Sécurité et insécurité de la *blockchain* et des *smart contracts*

Par Jean-Pierre FLORI
Expert en cryptographie à l'Agence nationale de la sécurité des systèmes d'information (ANSSI)

La sécurité des nombreuses applications utilisant les *blockchains* repose non seulement sur la façon dont sont construites les *blockchains* sous-jacentes, mais aussi sur les spécificités de l'application construite à l'aide des données stockées dans ces *blockchains*, ainsi que sur le comportement des entités participant à l'expansion de la *blockchain*. Dans cet article, nous nous intéresserons à ces différents niveaux de sécurité et nous les illustrerons en nous appuyant sur deux applications phares des *blockchains* : la crypto-monnaie bitcoin et l'ordinateur-monde d'Ethereum.

Introduction

Les *blockchains* sont un élément central de la monnaie cryptographique bitcoin, de l'ordinateur-monde Ethereum, ainsi que de nombreux projets récents. La sécurité de telles applications repose non seulement sur la façon dont sont construites les *blockchains* sous-jacentes, mais aussi sur les spécificités des applications construites à l'aide des données stockées dans ces *blockchains*.

La sécurité des *blockchains*

Pour aborder la sécurité apportée par les *blockchains*, il est tentant de réduire la *blockchain* à sa plus simple expression. On obtient alors un concept d'une simplicité, et même d'une pauvreté technique, extrême : c'est une liste chaînée où chaque nouveau bloc pointe vers le précédent bloc. Si une telle construction est constamment utilisée en informatique afin de stocker des données, elle n'apporte aucune garantie de sécurité sur les données stockées dans un environnement malveillant et il est encore plus malaisé de partager et de mettre à jour une telle structure de données de façon distribuée.

La rigidité des *blockchains*

Si aucun mécanisme de sécurité n'est mis en place, comment s'assurer qu'un bloc de la liste n'a pas été remplacé par un autre bloc ? Il est donc tout d'abord nécessaire de « rigidifier » la liste chaînée.

Cette rigidité est généralement obtenue en incluant dans le $n^{ième}$ bloc de données une empreinte numérique du $i\text{-}1^{ième}$ bloc de données de taille fixée (par exemple, 256 bits) calculée en appliquant à ce dernier une fonction H de hachage cryptographique publique et non paramétrée par un secret, telles que SHA-2 ou SHA-3 (les choix possibles sont encore plus nombreux). La propriété attendue de la fonction de hachage cryptographique est la résistance au calcul de seconde préimage : pour remplacer un bloc N au sein de la liste, il faudrait en effet être capable de trouver un autre bloc ayant le même haché que celui stocké dans le bloc N+1. Les fonctions de hachage sont bien heureusement spécialement conçues pour qu'il soit illusoire d'espérer pouvoir réaliser une telle opération.

Une telle structure peut alors être utilisée comme registre. Les blocs ajoutés deviennent immuables : en remontant depuis le dernier bloc ajouté à la chaîne, il est possible de s'assurer qu'aucun des blocs précédents n'a fait l'objet d'une substitution.

La mise à jour des *blockchains*

Le défi technologique majeur des *blockchains* réside dans la distribution d'un tel registre. Il faut s'assurer que l'ensemble des acteurs s'accordent sur une version commune de la *blockchain*, mais aussi empêcher des entités malicieuses de prendre le contrôle de l'ajout de blocs à la *blockchain*. Ce problème, qui a déjà été beaucoup étudié par les informaticiens, dans le cadre des modèles dits « à permission » (c'est-à-dire de modèles où l'ensemble des acteurs sont connus et contrôlés) est connu sous le nom du « problème des généraux byzantins ».

S'il existe de nombreux protocoles permettant de parvenir à un consensus dans un cadre maîtrisé où tous les acteurs sont préalablement connus, ceux-ci ne fonctionnent plus quand des acteurs non authentifiés peuvent prendre part au protocole, ou s'en désister à leur gré. La principale innovation de bitcoin est d'avoir proposé un protocole permettant de parvenir à un consensus dans un tel environnement « sans permission » en utilisant des puzzles cryptographiques appelés « preuves de travail ».

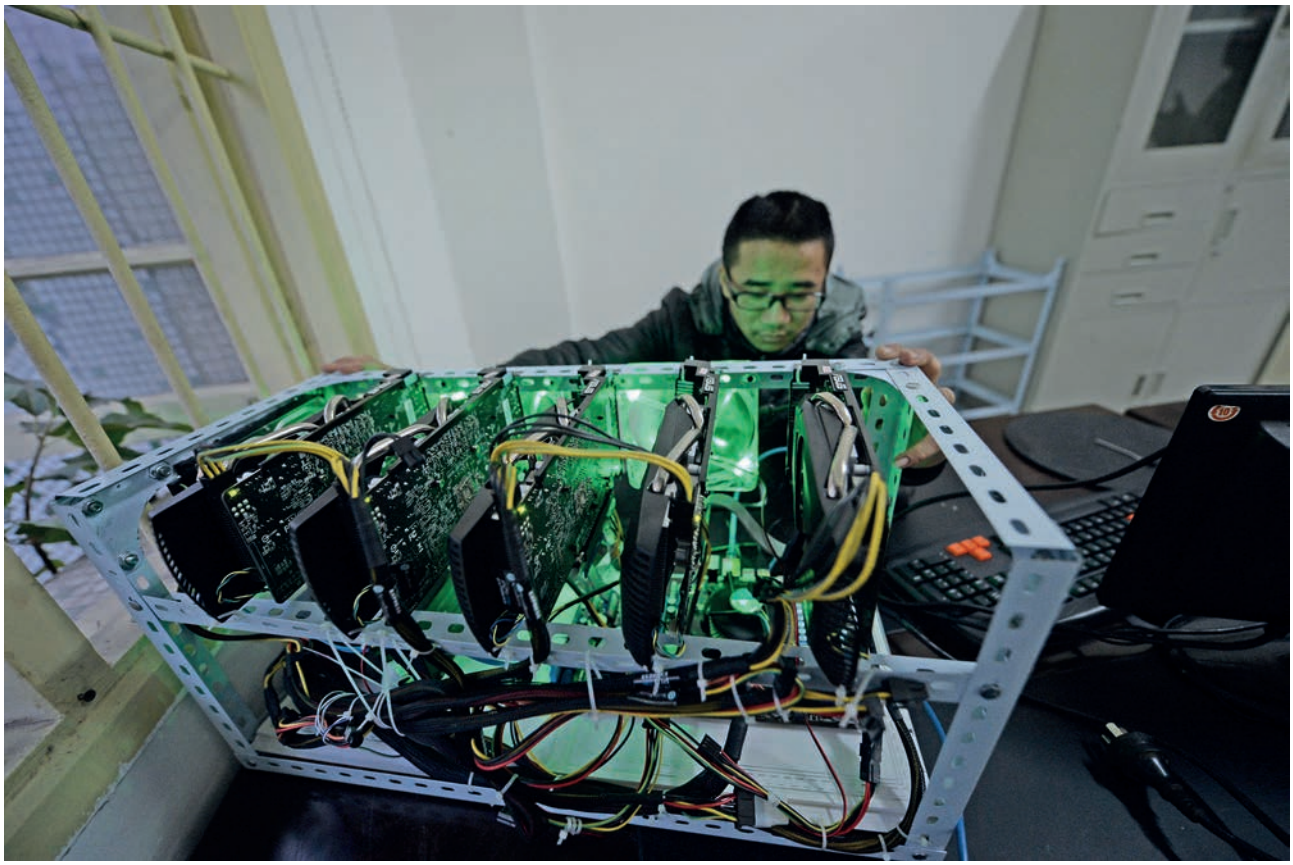

Photo © Ran Wen/Featurechina-ROPI-REA

Un employé de Landminers devant un ordinateur dédié au minage de bitcoins, une activité dans laquelle cette société est spécialisée, Chongqing (Chine), 6 décembre 2013.

« Le fait que la difficulté de la résolution du puzzle cryptographique soit suffisamment grande est un paramètre important de sécurité. »

L'utilisation de « preuves de travail »

Le principe de la « preuve de travail » est le suivant : afin de pouvoir proposer un nouveau bloc valide à ajouter à la *blockchain*, il faut résoudre un problème difficile.

Dans le cas du bitcoin, ces preuves de travail font à nouveau intervenir des fonctions de hachage cryptographiques ! Étant donné le haché H(Bi-1) du bloc précédent de la *blockchain,* ainsi que les données à intégrer au nouveau bloc Bi (une liste Li de transactions à valider dans le cadre de bitcoin), il s'agit essentiellement de trouver un préfixe Ni, tel que le nouveau bloc Bi=(Ni, H(Bi-1), Li) constitué de la concaténation de ces éléments satisfasse la condition que le haché H(Bi) commence par i bits nuls. Ce problème (encore plus difficile à résoudre que le problème évoqué dans le paragraphe précédent) se rapproche du problème de calcul de première préimage d'une fonction de hachage. Pour toutes les fonctions de hachage cryptographiques modernes, il n'existe à ce jour pas de meilleure façon de résoudre ce problème que de tirer des valeurs de Ni au hasard et de calculer le haché correspondant (plus exactement, il est possible d'omettre quelques tours de la fonction de hachage). Il suffit alors de jouer sur la valeur de i pour faire en sorte qu'un nouveau bloc soit généré toutes les dix minutes environ (i vaut aujourd'hui environ 70, dans le cas du bitcoin).

En soi, l'utilisation d'une « preuve de travail » ne permet pas de s'accorder lorsque deux blocs valides sont diffusés à travers le réseau, au même moment. Dans cette éventualité, et pour converger vers un consensus, bitcoin préconise de commencer par « ne rien faire » : chaque nœud choisit aléatoirement un des deux blocs valides pour prolonger sa copie locale de la *blockchain* et travaille dessus (on parle alors de « branche », ou « *fork* »). Avec le temps, il est fortement probable qu'un de ces deux choix conduira à une *blockchain* strictement plus longue que l'autre et le consensus sera alors que tous les nœuds se rabattent sur la *blockchain* la plus longue.

Le fait que la difficulté de la résolution du puzzle cryptographique soit suffisamment grande est un paramètre important de sécurité : tant qu'une entité malicieuse ne contrôle pas plus de la moitié de la puissance de calcul en obtenant ainsi une probabilité strictement supérieure à ½ de générer un bloc avant les autres nœuds, il paraît plausible (mais non prouvé de façon formelle !) que cette stratégie permette de se prémunir d'une entité essayant de réécrire l'histoire en créant sa propre « branche » à partir du nœud de son choix et en y ajoutant de nouveaux blocs jusqu'au moment où ladite branche deviendra plus longue que la « branche » officielle. En particulier, cette stratégie permet de se prémunir des doubles dépenses (un utilisa-

teur transfère une première fois ses fonds, attend que suffisamment de blocs soient ajoutés à la *blockchain* afin que le bénéficiaire considère la transaction comme confirmée, puis réécrit l'histoire pour remplacer le transfert initial par un autre), mais aussi de prévenir la censure (un attaquant pourrait produire une « branche » plus longue dès qu'une transaction qu'il ne voudrait pas voir apparaîtrait).

La sécurité formelle

Un axe de recherche actuel est de formaliser un tel protocole tout en prouvant de façon formelle sa sécurité. Parmi les notions de sécurité formalisées, notons :

- la cohérence spatiale : différents acteurs honnêtes doivent partager la même vision de la *blockchain* officielle (ou au moins un préfixe commun, afin de prendre en compte une légère désynchronisation ou de brefs « *forks* ») ;
- la cohérence temporelle : au cours du temps, la *blockchain* d'un acteur honnête doit conserver un préfixe fixe (c'est-à-dire que sa vision de la *blockchain* ne peut changer que marginalement, à la fin de la chaîne) ;
- la croissance : la longueur de la « chaîne » officielle croît tant qu'un nombre suffisant d'acteurs y contribue ;
- la qualité : une proportion satisfaisante des blocs générés par les acteurs honnêtes finit toujours par être intégrée à la chaîne.

Notons que tous ces travaux posent encore des restrictions plus ou moins fortes sur la modélisation du réseau reliant les acteurs et sur la dynamique de la population des acteurs. Cependant, ils proposent d'autres améliorations par rapport au protocole original du bitcoin afin d'accélérer la vitesse moyenne de validation des blocs en utilisant des chaînes secondaires et en mélangeant protocoles de consensus « à permission » et « sans permission ».

L'attaque des 51 %

Un certain nombre de crypto-monnaies alternatives dont la puissance de minage est moindre que celle du bitcoin ou d'Ethereum ont été attaquées par des groupes qui ont pu réécrire la *blockchain* et dépenser plusieurs fois leurs avoirs en regroupant des puissances de calcul supérieures à 51 % de la puissance de minage. C'est, par exemple, le cas de Krypton et de Shift, qui sont des clones d'Ethereum.

La sécurité du bitcoin

Bitcoin propose de construire une crypto-monnaie au-dessus de la *blockchain* en stockant dans chaque bloc un certain nombre de transactions monétaires. Chaque acteur possède un couple de clés (l'une étant publique et l'autre privée) de signature cryptographique et chaque transaction est représentée en utilisant un langage de script très simple reposant essentiellement sur l'instruction « transférer les bitcoins de la transaction y vers l'adresse z », accompagnée d'une signature « s », où :

- l'adresse z est (le haché d')une clé publique de signature ;
- le numéro de transaction pointe vers une transaction antérieure dans laquelle des bitcoins ont été transférés à la clé publique correspondant à la clé privée ayant produit la signature « s ».

La sécurité des transactions

Afin de dépenser les bitcoins transférés à l'adresse z, il faut être capable de produire une signature cryptographique à l'aide de la clé privée associée, ce que seul le possesseur de ladite clé pourra faire si le mécanisme cryptographique de signature sous-jacent est sûr et, bien évidemment, si cette clé privée n'a pas été divulguée ou dérobée.

Il existe de nombreux schémas de signature cryptographique adéquats. La plupart reposent sur des problèmes mathématiques tels que la factorisation de grands nombres entiers ou la résolution du problème du logarithme discret dans des groupes. C'est sur ce deuxième problème que s'appuie le mécanisme de signature cryptographique utilisé par bitcoin. Plus précisément, bitcoin s'appuie sur la difficulté de la résolution du logarithme discret sur une courbe elliptique à travers le mécanisme de signature ECDSA ***(****Elliptic Curve Digital Signature Algorithm)****.***

Le regroupement des « mineurs »

Un effet pervers de l'utilisation de « preuves de travail » (en dehors du côté hautement énergivore du minage) est que la majeure partie de la puissance de minage se concentre dans des zones proches de celles où sont produits les *Application-specific Integrated Circuits* (ASIC) et où l'électricité est bon marché.

De plus, les acteurs particuliers s'adonnant encore au minage n'ayant pas la puissance de calcul nécessaire pour espérer rentabiliser leur investissement énergétique se regroupent habituellement au sein de « piscines » de minage. Un serveur central centralise les calculs effectués par l'ensemble des mineurs de la piscine et leur redistribue les gains de manière proportionnelle. Le responsable du serveur central a alors un contrôle complet sur la puissance de calcul des mineurs de sa piscine vis-à-vis de l'extérieur.

Il n'est donc pas exclu qu'un acteur unique puisse finir par posséder plus de la moitié de la puissance de minage, remettant en cause le modèle sur lequel repose la non-malléabilité de la *blockchain*.

Des principes différents ont été proposés afin de pallier ces problèmes, tels les preuves d'espace (« *proof of space* ») ou les preuves de possession (« *proof of stake* »).

Les clients « légers »

Une autre dérive des utilisations des bitcoins remet en question le modèle entièrement distribué sur lequel repose la confiance dans cette crypto-monnaie. Afin de s'assurer qu'une transaction est valide, chaque acteur devrait stocker l'intégralité de la *blockchain* afin de s'assurer de la validité des transactions le concernant. Cette chaîne représente aujourd'hui plusieurs gigaoctets de données, et ne serait-ce que le fait de l'obtenir requiert beaucoup de temps. Pour cette raison, des clients dits légers sont apparus qui ne stockent pas la *blockchain* et préfèrent s'adresser au reste du réseau pour s'assurer qu'une transaction a bien été effectuée.

La gestion des portefeuilles

De nombreux utilisateurs préfèrent déléguer la gestion de leur portefeuille, et donc celle de leurs clés privées, à des sites spécialisés. Ces derniers deviennent donc des cibles de choix pour des attaquants cherchant à dérober ou à détruire des crypto-monnaies. De nombreuses attaques de ce type ont été perpétrées contre des sites de stockage et d'échange de bitcoins, tel le site d'échange de crypto-monnaies MtGox. Aucun recours technique n'est possible dès lors que le tiers dit « de confiance » n'est plus capable de restituer les clés secrètes des clients.

La sécurité d'Ethereum

Ethereum utilise la *blockchain* de la même façon que bitcoin : des utilisateurs proposent d'ajouter des transactions et celles-ci sont regroupées en blocs par des mineurs qui vérifient la validité des transactions et cherchent ensuite à résoudre un puzzle cryptographique pour ajouter officiellement le nouveau bloc à la chaîne. Tandis que bitcoin utilisait un langage de script simple de transfert monétaire, Ethereum permet d'utiliser un langage de script Turing-complet, et ainsi de déployer un « ordinateur-monde ». Chaque « transaction » permet de créer ou d'exécuter un programme de façon distribuée entre tous les mineurs. Cette richesse s'accompagne malheureusement de l'introduction de nouveaux problèmes de sécurité, qui ne sont plus d'ordre cryptographique mais d'ordre informatique et qui concernent la sûreté des langages de programmation.

Ethereum Virtual Machine et *Solidity*

Les données stockées sur la *blockchain* permettent de déterminer l'état courant de l'*Ethereum Virtual Machine*, une machine virtuelle complètement distribuée. À chaque utilisateur, mais aussi à chaque programme stocké dans la *blockchain*, est associée une variable « balance » représentant la quantité de crypto-monnaie en sa « possession ».

Afin de faciliter la conception des programmes destinés à être déployés sur l'*Ethereum Virtual Machine* (EVM), un langage de haut niveau nommé Solidity est utilisé, avant d'être compilé. Cependant, la sémantique d'appel de fonctions de Solidity est assez complexe. En particulier, chaque programme peut définir une fonction de « *callback* » qui sera appelée ou non, en fonction de la syntaxe utilisée.

L'attaque contre DAO

Le principe du programme DAO était relativement simple : recevoir de divers utilisateurs des dons destinés à un autre utilisateur, puis transférer à l'utilisateur final ces dons quand celui-ci en fait la demande (c'est une forme de cagnotte virtuelle).

Le code du programme peut être simplifié et ramené à deux fonctions :

- « *donate (address)* », qui enregistre le fait que la quantité de crypto-monnaie renseignée lors de l'appel de fonction est maintenant associée à l'adresse (*address)* dans un tableau interne au programme DAO ;
- « *withdraw (amount)* », qui permet à un utilisateur de demander au programme DAO de lui reverser une [certaine] quantité (*amount)* de crypto-monnaie.

Malheureusement, la syntaxe utilisée par le programme DAO avait la propriété d'appeler la fonction de « *callback* » du programme appelant. Ainsi, un utilisateur malveillant pouvait écrire un programme demandant le retrait d'une somme légitime en appelant « *withdraw* », dont la fonction de « *callback* » émettait un appel similaire permettant de récupérer deux fois la somme due, tirant ainsi profit d'un « *bug* » dans le programme originel.

Conclusion

Les aspects sécuritaires des *blockchains* et de leurs applications sont nombreux et une faille à n'importe quel niveau de la chaîne peut avoir des conséquences catastrophiques. Avant de parvenir à un niveau de maturité et de confiance permettant des applications critiques, de nombreuses innovations sont encore nécessaires.