres messages.
(6) Il suffit juste de la préciser, les fonctions de hachage étant standardisées, elles sont donc connues par tout le monde.

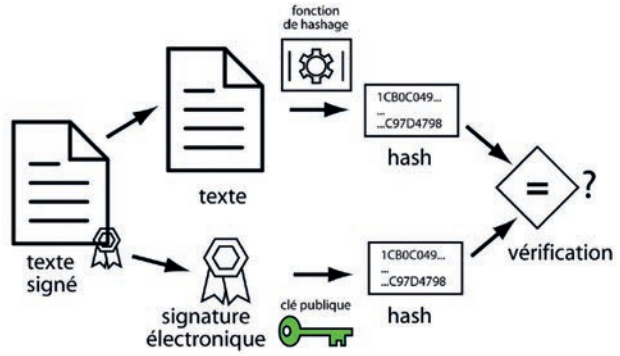

Figure 5 : Vérification de l'authenticité d'un texte signé électroniquement.

Les problèmes soulevés par la mise en place d'un système de paiement fondé sur la *blockchain* et leur résolution

On peut maintenant revenir aux questions posées en début d'article et essayer de les résoudre une par une.

Protection de la vie privée

Pour protéger leur vie privée, Alice et Bob vont faire appel à des adresses cryptées. Ce ne sont pas leurs noms qui figureront dans le registre des transactions, mais ces adresses.

Pour obtenir une adresse, l'utilisateur doit d'abord choisir une clé privée, qu'il doit garder précieusement. À partir de cette clé privée, créée à l'aide d'algorithmes de cryptage spécifiques [7], on génère une clé publique, qui sera mise à la disposition de l'ensemble des utilisateurs du réseau. Puis, en appliquant des fonctions de hachage [8], cette clé publique est transformée en adresse, que l'utilisateur utilisera par la suite dans ses opérations financières.

La chaîne des dérivations – clé privée => clé publique => adresse – est relativement facile à calculer directement ; en revanche, la remonter est pratiquement impossible.

D'un point de vue pratique, Alice dispose d'une certaine somme d'argent (« associée » à l'une de ses adresses), disons 0,1 *bitcoin* (BTC), qu'elle a reçu lors de précédentes transactions. Elle signe alors une transaction d'un montant de 0,015 BTC [9], en spécifiant l'adresse fournie

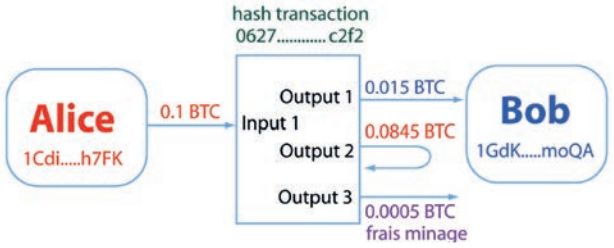

Figure 6 : Exemple d'une transaction financière simple, l'adresse d'Alice étant 1Cdi…h7FK, celle de Bob 1Gdk…moQA. Ces adresses sont codées sur 34 caractères. En pratique, on les saisit *via* des liens hypertext ou sous forme de codes-barres QR.

par Bob [10]. Alice décide également de payer des frais en valeur de 0,0005 BTC au mineur ayant validé cette transaction, comme nous l'expliquerons *infra*. Quant au reste, soit 0,0845 BTC, Alice se le paiera à elle-même, en faisant usage de l'une de ses adresses, qui peut être la même que celle associée à la somme initiale de 0,1 BTC [11] (voir la Figure 6 ci-dessus).

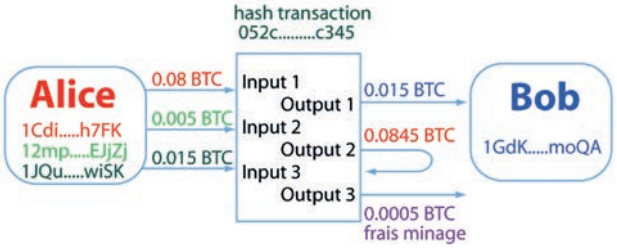

Figure 7 : Transaction multi-adresses. Pour payer Bob, Alice peut utiliser plusieurs adresses sur lesquelles elle dispose de diverses sommes d'argent.

Le protocole *bitcoin* offre en effet une grande flexibilité. Une transaction peut contenir plusieurs adresses d'entrée (Alice peut récupérer l'argent en provenance de plusieurs adresses, voir la Figure 7 ci-dessus) et plusieurs adresses de sortie (Alice peut payer plusieurs personnes par le biais de la même transaction).

Garantir l'authenticité et l'intégrité de la transaction

La transaction contient donc l'adresse d'Alice, l'adresse de Bob [12], le montant de la somme versée ainsi qu'un petit programme, appelé *script,* dont on verra par la suite l'usage qui en est fait. Utilisant sa clé privée, Alice signe cette transaction en apposant sa signature électronique, puis l'envoie sur le réseau.

(7) Le protocole *bitcoin* utilise le système de cryptage à base de courbes elliptiques définies sur un corps fini. À partir d'une clé privée, on peut rapidement générer la clé publique, mais à partir de la clé publique, il est pratiquement impossible de retrouver la clé privée. Ce dernier point tient des propriétés spéciales des opérations d'addition sur les courbes elliptiques.

(8) Dans le protocole *bitcoin*, on obtient l'adresse en appliquant successivement à la clé publique les fonctions de hachage SHA(256), puis RIPEMD(160). On obtient ainsi une adresse codée sur 160 bits. Pour des raisons pratiques, on rajoute à cette adresse un préfixe (qui spécifie que l'on va coder une adresse) et une clé de détection d'erreur, puis on code l'ensemble en utilisant le code Base58 (on utilise les minuscules et les majuscules de l'alphabet, les chiffres et deux symboles spéciaux (+ et /), mais l'on élimine les caractères ambigus qui pourraient engendrer des erreurs comme O et 0, I et l). On obtient ainsi le format habituel d'utilisation de ces adresses *bitcoin*.

(9) Pour des raisons de simplicité, on a choisi le cours 1 BTC = 10 000 €.

(10) Bien évidemment, Bob aura choisi au préalable une clé privée, il aura généré la clé publique, puis l'adresse, qu'il donnera ensuite à Alice pour effectuer ce paiement.

(11) Pour assurer l'anonymat, il est conseillé de ne jamais réutiliser une adresse ayant déjà servie. Cela implique qu'à chaque transaction, on doit utiliser une nouvelle adresse, il est donc également nécessaire de disposer d'une nouvelle clé privée. Le système risque de devenir très compliqué à gérer, mais heureusement, il existe des systèmes de portefeuilles spéciaux, qui facilitent énormément la gestion des clés privées.

(12) La situation est un peu plus complexe. En fait, ces adresses ne figurent pas en clair dans la transaction. Elles sont récupérées à partir de divers éléments : l'adresse d'Alice est contenue dans le hash de la transaction précédente (qui a crédité le compte d'Alice avec la somme d'argent nécessaire pour payer Bob), celle de Bob a été insérée par Alice dans le *script* de la transaction.

La transaction se propage alors de proche en proche dans les nœuds du réseau. À chaque nœud, grâce à la clé publique, on vérifie l'authenticité de la transaction. Une fois validée, celle-ci est envoyée vers les nœuds suivants. C'est ainsi que, de proche en proche, en quelques secondes seulement, la transaction se propagera à l'ensemble du réseau, son intégrité étant protégée par la signature électronique [13]. Supposons, par exemple, que, dans un nœud du réseau, une personne mal intentionnée modifie cette transaction, puis l'envoie aux nœuds suivants. Ces derniers, grâce à la signature électronique d'Alice, vont détecter cette modification, et vont donc qualifier cette transaction comme non valide et arrêter sa propagation au sein du réseau.

Autorisation des dépenses

Bob reçoit donc, sur l'une de ses adresses, la somme de 0,015 BTC de la part d'Alice. Supposons que cette transaction est inscrite dans le registre et validée par une opération de minage, que nous allons décrire ultérieurement. Comment permettre à Bob de dépenser cet argent et, en même temps, empêcher toute autre personne de le faire à sa place ?

En fait, la transaction signée par Alice contient un petit programme, appelé *script,* qui bloque par défaut l'utilisation ultérieure de cet argent. Pour le débloquer, Bob doit prouver qu'il est le possesseur de la clé privée qui génère l'adresse où Alice a effectué ce paiement. Comme il est le seul à avoir la clé privée, Bob est la seule personne qui puisse débloquer le *script* et, en conséquence, utiliser l'argent envoyé par Alice [14].

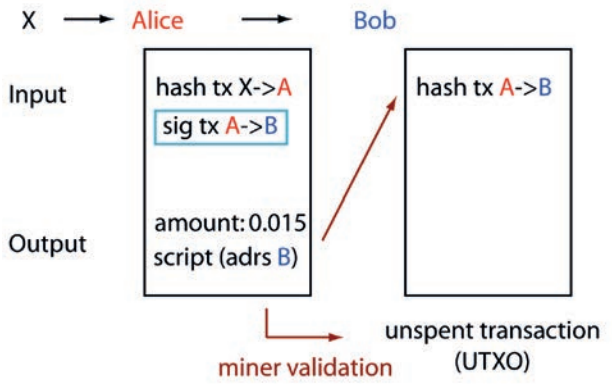

Figure 8.

Dans l'exemple de la Figure 8 ci-contre, Alice paie Bob par le biais de la transaction A => B, où, en sortie, elle indique le montant et insère le *script* qui précise l'adresse à laquelle cette somme doit être payée. On marque également, en entrée, le hash de la transaction précédente (X => A), qui a permis à Alice d'obtenir l'argent disponible, payé précédemment par la personne X (ce montant n'est pas indiqué, il sera vérifié par les nœuds du réseau, à partir du hash, tx X => A). C'est à partir de ces éléments qu'Alice calcule le hash de la transaction (tx A => B), puis, à l'aide de sa clé privée, appose sa signature électronique (sig tx A => B) et l'envoie dans le réseau. Cette transaction figure alors comme transaction non dépensée (UTXO), et ce tant que Bob ne l'utilise pas pour payer une autre personne.

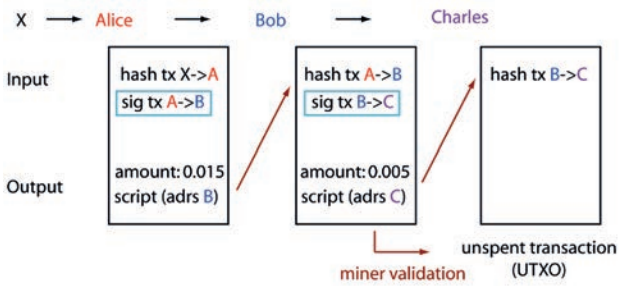

Figure 9.

Dans un second temps (voir la Figure 9 ci-dessus), Bob décide d'utiliser une partie de l'agent envoyé par Alice pour payer Charles. Comme Alice l'a fait auparavant, il indique le montant et le *script* qui précise l'adresse de Charles à laquelle cette somme doit être payée, ainsi que le hash de la transaction précédente (tx A => B), qui a été signée par Alice. Il signe alors cette transaction (sig tx B => C). Mais pour pouvoir payer Charles et donc dépenser l'argent reçu d'Alice, Bob doit montrer qu'il est le possesseur de la clé privée qui correspond à l'adresse marquée par Alice dans le *script* (adrs B).

En pratique, un nœud du réseau reçoit la transaction tx B => C, vérifie son authenticité *via* la signature électronique de Bob (sig tx B => C), puis cherche dans la *blockchain* les éléments de la transaction précédente (tx A => B), en particulier l'adresse adrs B et le montant payé par Alice. Il va hacher la clé publique de Bob, disponible sur le réseau, pour vérifier qu'elle correspond à l'adresse adrs B, puis, avec la même clé publique, il va décrypter la signature de Bob (sig txB => C) et vérifier que celle-ci correspond bien au hash de la transaction tx B => C. Pour valider complètement la transaction, il vérifiera une vingtaine d'autres conditions, comme s'assurer que le montant payé par Bob à Charles est bien inférieur à celui payé par Alice à Bob. Une fois validée, la transaction sera envoyée aux nœuds les plus proches, où auront lieu les mêmes vérifications.

En contrepartie, Bob devra garder précieusement la clé privée. La perdre signifierait l'impossibilité pour lui de débloquer le *script,* et donc d'effectuer des dépenses ultérieures avec l'argent envoyé à l'adresse générée à partir

(13) Outre l'authenticité et l'intégrité, la signature électronique garantit également la non-répudiation de la transaction : une fois signée, Alice ne pourra pas dénier le fait qu'elle est la signataire de cette transaction.
(14) En fait, Bob doit juxtaposer au *script* de blocage (mis par Alice dans la transaction vers Bob) un *script* de déblocage qui prouve qu'il est le possesseur de la clé privée qui a généré l'adresse de paiement. La vérification se fait en deux étapes : on vérifie que la clé publique de Bob engendre bien l'adresse de paiement qui figure dans la transaction signée par Alice, puis que la clé privée que Bob a générée est bien la clé publique. Cette procédure s'appelle "*Pay to the Public Key Hash*" (P2PKH). Mais dans le protocole *bitcoin*, il existe d'autres variantes, qui permettent en particulier des signatures multiples, utilisées dans les transactions signées par plusieurs personnes.

de cette clé. Cet argent sera donc irrémédiablement perdu. De même, si Bob se fait voler sa clé privée, le voleur pourra facilement usurper l'identité de Bob, et donc dépenser cet argent.

C'est pour ces raisons qu'il faut faire très attention dans l'utilisation des clés privées. Celles-ci sont gérées par des logiciels appelés portefeuilles [15]. Il est conseillé de ne pas les stocker directement sur l'ordinateur, mais plutôt d'utiliser des dispositifs dédiés connectés à l'ordinateur par des ports USB. La clé privée ne doit jamais quitter le portefeuille USB, la signature des transactions se fait directement au sein même de ce dispositif. Il est également conseillé, pour des montants importants, d'imprimer la clé privée sur plusieurs supports papier que l'on stockera dans des coffres-forts.

Grouper les transactions

Bien évidemment, la taille du registre augmente au fur et à mesure que l'on rajoute des transactions. Mais inscrire et valider ces transactions, une par une, serait trop laborieux. C'est pour cette raison que l'on préfère les grouper pour former un bloc [16], valider ensuite l'ensemble de ce bloc, puis le relier à la chaîne des blocs existants, déjà inscrits dans le registre. C'est ainsi que l'on forme la *blockchain.*

Pour relier ces transactions dans un bloc, on utilise des fonctions de hachage [17]. On extrait le hash de chaque transaction, on concatène ces hash deux par deux [18], puis on extrait le hash résultant de la concaténation. On répète successivement ces procédures de concaténation et de hash jusqu'à obtenir un seul hash, que l'on appelle la racine de Merkle. D'ailleurs, cette structure s'appelle plus exactement l'arbre de Merkle (voir la Figure 10 ci-contre) : à chaque étape, le nombre de hash intermédiaires diminuent d'un facteur deux [19].

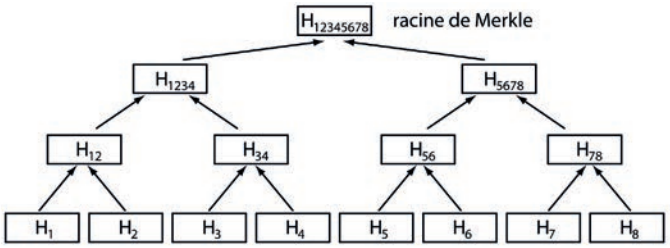

Figure 10 : L'arbre de Merkle.

Les transactions sont alors liées au sein du bloc par les hash successifs dans l'arbre de Merkle : la moindre modification de l'une d'entre elles modifie son hash, puis ce changement se propage sur cet arbre, jusqu'à la racine de Merkle. C'est ainsi que tout nœud du réseau peut vérifier l'intégrité des transactions regroupées au sein d'un bloc : si la racine de Merkle calculée par le nœud à partir de la transaction n'est pas identique à celle inscrite dans l'en-tête du bloc, c'est qu'il doit y avoir un problème, ce bloc est alors rejeté.

Valider les transactions, le résultat d'un consensus sur un réseau décentralisé

La validation d'un bloc se fait par l'opération de minage réalisée par certains nœuds du réseau, appelés mineurs, qui disposent des grandes puissances de calcul nécessaires à la résolution d'un problème mathématique relativement simple à énoncer, mais difficile à résoudre. On l'appelle habituellement « preuve de travail » (*proof of work*).

Un mineur choisit alors les transactions disponibles dans le réseau, il les regroupe dans un bloc et calcule la racine de Merkle. Il constitue ensuite l'en-tête du bloc, composé par cette racine de Merkle, le hash du dernier bloc miné de la *blockchain* (auquel il souhaite attacher son nouveau bloc), la date et l'heure (horodatage), ainsi qu'un paramètre appelé « difficulté » et un nombre variable, appelé « nonce ». C'est en calculant le hash de cet en-tête que le mineur essaie de résoudre le problème [20] : le hash calculé doit être inférieur à une certaine valeur cible déterminée par le paramètre difficulté [21].

Le mineur commence en choisissant un nonce, par exemple 0. Puis, il calcule le hash de l'en-tête, qu'il compare avec le seuil cible. Dans la plupart des cas, la cible n'est pas atteinte, le mineur va incrémenter le nonce d'une unité et il va refaire le calcul. Il continue ainsi à incrémenter le nonce et à refaire le calcul de hachage, jusqu'à ce que la valeur cible soit atteinte.

Le mineur ayant réussi le premier à atteindre cette valeur cible transmet le bloc à l'ensemble du réseau. Les autres mineurs vont hacher l'en-tête du bloc et vérifier que la va-

(15) En fait, le portefeuille ne contient pas d'argent, c'est plutôt un système qui gère les clés privées, les clés publiques et les adresses. En fait, dans la *blockchain*, l'argent ne circule pas d'une personne à une autre. Les transactions sont enchaînées les unes aux autres, et ce sont seulement les autorisations de dépenses qui sont transférées d'une personne à une autre par le jeu des clés et des adresses.
(16) Dans le protocole *bitcoin*, la taille d'une transaction varie en fonction du nombre d'adresses auxquelles elle fait référence, elle se situe en général entre 350 et 1 000 ko. Pour faciliter la propagation au sein du réseau, on a imposé des blocs qui ne dépassent pas 1 Mo, ce qui correspond à 1 000-3 000 transactions par bloc. Pour augmenter le nombre des transactions, certains protocoles, par exemple *Bitcoin Cash*, ont augmenté la taille du bloc à 8 Mo, puis à 32 Mo.
(17) Dans le protocole *bitcoin*, on applique deux fois la fonction de hachage SHA256, dont le résultat est un hash de 256 bits.
(18) Si, à un niveau donné de l'arbre de Merkle, on ne dispose pas d'un nombre pair de hash, on double le dernier, ce qui permet de les grouper ensemble deux par deux.
(19) On peut se poser la question : pourquoi ne pas concaténer tous les hash des transactions constituant un bloc, puis appliquer une seule fois la fonction de hachage ? La raison est que l'on préfère avoir une structure en arbre (de Merkle), qui permet aux utilisateurs de terminaux mobiles de vérifier facilement une transaction au sein d'un bloc. Actuellement, la *blockchain* du réseau *bitcoin* est de 220 Go, elle est stockée dans les nœuds fixes. En revanche, les terminaux mobiles ne disposent pas d'une capacité de stockage aussi grande, on stocke alors seulement l'arbre de Merkle, ce qui allège d'un facteur 1 000 la taille de stockage nécessaire.

(20) Dans le protocole *bitcoin,* on applique deux fois la fonction de hachage SHA256.
(21) Appliquer deux fois SHA256 à l'en-tête du bloc conduit à un hash distribué quasiment de manière uniforme sur l'ensemble de l'intervalle entre 0 et 2^{256-1}. La résolution du problème, et donc la validation du bloc, consiste à trouver un hash inférieur à un seuil cible, fixé par le paramètre difficulté. Plus le seuil cible est faible, plus le problème est difficile à résoudre.

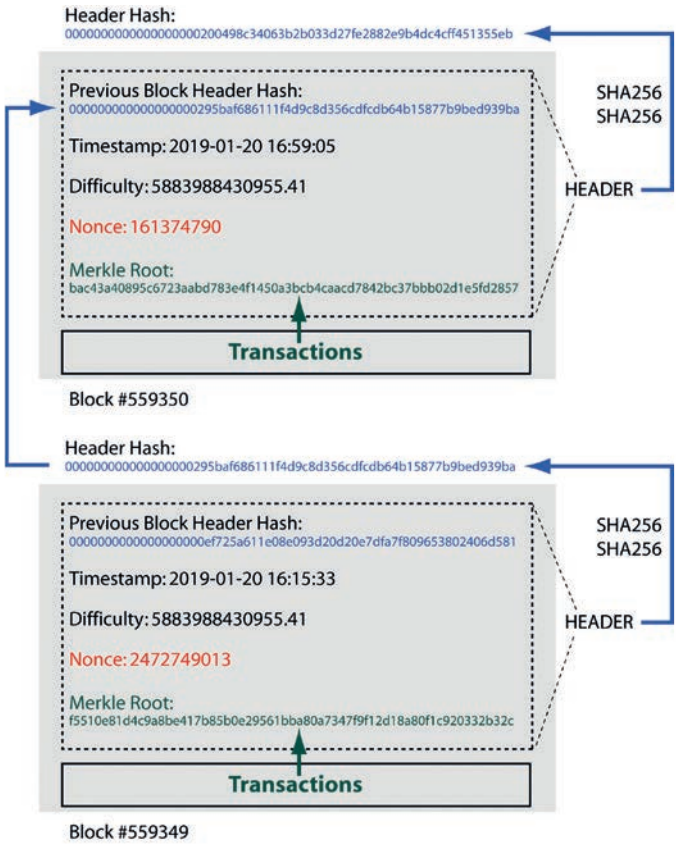

Figure 11 : L'enchaînement de blocs dans une *blockchain* : le hash de l'en-tête du bloc précédent se trouve dans l'en-tête du suivant, où l'on trouve également la racine de Merkle des transactions. Toute modification au sein d'une seule transaction modifie la racine de Merkle, puis le hash du bloc, brisant ainsi la *blockchain*.

leur cible est bien atteinte [22]. Le bloc est alors validé et lié à la *blockchain* (voir la Figure 11 ci-dessus).

Le paramètre difficulté est choisi de telle manière que le minage d'un bloc prenne en moyenne dix minutes. Tous les 2 016 blocs minés, ce qui correspond à deux semaines, on calcule la moyenne du temps nécessaire au minage d'un bloc. Si ce temps est inférieur à dix minutes, le problème est trop facile à résoudre, on augmente alors la difficulté. S'il est supérieur, le problème est trop difficile à résoudre, on diminue donc la difficulté. C'est ainsi que l'on adapte la vitesse de minage des blocs à la puissance de calcul du réseau [23].

Un mineur ayant réussi le minage d'un bloc est récompensé par de l'argent, actuellement 12,5 BTC, qui constitue une première source de revenu. Par construction du protocole, au démarrage du réseau *bitcoin,* ce montant était de 50 BTC, puis il a été prévu de le diminuer de moitié tous les 210 000 blocs minés, soit tous les quatre ans. Après 32 étapes de divisions successives par 2, soit vers 2140, le montant payé pour le minage d'un bloc sera inférieur à la plus petite unité monétaire, qui est le *satoshi,* lequel vaut 10^{-8} BTC. Un mineur ne recevra plus d'argent pour le minage d'un bloc, néanmoins il sera rémunéré par les frais de transaction.

Comme on l'a vu précédemment, on peut toujours rajouter à une transaction des frais destinés à rémunérer le mineur ayant miné le bloc qui contient la transaction en cause. Plus les frais d'une transaction sont importants, plus elle attire l'attention des mineurs, qui ont tout intérêt à l'inclure dans leur prochain bloc à miner [24].

La récompense du minage d'un bloc est également le moyen d'introduire de la masse monétaire dans le réseau *bitcoin*. Un calcul simple montre qu'en 2140, la masse monétaire atteindra 21 millions BTC [25].

Il peut arriver que deux mineurs, A et B, minent pratiquement en même temps leur bloc. Ils vont alors l'envoyer à l'ensemble des mineurs pour vérification. En raison de temps de propagation différents, on peut arriver à la situation où une partie du réseau reçoit pour validation le bloc du mineur A, alors que l'autre partie reçoit le bloc du mineur B. La *blockchain* bifurque, cette situation est appelée « fourche ».

La situation se résout simplement : chaque mineur ayant validé un bloc continue à miner le bloc suivant malgré la fourche : ceux qui ont validé le bloc A essaient de prolonger la *blockchain* sur la branche du bloc A, ceux qui ont validé le bloc B, en font de même pour la *blockchain* contenant le bloc B. À un certain moment, un mineur arrive à miner un bloc, par exemple, dans la branche A, alors que ceux de la branche B sont toujours en train d'effectuer leurs calculs de minage. Il va alors envoyer ce nouveau bloc, appelons-le C, à l'ensemble du réseau. Après vérification, les mineurs de la branche B qui étaient en train de miner vont alors abandonner cette branche, ils vont concentrer dès lors leurs efforts pour miner le bloc qui succédera au bloc C situé sur la branche A. La branche B

(22) Calculer le hash de l'en-tête du bloc quand on donne le nonce est facile, en revanche déterminer directement le nonce à partir de la valeur cible est impossible : on ne peut pas inverser les fonctions de hachage. Donc, la vérification du « bon » nonce peut être effectuée rapidement par les nœuds du réseau. En revanche, pour trouver le nonce qui satisfait la valeur cible du hash, les mineurs n'ont d'autre choix que de tester successivement divers nonces, en pratique en partant de 0 et en effectuant des incréments d'une unité. C'est là toute la difficulté de la preuve de travail !

(23) Un des points faibles de l'approche « preuve de travail » est que cela peut conduire à mobiliser de grandes puissances de calcul sur le réseau, accompagnées d'une forte augmentation de la consommation d'électricité. Lors des deux dernières années, la consommation d'électricité du réseau *bitcoin* a augmenté dix fois, pour atteindre 4 GW, soit la puissance de quatre centrales nucléaires !

(24) Les mineurs utilisent des métriques pour déterminer l'importance d'une transaction. Celle-ci augmente avec le montant de la transaction et diminue en fonction de la taille de la transaction (en octets). Cependant, le protocole peut imposer l'obligation pour les mineurs d'inclure des transactions anciennes, qui souvent n'ont pas prévu la rétribution des frais de transaction pour les mineurs. Ainsi, après un laps de temps suffisant, toutes les transactions pourront être incluses dans la *blockchain*.

(25) On récompense en attribuant 50 BTC pour les 210 000 blocs minés en premier, puis 25 BTC pour les 210 000 suivants, et ainsi de suite, cela revient à multiplier 210 000 blocs par les sommes 50+25+12,5+…, dans la limite d'un montant cumulé de 100 BTC. Le fait que cette masse monétaire soit bornée a engendré de grands débats parmi les économistes qui se sont interrogés sur la viabilité d'un tel système, qui serait vulnérable à une spirale déflationniste.

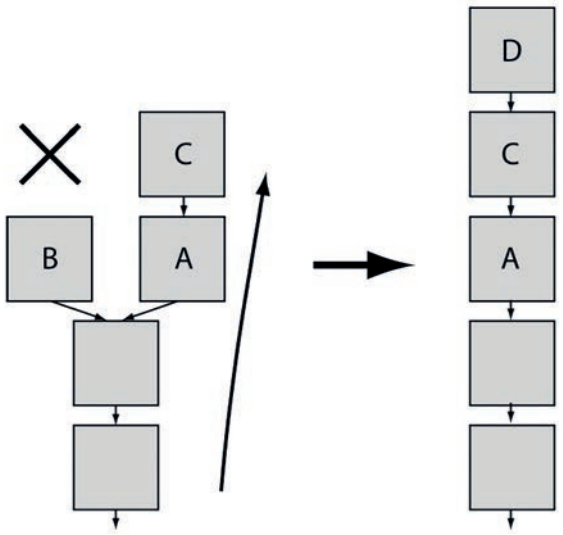

Figure 12 : Phénomène de fourche : cas où deux mineurs A et B arrivent à miner pratiquement en même temps leurs blocs, notés A et B. Ces blocs sont par essence différents, puisque chaque mineur choisit librement les transactions qu'il souhaite inclure dans son bloc. La communauté de mineurs se trouve alors divisée, chaque partie essaie de prolonger sa propre branche en continuant à miner des blocs. C'est la communauté ayant réussi à construire la branche la plus longue qui l'emporte.

est donc abandonnée, la *blockchain* se reforme en une branche unique contenant les blocs A et C [26] (voir la Figure 12 ci-dessus).

(26) Il peut arriver que le bloc sur la branche A et celui sur la branche B soient pratiquement minés en même temps. La fourche se maintient alors, les deux communautés de mineurs continuent à miner chacune sur sa branche. La situation peut continuer jusqu'à ce qu'une des branches devienne plus longue que l'autre, et c'est seulement à ce moment-là que la branche plus courte est abandonnée et que la *blockchain* se reforme sur la branche la plus longue. Ce type d'événement n'arrive cependant que très rarement.

Conclusion

Nous avons répondu à quelques-unes des questions posées par la mise en place d'un système de paiement décentralisé fondé sur la *blockchain*. Elles nous ont permis d'introduire de manière naturelle plusieurs concepts de base utilisés dans cette technologie : fonctions de hachage, cryptographie à clé asymétrique, signature électronique, arbre de Merkle et établissement du consensus sur un réseau décentralisé.

Bien que nous n'ayons abordé que la technologie *bitcoin,* ces concepts sont largement utilisés dans les autres réseaux de crypto-monnaies ou d'*alt-chains.*

Enfin, il faut préciser que le fonctionnement de la *blockchain* reste assez complexe à appréhender, mais il reste transparent pour l'utilisateur, qui n'a que quelques simples manipulations à faire, comme celles permettant des paiements classiques sur les portails Web ou *via* les terminaux mobiles.

Bibliographie

NAKAMOTO S., "Bitcoin: A Peer-to-Peer Electronic Cash System", https://bitcoin.org/bitcoin.pdf

ANTONOPOULOS A. (2017), *Mastering Bitcoin,* O'Reilly Media.

LEWIS A. (2018), *The Basics of Bitcoins and Blockchains,* Mango Publishing Group.

NARAYANAN A. *et al.* (2016), *Bitcoin and Cryptocurrency Technologies,* Princeton University Press.

SINGHAL B. *et al.* (2018), *Beginning Blockchain,* Apress.

DRESCHER D. (2017), *Blockchain Basics*, Apress.

CASTIGLIONE MALDONADO F. (2018), *Introduction to Blockchain and Ethereum*, Packt Publishing.

CAETANO R. (2015), *Learning Bitcoin,* Packt Publishing.