LOGIQUE & CALCUL

Les preuves de travail

Donner à des ordinateurs des problèmes à résoudre permet de les freiner. Cette procédure est utile pour lutter contre les attaques sur le réseau Internet, pour limiter l'envoi de spams ou pour organiser des courses entre machines.

Jean-Paul DELAHAYE

Aujourd'hui, les ordinateurs qui sont à notre disposition sont très puissants et nous nous en réjouissons, car cela les rend plus utiles : ils manipulent de longs textes, des photos ou même de la vidéo, ils explorent rapidement des pages Internet à l'autre bout du monde, ils exécutent des programmes de jeux aux effets graphiques époustouflants, etc.

Cependant, cette puissance est parfois mal utilisée et crée des dysfonctionnements. Ainsi, n'importe qui peut envoyer des dizaines de milliers de messages électroniques qui encombrent les boîtes aux lettres au point de les rendre inutilisables : c'est le problème des courriels non sollicités, ou spams. Par ailleurs, quelques ordinateurs coordonnés sont en mesure de soumettre, plusieurs milliers de fois par seconde, des requêtes à un même service en ligne et ainsi d'en empêcher le fonctionnement : ce sont les attaques par « déni de service » ou DoS (pour *Denial of Service*).

Une idée étonnante a été proposée pour contrôler les effets de cet excès de puissance de nos machines. Il s'agit de soumettre aux ordinateurs des problèmes à résoudre, afin qu'ils calculent longuement pour trouver la solution ; alors seulement, ces ordinateurs ont le droit d'accéder à une boîte à lettres ou à un service informatique en ligne.

Ces « preuves de travail » jouent un rôle de plus en plus important. Elles sont combinées aux outils de base de la cryptographie moderne (chiffrage, signature numérique, authentification, etc.) pour concevoir des protocoles complexes réalisant des opérations considérées comme impossibles il y a peu. Les monnaies numériques décentralisées, dont le célèbre et controversé *Bitcoin*, fonctionnent toutes selon un protocole où les « preuves de travail » jouent un rôle essentiel. Nous y reviendrons.

Ces preuves de travail sont toujours des problèmes dont la solution est difficile à trouver, mais facile à vérifier. L'ordinateur qui doit résoudre le problème soumis ne peut le faire rapidement, mais quand il propose une solution pour accéder au service qui l'exige, la vérification de la solution est immédiate. Il existe de nombreuses méthodes pour fabriquer les problèmes des preuves de travail. On préfère bien sûr celles dont il est facile d'ajuster la difficulté à la situation, pour que le temps de résolution ne soit ni trop court ni trop long.

Des preuves avec ou sans échanges de messages

Deux types de preuves de travail sont utilisés : les systèmes fonctionnant par échanges multiples de messages et les systèmes à un seul envoi.

Dans le cas des « preuves de travail par échanges multiples », voici les étapes du protocole. La machine D (pour *Demandeur*) qui souhaite accéder au service S le contacte ; le service S élabore un problème et l'envoie à D ; la difficulté du problème est ajustée en prenant par exemple en compte la surcharge du service S à l'instant de la demande, ou ce que S sait déjà de D ; la machine D résout le problème, ce qui exige d'elle un certain travail, donc du temps ; quand D a résolu le problème, il envoie la solution au service S ; le service S vérifie que la solution soumise est bonne, ce qui est facile pour lui, et, si c'est le cas, donne accès au service, c'est-à-dire accepte de placer un message dans la boîte à lettres, ou ouvre la page d'accueil du service en ligne que demande D.

Dans le cas du protocole des « preuves de travail sans échange », tout se passe au cours d'un seul envoi. Selon des paramètres que le demandeur D connaît sans avoir à contacter le service S, un problème à résoudre est défini. La façon de le définir est fixée une fois pour toutes et connue d'avance. Le contenu du message que D veut envoyer détermine par exemple le problème (et donc le problème à résoudre change à chaque demande, ce qui est évidemment essentiel). Le demandeur D connaît donc, sans contacter le service S, le travail qu'il doit réaliser. Quand D a résolu le problème, ce qui lui prend un peu de temps, il envoie la solution trouvée au service S qui ne donne suite à sa demande que si la solution proposée est correcte.

La seconde méthode est souvent préférable : elle est plus simple pour le service qui n'a pas à s'occuper de définir un pro-

© Pour la Science - n° 438 - Avril 2014

pls_0438_p086_092_logique_calcul.indd 86 04/03/14 12:13

blème particulier à chaque demande. Mais la première méthode permet une prise en compte plus fine de la surcharge du service à l'instant de la demande ou d'informations connues de S concernant D.

Le délai nécessaire pour que le demandeur résolve le problème posé est le plus souvent fondé sur un calcul à faire par le demandeur. Le temps qu'il met à résoudre le problème dépend donc de sa puissance de calcul et les machines puissantes sont avantagées. Aussi a-t-on envisagé des preuves de travail exigeant beaucoup de mémoires (plutôt que beaucoup de calculs) ou nécessitant la recherche d'informations sur le réseau. Dans ce dernier cas, les machines puissantes ne sont pas avantagées, puisque ce qui nécessite du temps est lié à la vitesse de circulation des informations sur le réseau, qui est la même pour tout le monde.

L'idée des preuves de travail est due à Cynthia Dwork et Moni Naor qui, dès 1993, suggérèrent cette méthode pour lutter contre les spams. Le système *Hashcash* conçu par Adam Back en 1997 (indépendamment de C. Dwork et M. Naor) et servant aujourd'hui de base dans la plupart des mises en œuvre pratiques des preuves de travail, utilise des « fonctions à sens unique » pour définir les problèmes soumis et en contrôler finement la difficulté *(voir l'encadré 2 pour des exemples)*.

Une fonction f à sens unique se calcule facilement, mais s'inverse difficilement. Autrement dit, il est rapide de passer de x à $f(x) = y$, mais difficile de trouver, à partir d'un y donné, un x tel que $y = f(x)$. On exigera souvent aussi que les valeurs de $f(x)$ quand x varie se présentent comme si elles étaient tirées au hasard.

Fonctions à sens unique, avec ou sans paramètre

Si l'on dispose d'une telle fonction f, on en déduit facilement une preuve de travail : le service S choisit un y et exige du demandeur D qu'il trouve un x tel que $f(x) = y$. Le demandeur ne peut guère faire mieux qu'essayer des valeurs de x, jusqu'à trouver la bonne, ce qui est impossible à faire rapidement. Il s'agit ici d'une impossibilité pratique : trouver le x est possible en essayant de manière ordonnée toutes les x envisageables, mais cela nécessite du temps.

Plus subtil, car cela ne demande pas l'intervention de S pour la formulation du problème (donc utilisable pour les preuves de travail sans échange), on utilise une fonction à sens unique dépendant d'un paramètre p, $f_p(x)$, et dont le résultat, y, peut être interprété comme un nombre réel positif. Une telle interprétation est toujours possible : on écrit y en binaire, par exemple 01001110, et on assimile cette suite de 0 et de 1 au nombre réel dont l'écriture en base 2 est 0,01001110. Le service S attend que D lui propose un x tel que $f_p(x) < y$, où p est par exemple le message que D veut envoyer et y un nombre réel, appelé seuil, fixé à l'avance par S. La valeur de y détermine très précisément la difficulté de la preuve de travail, car plus y est petit, plus il est difficile de trouver un x convenable.

Les preuves de travail fournissent aussi un moyen simple d'organiser des courses entre machines sur un réseau. Or ces courses sont au cœur des protocoles de fonctionnement des cryptomonnaies.

Ces monnaies, dont le *bitcoin* est l'exemple le plus connu, fonctionnent sans autorité centrale, la gestion des comptes étant opérée par les machines des volontaires qui se contrôlent mutuellement. Ces contrôles empêchent toute manipulation du cahier de compte de la monnaie (la « blockchain »), lequel indique qui détient des devises *(voir Pour la Science, décembre 2013)*.

1. L'idée des preuves de travail

Pour limiter le pouvoir de nuisance que chacun possède du fait de la puissance de son ordinateur, on peut soumettre des énigmes obligeant les machines à de longs calculs et dont seul l'aboutissement leur donne le droit de mener une action, comme envoyer un message électronique ou accéder à un service en ligne. Une machine soumise à ces énigmes ne pourra pas envoyer des milliers de messages non sollicités (spams), ou participer à une tentative de mise en panne d'un service informatique en le submergeant de requêtes (attaque par déni de service).

Le temps de calcul imposé est un prix à payer pour que l'ordinateur acquière le droit de mener une action. Il doit travailler et le prouver par un résultat. La solution produite est la preuve qu'il a fourni ce travail. C'est l'idée des « preuves de travail ».

Les problèmes qu'on demande de résoudre pour ces défis doivent être asymétriques. Il faut que le calcul de la solution prenne du temps, mais que la vérification soit rapide ; sinon, celui qui se protégerait par les preuves de travail serait autant ennuyé que ceux dont il tenterait de se protéger.

On connaît une multitude de problèmes asymétriques. En voici deux exemples :

– L'entier b étant fixé, et p étant un grand nombre premier, trouver un nombre a tel que $a^2 = b \bmod p$. Ce problème de la « racine carrée modulo un grand nombre premier » a été proposé comme preuve de travail dès 1993 par Cynthia Dwork et Moni Naor.

Jean-Michel Thiriet

– Trouver les deux nombres premiers p et q dont le produit est le nombre n qu'on vous a donné sans vous avoir dévoilé p ni q. Il est facile de formuler de telles énigmes – on multiplie deux grands nombres premiers entre eux, ce qui produit n – mais aucun algorithme de factorisation rapide n'est connu, et il faut donc du temps pour retrouver p et q à celui qui ne connaît que n. Il ne faut pas prendre p et q trop grands, car le problème de la factorisation de n deviendrait impossible à résoudre en un temps raisonnable.

Un troisième exemple – le plus important, car le plus utilisé – est expliqué en détail dans les encadrés 2 et 3 et se fonde sur les fonctions « à sens unique ».

pls_0438_p086_092_logique_calcul.indd 87 04/03/14 12:13

Pour qu'il y ait un intérêt pratique à participer à ce contrôle collectif sans lequel la cryptomonnaie n'existe plus, une récompense est donnée à intervalles réguliers à l'une des machines tenant les comptes. Aujourd'hui par exemple, 25 *bitcoins* sont distribués toutes les dix minutes par le protocole *Bitcoin*. Cette machine gagnante est désignée par un procédé fondé sur une preuve de travail dont le fonctionnement doit être parfait pour que le protocole résiste aux attaques. C'est la plus belle application à ce jour de l'idée des preuves de travail, et de nombreuses caractéristiques de *Bitcoin* en dépendent. Donnons quelques précisions.

Chaque machine candidate essaie de résoudre un problème dont les paramètres sont fixés par le protocole *Bitcoin* et qui est formulé sans intervention d'un service centralisé. Il s'agit d'un problème du type « trouver un x tel que $f_p(x) < y$ » évoqué plus haut. La fonction f à sens unique est connue de tous, et c'est la fonction de hachage SHA-256. On l'utilise beaucoup en cryptographie et les spécialistes s'accordent à considérer qu'elle a les propriétés voulues pour servir dans une preuve de travail. La première machine qui résout le problème gagne la compétition et emporte les 25 *bitcoins*. Le protocole ajuste régulièrement le seuil donné par le nombre y, qui fixe la difficulté de la preuve de travail en prenant en compte les concours précédents, de telle façon que le temps moyen entre deux concours soit de dix minutes environ.

Des preuves de travail pour les bitcoins

Cette émission contrôlée, prévisible et faible de *bitcoins* toutes les dix minutes est la seule façon d'en créer. Cela a pour intérêt que les détenteurs de *bitcoins* sont certains de ne pas être victimes des phénomènes inflationnistes provoqués par l'émission massive de devises, comme c'est parfois le cas pour les monnaies contrôlées par les banques centrales.

Participer à ce concours porte le nom de « minage », car le travail de calcul fait par les machines évoque celui d'un mineur dans une mine d'or, qui le conduit à s'enrichir s'il a la chance de trouver une pépite. Les informaticiens propriétaires des machines participant à ce concours permanent, répété toutes les dix minutes, se nomment entre eux des « mineurs ».

Il est important de comprendre que puisqu'il n'y a pas d'autorité centrale gérant les *bitcoins* (il en va de même pour les autres cryptomonnaies), l'acceptation de la preuve de travail se fait collectivement par tous les mineurs. Quand l'un d'eux a réussi à relever le défi de trouver un x tel que $f_p(x) < y$, il envoie un message sur le réseau, qui prévient les autres mineurs. Ceux-ci vérifient alors, chacun de son côté, que la solution proposée est correcte et se remettent au travail pour résoudre le défi suivant.

Le temps nécessaire pour qu'un gagnant soit trouvé n'est pas fixé avec précision : la

2. Le hachage

Une fonction de hachage associe, à un x donné (une suite de caractères ou un nombre) de longueur quelconque une valeur $f(x)$ de taille fixe nommée « empreinte de x » ou « condensé de x » ou *hash*. Le passage de x à $f(x)$ se fait par une série d'opérations qui commencent en général en découpant x en petits morceaux (d'où le nom de hachage), qui sont ensuite rassemblés et emmêlés de façon à obtenir un mélange aussi parfait que possible. Pour la fonction classique de hachage SHA-256, la longueur de $f(x)$ est de 256 bits ; il y a donc $2^{256} = 1{,}157 \times 10^{77}$ valeurs possibles, ce qui est considérable.

Les fonctions de hachage à usage cryptographique ont les propriétés suivantes.

(a) Elles sont « à sens unique » : le passage de x à $f(x)$ se calcule facilement et rapidement, mais pour un y donné, trouver un x tel que $y = f(x)$ est impossible en pratique.

(b) Il est impossible en pratique de trouver deux valeurs x et x' différentes telles que $f(x) = f(x')$.

(c) Elles produisent des valeurs assimilables à des valeurs aléatoires uniformes (prises dans l'ensemble des valeurs possibles) et, en particulier, même si x et x' sont deux suites de caractères n'ayant qu'un seul caractère différent, les valeurs $f(x)$ et $f(x')$ n'ont aucun lien apparent.

Voici un exemple. La donnée x est transformée en une suite de 0 et de 1 qui est découpée en morceaux de longueur 8. Les morceaux sont ensuite additionnés comme des nombres en base 2. On ne garde du résultat que les quatre chiffres en positions 3, 4, 5 et 6 à partir de la droite :

x=01010010100101010010010101010 → 01010010 + 10010101 + 00100101 + 010 = 100001110 → $f(x)$ = 0011.

Cet exemple est bien sûr trop simple pour avoir les propriétés *(a)*, *(b)* et *(c)*. La fonction de hachage SHA-256 mélange soigneusement les bits de la donnée x selon un algorithme public (voir http://en.wikipedia.org/wiki/SHA-2).

Les fonctions de hachage cryptographiques ont de multiples usages dont voici trois exemples.

– Elles permettent d'assurer l'intégrité d'un gros fichier x, auquel on lui associera son empreinte $f(x)$, beaucoup plus courte. Celui qui téléchargera x s'assurera que x lui parvient sans erreur et en totalité en calculant l'empreinte du fichier qu'il aura reçu, qui doit être la même que celle publiée sur le site. Si x a mal été copié, il le saura, car la probabilité que la modification n'ait pas changé l'empreinte est infinitésimale.

– Au lieu de mémoriser les mots de passe d'accès aux comptes sur un service en ligne, celui-ci ne mémorise que les empreintes des mots de passe. Le serveur peut vérifier que les utilisateurs donnent les bons mots de passe, mais si un pirate accède à sa liste d'empreintes, il ne pourra rien en faire.

– Pour produire une suite de nombres aléatoires dépendant d'un fichier x, on calculera par exemple $f(x.1)$, $f(x.2)$,... où $x.i$ désigne le fichier x suivi de l'écriture du nombre i (par exemple en binaire). En changeant le x, on a autant de suites de nombres aléatoires qu'on veut (ce qui permet par exemple de chiffrer des messages).

L'importance des fonctions de hachage est telle, en cryptographie, qu'un grand soin est mis à les élaborer et à les soumettre à toutes sortes de tests et d'attaques. Le NIST, aux États-Unis, propose des fonctions bien contrôlées. La fonction SHA-256 utilisée par *Bitcoin* en est une. Le NIST publie aussi des recommandations concernant le choix et l'usage des fonctions de hachage. Il a récemment organisé une compétition pour concevoir une nouvelle génération de fonctions de hachage qui a abouti à la norme SHA-3 en 2012. Voir : http://www.nist.gov/itl/csd/ct/hash_competition.cfm.

© Pour la Science - n° 438 - Avril 2014

pls_0438_p086_092_logique_calcul.indd 88 04/03/14 12:13

3. L'inversion d'une fonction de hachage

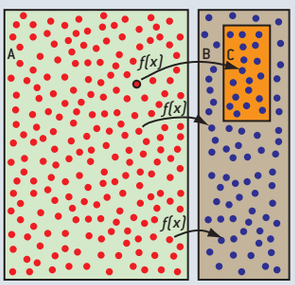
Explicitons une méthode pour définir et ajuster l'énigme d'une preuve de travail.

La fonction f de hachage transforme une suite x de 0 et de 1 de longueur quelconque en une autre, y, de longueur 256 (comme le SHA-256). Puisqu'il est impossible en pratique de trouver pour un y donné un x tel que $f(x) = y$ (inversion complète de f), on crée une énigme plus raisonnable et exigeant un travail faisable en ne demandant qu'une inversion partielle de f : trouver un x tel que $f(x)$ soit une suite de caractères commençant par k fois '0' (donc une suite de la forme 0 0 0 ... 0 a_1 a_2 a_3... a_n).

Plus k est grand, plus il sera difficile de trouver un x satisfaisant. On peut même être plus précis : pour trouver un tel x, il faut en moyenne essayer environ 2^k valeurs de x. L'énigme « trouver un x tel que $f(x)$ commence par dix fois '0' » exige donc environ $2^{10} = 1024$ calculs de $f(x)$. Pour 20 fois '0', il en faudrait un million environ. En choisissant k, on ajuste donc la difficulté de l'énigme qu'on formule. On a là une méthode rapide pour formuler des énigmes de « preuve de travail » dont la difficulté est facile à évaluer.

Dans les preuves de travail (pour se protéger des spams, ou interdire les attaques par déni de service), il faut éviter que les mêmes énigmes soient soumises plusieurs fois. Si c'était le cas, des bibliothèques de solutions seraient calculées par avance et les attaquants, au lieu de faire les calculs attendus, se contenteraient d'aller y puiser les solutions des problèmes qu'on leur soumet.

Pour éviter cela, on fait dépendre l'énigme posée d'un paramètre p. On demandera par exemple de trouver un x commençant par la chaîne $p = 0100011010100$ et tel que $f(x) < y$.



La fonction f envoie les éléments x de l'ensemble A (grand) dans l'ensemble B (petit). Inverser partiellement f, c'est trouver un élément x tel que f(x) soit dans C. Plus C est petit, plus la tâche est difficile.

Le nombre de paramètres possibles p étant illimité, l'idée de constituer des bibliothèques de solutions calculées par avance n'est plus applicable.

Les énigmes possibles posées par la méthode de l'inversion partielle des fonctions de hachage sont donc à la fois parfaitement ajustables, et si nombreuses que celui à qui on les propose ne peut que se soumettre et faire le travail de les résoudre en essayant de très nombreux x, ce qui en moyenne lui prendra le temps qu'on aura choisi. Quand il aura trouvé la solution, il aura vraiment prouvé qu'il a travaillé !

Notons encore que si l'on est gêné par le fait que le temps de travail varie trop, selon que celui qui tente de résoudre le problème posé a de la chance ou pas (variance importante), on peut remplacer la demande de résoudre un problème, par la demande de résoudre plusieurs petits problèmes, ce qui a pour effet de diminuer la variance du temps nécessaire pour aboutir.

durée moyenne est de dix minutes, mais elle sera plus longue ou plus courte selon que les mineurs auront eu de la chance ou pas dans leur tentative de résolution du problème de la preuve de travail.

Une course folle à la puissance...

Si chaque machine n'utilisait que sa propre puissance pour gagner, la probabilité pour chacune de gagner serait proportionnelle à sa puissance. Cependant, comme la fonction f est fixée et connue de tous, le problème de trouver un x tel que $f_p(x) < y$ est très particulier. Les personnes intéressées par le minage des *bitcoins* se sont mises à utiliser des puces conçues spécialement pour résoudre rapidement le type particulier de problèmes posés dans les preuves de travail de *Bitcoin*. Une course folle à la puissance de minage en a résulté, conduisant à une véritable industrie du minage !

La puissance de minage d'un système informatique se mesure en comptant le nombre de calculs de type $x \to f(x)$ que le système est capable de faire par seconde, pour la fonction SHA-256. On est ainsi passé de 24×10^{12} calculs de valeurs de f par seconde pour l'ensemble des mineurs en janvier 2013 à 11×10^{15} en janvier 2014, soit une multiplication par plus de 500 en un an.

Le coût en électricité de ces minages (coût des « preuves de travail ») a été évalué. Certains ont calculé qu'il est de plus de dix millions d'euros par jour. C'est plus que la valeur des *bitcoins* gagnés chaque jour (environ deux millions d'euros). Les spécialistes ne sont pas d'accord sur ce nombre de dix millions, mais, même imprécis, il signifie qu'être mineur de *bitcoins* aujourd'hui n'est pas nécessairement rentable. Acheter du matériel et dépenser de l'électricité en le faisant fonctionner est un pari risqué, dont le succès dépend de deux facteurs très difficiles à contrôler ou même à anticiper : la puissance de calcul de vos concurrents, et le cours extrêmement volatil du *bitcoin*.

Notons que certains pirates informatiques ont conçu des programmes, nommés chevaux de Troie, qui s'installent par le biais du réseau sur une multitude d'ordinateurs (le vôtre peut-être) et leur font exécuter le minage. Bien évidemment, lorsque le minage réussit, la solution est transmise au pirate qui, pour son propre compte, touche les 25 *bitcoins*. C'est vous qui payez l'électricité, mais c'est lui qui empoche le bénéfice !

...et un énorme gâchis ?

Parmi les nombreuses critiques formulées à l'encontre de *Bitcoin*, l'une d'elles concerne justement les preuves de travail qui définissent le minage. Les sommes d'argent considérables dépensées en électricité et pour fabriquer du matériel spécialisé calculant le plus rapidement possible des $f_p(x)$ semblent absurdes, car en soi de tels calculs ne servent à rien. N'est-ce pas là un gâchis terrifiant, engendrant par ailleurs une importante pollution liée à l'électricité consommée, qu'il faut bien produire ?

Une double réponse a été donnée à cette critique. D'une part, il a été répliqué

pls_0438_p086_092_logique_calcul.indd 89 04/03/14 12:14

que les moyens dépensés pour le minage n'étaient pas vains, puisqu'ils permettent à ces monnaies de fonctionner. Fabriquer des billets de banques, les transporter dans des véhicules blindés, les enfermer dans des coffres que l'on fait garder : tout cela coûte cher et pourtant personne ne conteste la nécessité de ces dépenses puisqu'elles servent à faire fonctionner les monnaies fiduciaires habituelles émises par les banques centrales. Le *bitcoin* n'a pas besoin d'être imprimé (c'est une monnaie numérique), il n'y a pas besoin de voitures blindées pour son transport, mais il faut que sa gestion soit assurée. Le minage assure en partie cette gestion, qui n'est donc que l'équivalent des dépenses de fonctionnement des autres monnaies et n'a finalement rien d'absurde.

Concevoir des preuves de travail utiles

L'autre réponse est plus intéressante, elle est aujourd'hui encore en discussion. Les preuves de travail les plus habituelles (dont celle utilisée par *Bitcoin*) consistent à faire des calculs pour résoudre un problème qui, malheureusement, est sans intérêt réel : une fois que l'on a trouvé un x tel que $f_p(x) < y$, ce x n'est utile à personne et ne le sera sans doute jamais. Ne pourrait-on pas concevoir des preuves de travail fondées sur des problèmes dont la solution serait utile ?

On le sait, les mathématiciens aiment les nombres premiers dont la connaissance progresse et est utile (en particulier en cryptographie !). Divers groupes de nombres premiers sont jugés intéressants. Les paires de nombres premiers jumeaux sont les paires $(p, p+2)$ de nombres premiers ayant un écart de deux unités, par exemple (17, 19). On cherche toujours à prouver qu'il en existe une infinité. Il y a aussi les paires de nombres premiers de Sophie Germain (mathématicienne française qui vécut de 1776 à 1831) qui sont de la forme $(p, 2p+1)$, par exemple (3, 7), (5, 11), (11, 23). Elles conduisent aux chaînes de Cunningham $(p_1, p_2, \ldots, p_k)$ où chaque couple (p_i, p_{i+1}) est une paire de nombres de Sophie Germain : $p_2 = 2p_1 + 1$, $p_3 = 2p_2 + 1$, ..., $p_{k+1} = 2p_k + 1$, chaque p_i étant un nombre premier.

Des chercheurs tiennent à jour des sites Internet qui indiquent les records de longueur des chaînes de Cunningham. Sunny King a montré en 2013 qu'une telle recherche peut servir de base à des preuves de travail aux propriétés à peu près équivalentes à celles provenant des fonctions à sens unique.

S. King a créé en juillet 2013 sa propre cryptomonnaie, *Primecoin*, proche dans sa conception de *Bitcoin*, mais utilisant des preuves de travail qui conduisent à des chaînes de Cunningham nouvelles. Des chaînes de Cunningham records, de longueur 10, 11, 12 et 13, ont ainsi été découvertes, ce qui prouve l'efficacité de la méthode proposée et sa capacité à contribuer à la recherche mathématique.

Précisons toutefois que l'intérêt de connaître des chaînes de Cunningham n'est pas très grand, et que la sûreté des preuves de travail fondées sur leur recherche n'est pas aussi bonne que celle des preuves de travail fondées sur les fonctions à sens unique. Les utilisateurs des cryptomonnaies ont cependant fait bon accueil à cette nouveauté. Parmi toutes les cryptomonnaies existantes (il y en a aujourd'hui plus de 80, voir http://coinmarketcap.com/), *Primecoin* se classe onzième pour la capitalisation. Le total des *primecoins* émis vaut dix millions de dollars (le 7 février 2014). Qui a dit que les mathématiques n'étaient pas un bon moyen de gagner de l'argent ?

Un autre projet en cours de développement, nommé *Curecoin,* tente de concevoir une preuve de travail (et une cryptomonnaie associée) qui soit beaucoup plus clairement utile. Le calcul mené par les machines de-

4. Des preuves de travail utiles

Les preuves de travail utilisées aujourd'hui ont un côté absurde : on demande aux ordinateurs de résoudre des problèmes mathématiques qui n'ont aucun intérêt. Sunny King (pseudonyme d'une personne ou d'un groupe) a conçu une preuve de travail aussi facile à ajuster que l'inversion partielle des fonctions de hachage cryptographiques et qui permet de trouver des chaînes de nombres premiers intéressantes : les chaînes de Cunningham.

Ce sont des suites de nombres premiers $(p_1, p_2, \ldots, p_k)$ vérifiant : $p_2 = 2p_1 + 1$, $p_3 = 2p_2 + 1$, ..., $p_{k+1} = 2p_k + 1$. Ces chaînes de Cunningham sont reliées aux nombres premiers de Sophie Germain, une mathématicienne française (1776-1831) : un nombre premier p est un nombre premier de Sophie Germain si $2p + 1$ est aussi un nombre premier.

Ces nombres premiers particuliers ont un intérêt en raison d'un théorème de Sophie Germain lié au théorème de Fermat. Dans les chaînes de Cunningham, tous les nombres sont des nombres de Sophie Germain, sauf le dernier.

L'utilisation de ces preuves de travail où l'on recherche des nombres premiers est au cœur du fonctionnement de la cryptomonnaie *Primecoin*, concurrente de *Bitcoin*. Le travail de minage de *Primecoin* a aujourd'hui produit plusieurs chaînes nouvelles intéressantes. La chaîne de Cunningham de longueur 11 comportant les plus grands nombres premiers est l'une d'elles. Le premier nombre de cette chaîne record découverte le 3 août 2013 comporte plus de 200 chiffres. Il s'écrit :

$p_1 = k \times 151\# - 1$,

où k = 73 853 903 764 168 979 088 206 401 473 739 410 396 455 001 112 581 722 569 026 969 860 983 656 346 568 919

et où 151# désigne le produit des nombres premiers inférieurs ou égaux à 151 (voir http://users.cybercity.dk/~dsl522332/math/Cunningham_Chain_records.htm).

Sophie Germain à l'âge de 14 ans, portrait de Auguste Eugène Leray.

© Pour la Science - n° 438 - Avril 2014

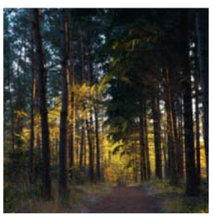
pls_0438_p086_092_logique_calcul.indd 90 04/03/14 12:14

© Thomas Bolognesi

Master 2

Économie
Énergie
Développement
Durable

FACULTÉ d'ÉCONOMIE de GRENOBLE



Deux parcours distincts

• **Le master Professionnel** forme de futurs cadres spécialistes de l'énergie pour des entreprises énergétiques, des grandes entreprises hors énergie, des cabinets de consultants, des collectivités territoriales et des organismes internationaux.

• **Le master Recherche** initie l'étudiant à la recherche en vue de son admission en thèse.

CONTENU

1er semestre : commun aux parcours Professionnel et Recherche

Fondamentaux de l'énergie
Climat et énergie
Marchés énergétiques internationaux
Industries de réseau
Stratégies et politiques d'innovation
Anglais pour l'énergie / autre langue vivante

2e semestre, parcours Professionnel

Institutions internationales et énergie
Politiques territoriales énergie-climat
Approches sectorielle et technologique énergie-climat
Stage

2e semestre, parcours Recherche

Economie de l'énergie
Environnement et innovation
Approches expérimentales
Mémoire

DÉBOUCHÉS

Chargé de mission ou de projets énergie
Analyste de marché énergétique
Responsable énergie
Expert-conseil énergie-climat
Economiste de l'énergie et de l'environnement
Enseignant-chercheur

INTERVENANTS

Conférenciers de renom
Enseignants et chercheurs de l'équipe Economie du Développement Durable et de l'Energie (CNRS-PACTE-EDDEN) et de Grenoble Applied Economics Laboratory (INRA-GAEL)

Les + du Master 2 EEDD

- Formation présentielle en alternance
- Formation à distance
- Large réseau de partenaires

POUR PLUS D'INFORMATION

http://www.upmf-grenoble.fr/master2-EEDD
contact : odile.blanchard@upmf-grenoble.fr

Université Pierre-Mendès-France
Faculté d'Économie
1241 rue des Résidences
F - 38400 Saint-Martin-d'Hères

upmf Grenoble Université Pierre Mendès France Sciences sociales & humaines

L'AUTEUR

J.-P. DELAHAYE est professeur à l'Université de Lille et chercheur au Laboratoire d'informatique fondamentale de Lille (LIFL).

BIBLIOGRAPHIE

S. King, **Primecoin : Cryptocurrency with prime number proof-of-work**, 2013 : http://primecoin.org/static/primecoin-paper.pdf

R. Alexander, **Determining electrical cost of Bitcoin mining**, *Bitcoin Magazine*, 18 déc. 2013 : http://bitcoinmagazine.com/8994/determining-electrical-cost-bitcoin-mining/

N. Popper, **Into the bitcoin mines**, *New York Times*, 12 décembre 2013 : http://dealbook.nytimes.com/2013/12/21/into-the-bitcoin-mines/

J.-P. Delahaye, ***Bitcoin*, la cryptomonnaie**, *Pour la Science* n° 434, décembre 2013.

Wikibooks, **Les fonctions de hachage cryptographiques**, http://fr.wikibooks.org/wiki/Les_fonctions_de_hachage_cryptographiques

A. Back, ***Hashcash*, a Denial of Service Counter-Measure**, 2002 ftp://sunsite.icm.edu.pl/site/replay.old/programs/hashcash/hashcash.pdf

C. Dwork et M. Naor, **Pricing via processing or combatting junk mail**, *Advances in Cryptology*, CRYPTO' 92, Lecture Notes in Computer Science, vol. 740, pp. 139-147, 1993.

vrait aider à découvrir comment se replient diverses protéines, ce qui, pour certaines d'entre elles, serait utile en médecine.

Les preuves de travail sont, on le voit, au cœur d'une petite révolution dont le succès, s'il se confirme, conduira à faire que les calculs réalisés pour la gestion des monnaies cryptographiques serviront la recherche mathématique ou médicale… ce qui n'est bien sûr pas le cas du transport des billets de banque, ni de la fabrication des coffres-forts.

Des jeux « prouvablement équitables »

Le monde des cryptomonnaies, avec ses preuves de travail à base de fonctions à sens unique, a amené la mise en place de casinos en ligne qui ne peuvent pas tricher, chose que le joueur lui-même vérifie sans intervention d'aucune autorité centrale.

Puisque le minage détermine un gagnant parmi les machines participantes et qu'on ne peut pas tricher (sous réserve que la fonction f utilisée possède bien les propriétés requises), on pourrait concevoir des jeux de hasard fondés directement sur le protocole de désignation d'un gagnant de 25 *bitcoins* toutes les dix minutes. Cependant, les chances de gagner d'un joueur dépendraient de la puissance de calcul de sa machine, ce qui ne serait pas juste.

Il existe des méthodes plus directes conduisant à des tirages au hasard parfaitement équitables et contrôlables pour, par exemple, lancer des dés en ligne sans qu'aucune tricherie ne soit possible, ni pour le casino ni pour le joueur. Ces protocoles sont connus depuis longtemps en cryptographie (Manuel Blum a proposé une méthode de tirage à pile ou face par téléphone en 1981), mais c'est seulement depuis peu que des casinos en ligne proposent aux joueurs des jeux « prouvablement équitables ». On peut donc aujourd'hui jouer en ligne à des jeux de hasard dont l'honnêteté est garantie par la cryptographie mathématique, et cela sans faire intervenir aucune autorité de contrôle des casinos auxquels on se connecte. Le paiement des taxes dues par le casino et son honnêteté commerciale (le casino paye-t-il toujours ce qu'il doit aux joueurs ?) ne sont pas garantis par ces protocoles, qui ne concernent que le hasard des tirages aléatoires.

Voici, ci-dessous, un protocole réalisant un tirage au sort ayant une chance sur k de réussir (k étant un entier supérieur ou égal à 2). Pour jouer, vous engagerez la somme S ; si le tirage vous est défavorable, le casino gardera votre mise ; si vous gagnez, le casino vous rendra votre mise, y ajoutera $(k-1)S$, et enlèvera par exemple un pour cent du total pour ses frais.

Protocole prouvablement équitable pour effectuer un tirage ayant une chance sur k de réussir (on prendra k = 2 pour simuler un lancer de pièce à pile ou face).

1) Le casino choisit un nombre x, et envoie le résultat du calcul de $f(x) = y$ au joueur. La fonction f est une fonction à sens unique produisant des résultats assimilables à du hasard, comme c'est le cas de la fonction SHA-256 utilisée par *Bitcoin*.

2) Le joueur envoie au casino un nombre z qu'il choisit comme il veut, éventuellement avec l'aide de son ordinateur.

3) Le casino calcule $t = x + z$, puis $u = f(t) \bmod k$ (le reste de la division de $f(t)$ par k).

4) Si $u = 0$, le joueur a gagné, sinon il a perdu.

5) Après le tirage, le casino fait parvenir x au joueur. Le joueur contrôle que le casino n'a pas triché en vérifiant que le y qu'il avait reçu avant de choisir z est bien $f(x)$; il contrôle aussi les calculs de $t = x + z$ et de $u = f(t) \bmod k$.

Le casino n'a pas pu manipuler le résultat, car, quand il a choisi x, il ne connaissait pas z. Le joueur n'a lui non plus pas pu manipuler le résultat, car, quand il a choisi z, il ne connaissait pas x. Le protocole est donc équitable et il n'existe aucune possibilité de tricherie, ni pour le casino ni pour le joueur.

Les outils de la cryptographie mathématique semblent concerner de plus en plus d'aspects de la vie économique et numérique. Les protocoles à base de preuves de travail et de fonctions à sens unique non seulement conduisent à mieux contrôler ce qui se passe sur les réseaux, mais ouvrent aussi des perspectives nouvelles, comme celles des cryptomonnaies et des jeux prouvablement équitables. ■

© Pour la Science - n° 438 - Avril 2014

pls_0438_p086_092_logique_calcul.indd 92 04/03/14 12:14