3

UN SECRET PARTAGÉ EN MORCEAUX

Il est possible de partager un message M en n morceaux $M_1, M_2, \ldots, M_n$ de telle façon que l'on n'ait aucune information sur M tant que l'on ne dispose pas de tous les n morceaux. Expliquons comment. On suppose M donné sous forme binaire et ayant la longueur L.

Pour $M_1, M_2, \ldots, M_{n-1}$, on prend des suites de 0 et de 1 aléatoires, différentes et indépendantes, chacune de longueur L. Pour définir M_n, on procède bit par bit.

Pour le premier bit de M_n, on prend un 0 ou un 1 de façon que le nombre de 1 en première position dans les messages $M_1, M_2, \ldots, M_n$ soit pair si le premier bit de M est 0, et impair sinon.

Pour le deuxième bit de M_n, on procède de la même façon à partir des bits en deuxième position de M et de $M_1, M_2, \ldots, M_{n-1}$. Et ainsi de suite (la procédure est illustrée ci-dessous pour $n = 10$, sur un message court).

Connaître $M_1, M_2, \ldots, M_n$ permet de reconstituer M, mais connaître $n - 1$ de ces messages ne donne aucune information sur M.

On peut faire mieux et partager un message M en n morceaux de telle façon que :
– connaître k morceaux, quels qu'ils soient ($k < n$), redonne M ;
– connaître moins de k morceaux ne donne aucune information sur M.

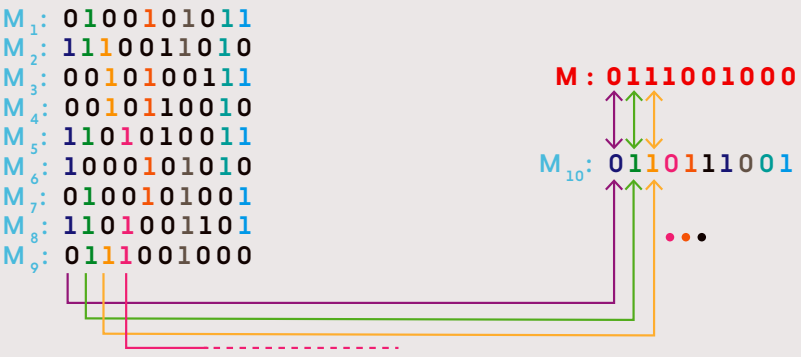
Plusieurs méthodes ont été proposées, dont celle d'Adi Shamir (« How to share a secret », *Communications of the ACM*, vol. 22(11), pp. 612-613, 1979).

Son idée repose sur le fait que disposer de k valeurs d'un polynôme P de degré $k - 1$ permet de le déterminer, mais que c'est impossible si l'on en connaît moins.

Le premier bit de M étant d_1, on commence par trouver un polynôme P_1 de degré $k - 1$ dont la valeur en 0 est d_1. Ensuite, pour chaque i de 1 à n, on fixe comme premier élément du message M_i la valeur $P_1(i)$.

On fait de même avec le second bit d_2 de M : on choisit un polynôme P_2 de degré $k - 1$ dont la valeur en 0 est d_2. Puis, pour chaque i de 1 à n, on fixe comme second élément du message M_i la valeur $P_2(i)$. Et ainsi de suite.

En conséquence :
– celui qui connaît k messages parmi $M_1, M_2, \ldots, M_n$ connaîtra le 1er bit de M, car il peut calculer le polynôme P_1. Il connaîtra le 2e bit de M, car il peut calculer le polynôme P_2. Etc.
– celui qui connaît moins de k messages parmi $M_1, M_2, \ldots, M_n$ ne peut rien connaître de M.



> programmé fera l'affaire : vous délivrerez le message à la date voulue.

Ces méthodes sont intéressantes, mais uniquement pour des délais assez courts ne dépassant pas quelques mois ou années, car les batteries du téléphone ou de l'ordinateur caché doivent tenir jusqu'au moment du déclenchement. Elles ne permettent pas d'envoyer un message un siècle à l'avance.

De plus, les solutions avec téléphone ou ordinateurs cachés ne produisent pas un fichier informatique qu'on laisse circuler et qui est tel que tous ceux le détenant à l'heure voulue en découvrent le message sans avoir à interagir avec autre chose. Elles ont aussi l'inconvénient d'être lourdes à mettre en œuvre, sans parler du risque que le dispositif caché soit découvert et arrêté ou détruit par accident. Réfléchissons à mieux.

FAIRE APPEL À UN OU PLUSIEURS TIERS DE CONFIANCE

Une idée évidente est celle du tiers de confiance. Vous confiez le message, placé dans une enveloppe cachetée par exemple, à un huissier en lui indiquant la date à laquelle il devra ouvrir l'enveloppe et en délivrer le contenu. Vous pouvez améliorer la robustesse du procédé en utilisant plusieurs tiers de confiance dont aucun ne pourra divulguer seul le message avant l'instant choisi par vous. Voici la méthode.

Vous découpez le message en n morceaux dont aucun ne permet d'avoir la moindre idée du message complet et dont même la connaissance de $n-1$ morceaux ne révèle rien (*voir l'encadré 3*). Vous confiez ces n morceaux à n tiers de confiance en leur demandant de se contacter les uns les autres à la date voulue pour s'échanger les bouts du message, le reconstituer et le rendre public.

Le risque sera cependant que l'un des tiers de confiance soit indisponible le jour voulu, malade ou décédé par exemple. Cela empêcherait que le message soit reconstitué et délivré. Mais il existe des procédés de découpage d'une information, inventés en 1979 par Adi Shamir et George Blakley, qui résolvent le problème. Le message est découpé en n morceaux qui ont la propriété que dès que l'on dispose de k morceaux ($k<n$) du découpage, il devient possible de retrouver tout le message, et qu'avec moins de k morceaux, rien ne peut être dévoilé du message découpé (*voir l'encadré 3*).

Ces idées résolvent sans doute un certain nombre de cas, mais, comme précédemment, ce ne sont que des méthodes partiellement satisfaisantes : elles ne produisent pas une capsule temporelle, c'est-à-dire un message qui libère seul son information à la date voulue à ceux qui le détiennent.

Explorons d'autres idées. Ronald Rivest

(le R du protocole de chiffrement à double clé RSA), Adi Shamir (le S de RSA) et David Wagner ont proposé en 1996 une méthode consistant à utiliser un protocole de chiffrement dont on efface volontairement la clé de déchiffrement. La seule façon de lire le message chiffré est alors de mener une attaque par calculs massifs dont on évalue la durée. Le message est illisible pendant tout le temps nécessaire aux calculs. Ceux qui les effectuent obtiennent au bout du temps prévu le moyen de déchiffrer le message. Cette fois, on a bien un message chiffré qui peut circuler et même être dupliqué et qui est protégé en lui-même par un système ne permettant pas d'en lire le contenu avant une certaine date. La méthode est astucieuse, mais on doit noter deux points.

UN RETARD IMPOSÉ PAR DES CALCULS NON PARALLÉLISABLES

Tout d'abord, le message ne sera déchiffré que par ceux qui effectueront le calcul massif et l'auront poursuivi pendant la durée nécessaire. C'est coûteux et cela demande des compétences; le message ne sera donc accessible qu'à certains acteurs très particuliers ayant accepté de s'en occuper en continu. De plus, si vous ne vous occupez pas de la capsule temporelle dès qu'elle est émise, vous ne pourrez disposer du message qu'avec retard.

Ensuite, on ne connaît pas à ce jour de méthode permettant d'ajuster avec précision la difficulté de la recherche de la clé. On peut d'ailleurs remarquer que, en 1996, les trois chercheurs ont décrit leurs idées en évoquant un système où le calcul massif consistait à calculer successivement des carrés de nombres entiers.

En 1999, un défi a été lancé concernant ce procédé, le « LCS35 Time Capsule Crypto-Puzzle ». Le message caché ne devait pouvoir se libérer que trente-cinq ans plus tard environ, donc vers 2034. Cette évaluation de la résistance du problème tenait compte de la loi de Moore, selon laquelle nos capacités de calcul doublent tous les dix-huit à vingt-quatre mois environ. Mais les progrès en matière de calcul de carrés de nombres entiers ont été plus rapides que prévu et le défi est tombé en 2019, quinze ans plus tôt que ce qui était programmé. Ronald Rivest a dû reconnaître qu'il s'était trompé dans son anticipation.

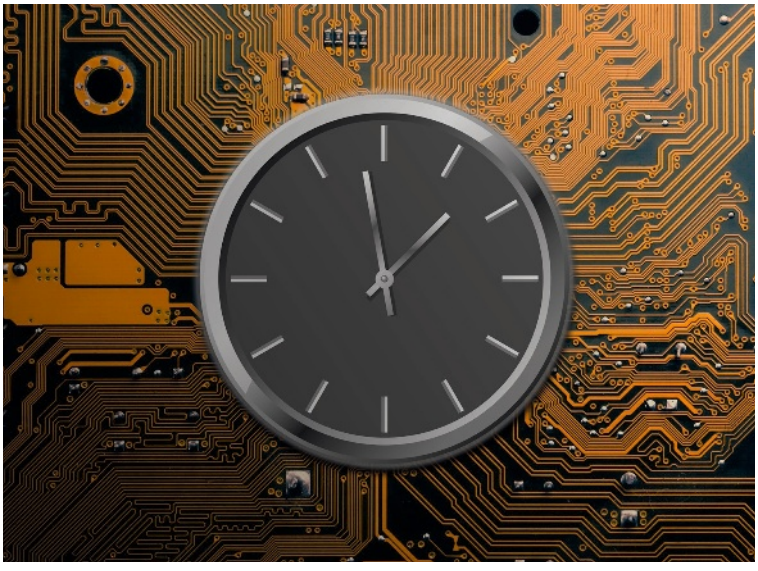
L'erreur ne provient pas de la loi de Moore, mais de la réalisation de puces spécialisées dans le calcul de carrés et des progrès faits par les algorithmes de calcul du carré d'un nombre entier. Les informaticiens et les mathématiciens travaillent en effet à des avancées en termes d'algorithmes ou dans la conception des dispositifs de calcul qui accélèrent la résolution de certains problèmes particuliers indépendamment des progrès de l'électronique de base, lesquels sont les seuls que la loi de Moore prend en compte.

Le calcul de carrés successifs était pourtant « intrinsèquement séquentiel », ce qui signifie que l'utilisation de calculs parallèles ne pouvait pas aider à accélérer sa réalisation. Aujourd'hui, d'autres méthodes – les VDF ou fonctions vérifiables à délai – ont été proposées pour concevoir des énigmes dont la résolution exige une quantité de calcul impossible à mener en parallèle et donc demandent un temps de calcul impossible à raccourcir sensiblement. Cependant, la durée minimale que de tels calculs exigent reste imprécise. >

4 UN DÉFI DE CALCUL TROP VITE RÉSOLU

Une idée pour concevoir des capsules temporelles est de chiffrer le message avec un procédé dont la clé de déchiffrement ne se découvre qu'en menant un long calcul dont la durée amènera à la date fixée pour la libération du message. Proposée par Ronald Rivest, Adi Shamir et David Wagner en 1996, cette idée ne permet pas de fixer avec une grande précision la date de libération du message, car selon qu'on aura consacré beaucoup de moyens ou non pour mener le calcul, on arrivera plus ou moins vite au résultat provoquant l'ouverture.

Les calculs nécessairement longs proposés par les chercheurs étaient des calculs enchaînés de carrés de nombres entiers. Un problème de ce type fut proposé en 1999 à titre de défi (https://en.wikipedia.org/wiki/LCS35) ; il devait tenir trente-cinq ans, mais il a été résolu en moitié moins de temps car on a su améliorer les algorithmes de calcul de carrés et créer des processeurs spécialisés (voir « Programmers solve MIT's 20-year-old cryptographic puzzle », laboratoire CSAIL du MIT, avril 2019 : https://www.csail.mit.edu/news/programmers-solve-mits-20-year-old-cryptographic-puzzle).



> Pour atteindre l'objectif recherché, on peut s'appuyer sur les techniques d'« offuscation » de programmes et sur l'existence d'horloges publiques sécurisées ou aux propriétés particulières.

DES PROGRAMMES OFFUSQUÉS QUI SURVEILLENT L'HEURE

L'offuscation de programmes consiste à écrire des programmes informatiques que tout le monde peut avoir et faire fonctionner, mais dont la logique de fonctionnement est tellement embrouillée et masquée que même en étudiant soigneusement le programme, on ne saura pas bien ce qu'il fait. Cela empêche qu'on les modifie ou qu'on exploite les idées qui ont conduit à leur mise au point.

Si l'on maîtrise l'offuscation, on peut créer un programme à retardement livrant un message à une date fixée par le créateur: une fois lancé sur une machine ayant accès à internet, le programme offusqué lit l'heure sur un ou plusieurs sites qui la publient en continu. Quand l'heure trouvée dépasse la date programmée, le programme déchiffre l'information qui est cachée en lui et la fait alors apparaître en clair. Si l'information qu'on veut libérer est très volumineuse, l'information libérée sera la clé de déchiffrement du fichier chiffré contenant cette information, le programme offusqué ne servant qu'à cacher la clé de déchiffrement.

Il existe de nombreux sites publiant l'heure, en particulier l'horloge parlante (http://www.horlogeparlante.com). On peut imaginer qu'ils fonctionneront encore longtemps et suivront le même mode de publication. Si l'on craint qu'ils soient compromis ou simplement arrêtés, l'utilisation de plusieurs horloges dispersées en des endroits divers du monde fournira une garantie renforcée. Le programme consultera par exemple n horloges et considérera comme certain que le moment attendu est atteint seulement si k horloges parmi n l'indiquent.

Cette méthode est moins risquée que celle où l'on confie le message à un tiers de confiance: les horloges utilisées ne sont pas informées du message caché dans le programme offusqué, et ne savent même pas qu'on les utilise. Toutefois, certains résultats théoriques semblent indiquer qu'aucune méthode parfaite d'offuscation n'existe. D'autres résultats récents utilisant une définition moins stricte de l'offuscation montrent qu'on peut réussir, mais au prix d'un ralentissement du fonctionnement du programme offusqué.

C'est pourquoi on a recherché des méthodes ne faisant pas appel à l'offuscation. L'une d'elles, décrite en 2004 par Aldar Chan et Ian Blake, et perfectionnée en 2006 par Julien Cathalo, Benoît Libert et Jean-Jacques Quisquater, décrit comment une horloge de référence fiable peut produire et publier des informations qui permettront à quelqu'un voulant émettre un message à retardement de le faire sans que l'horloge ait à connaître le message temporairement chiffré ni même savoir qu'un tel message existe.

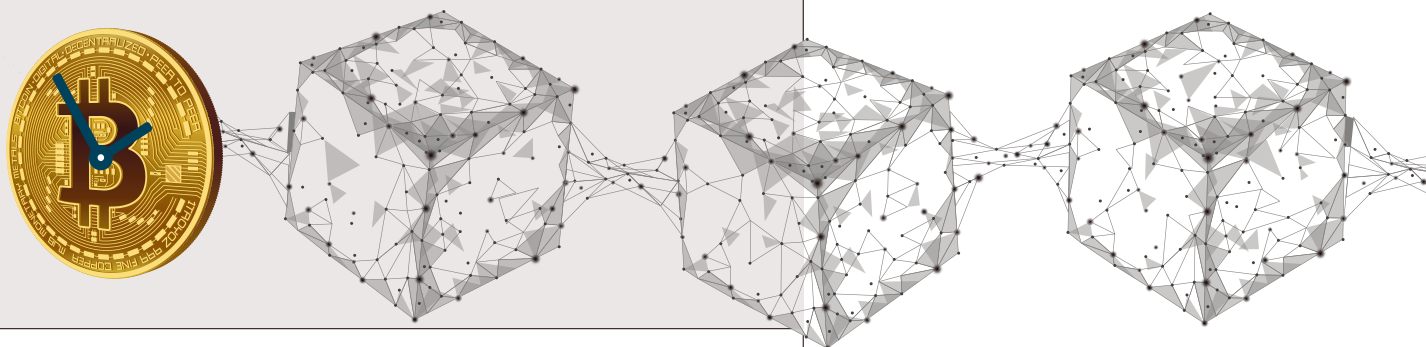
L'horloge produit à la date voulue des informations permettant de déchiffrer les fichiers secrets. Ni le créateur de la capsule temporelle ni ceux à qui elle est adressée n'ont à interagir avec l'horloge qui diffuse les messages permettant de déchiffrer les capsules temporelles ayant atteint l'instant de libération. Ces méthodes permettent aussi de préserver l'anonymat des utilisateurs qui créent les capsules temporelles ou les lisent l'heure venue.

Malheureusement, celui qui fait fonctionner horloge peut produire avant l'heure les clés de déchiffrement des capsules temporelles. La solution est donc encore imparfaite.

5 LA SOLUTION BLOCKCHAIN

La technologie des blockchains qui est utilisée pour émettre et faire circuler des cryptomonnaies telles que le Bitcoin met à la disposition de tous une horloge décentralisée dont le fonctionnement ne peut être perturbé par personne.

Cette technologie permet de concevoir des capsules temporelles dont la sûreté n'est pas liée à la confiance accordée à un acteur particulier. Cette dernière avancée dans la conception de capsules temporelles de plus en plus proches de l'idéal, décrit dans l'encadré 2, est due à Jia Liu, de l'université de Surrey, Tibor Jager et Saqib Kakvi, de l'université de Paderborn, et Bogdan Warinschi, de l'université de Bristol, et a été publiée en 2018 *(voir la bibliographie)*.



© Shutterstock.com/Iurii Motov/LuckyStep