Avec ce procédé, vous pouvez faire exécuter par l'opérateur extérieur d'autres types de calculs. Pour connaître le nombre de données qui dépassent un certain nombre B, vous lui demanderez le nombre de données qui dépassent $C\,B$. Le tri de vos données par ordre croissant est aussi faisable, puisque si C est positif, le tri de $C\,d_1, C\,d_2, \ldots, C\,d_n$ exige les mêmes permutations que $d_1, d_2, \ldots, d_n$. Toutes les moyennes pondérées, par exemple la moyenne $(d_1 + 2d_2 + 5d_3 + d_4)/9$, sont aussi possibles sans dévoiler C, car le résultat sur les données multipliées par C est égal à C fois le résultat attendu. Certaines opérations un peu plus compliquées sont envisageables : pour le calcul de $d_1d_2 + d_3d_4 + d_5d_6$, il suffira de diviser le résultat que $\mathbb{O}$ fournit par C^2.

Cependant, vous ne pourrez pas demander $d_1 + d_2d_3$ et, finalement, ce n'est qu'un faible sous-ensemble de calculs que vous saurez faire exécuter à l'extérieur sans avoir à divulguer vos secrets. En particulier, pas question de faire opérer des algorithmes complexes sur vos données, comme l'identification, parmi les nombres confiés, de ceux qui sont des nombres premiers. Remarquons aussi que le chiffrement de vos données en les multipliant par une constante C laisse transparaître certaines informations que vous ne souhaitez peut-être pas que l'opérateur $\mathbb{O}$ connaisse : le numéro de la donnée la plus grande ou la plus petite, les valeurs égales, les données classées, etc.

Le problème sera résolu si vous disposez d'une méthode de chiffrement permettant à la fois de bien masquer toute l'information et autorisant l'exécution d'additions et de multiplications sans rien révéler : avec ces deux opérations, on reconstitue tout calcul.

La solution : des chiffrements pleinement homomorphes

Notons aussi qu'il n'est pas essentiel que l'opération effectuée par $\mathbb{O}$ soit identique à celle que vous voulez au final. En effet, supposons que le chiffrement de la donnée d soit $\mathscr{C}(d)$ et que le déchiffrement (que vous seul savez effectuer) soit l'opération $\mathscr{C}'$: $\mathscr{C}'(\mathscr{C}(d)) = d$. Pour réussir à faire exécuter toutes sortes de calculs par un opérateur extérieur, il suffirait qu'il existe une opération $*_1$ et une opération $*_2$ (pas nécessairement l'addition et la multiplication habituelles) vérifiant :

$\mathscr{C}(d_1 + d_2) = \mathscr{C}(d_1) *_1 \mathscr{C}(d_2)$ (propriété 1)

$\mathscr{C}(d_1 \times d_2) = \mathscr{C}(d_1) *_2 \mathscr{C}(d_2)$ (propriété 2).

En exécutant $*_1$ et $*_2$ à la place de $+$ et $\times$, l'opérateur $\mathbb{O}$ produira à partir des données chiffrées un résultat qui, après le déchiffrement par $\mathscr{C}'$, sera exactement ce que vous voulez.

Par exemple, si vous souhaitez connaître $d_1 + d_2d_3$ sans le calculer vous-même (ce qui est impossible avec la méthode de multiplication par C), vous enverrez vos données cryptées $\mathscr{C}(d_1)$, $\mathscr{C}(d_2)$, $\mathscr{C}(d_3)$ à l'opérateur en lui demandant de calculer $\mathscr{C}(d_1) *_1 (\mathscr{C}(d_2) *_2 \mathscr{C}(d_3))$ et de vous renvoyer le résultat R. Grâce à l'opération $\mathscr{C}'$, vous obtiendrez alors la valeur de $d_1 + d_2d_3$, puisque :

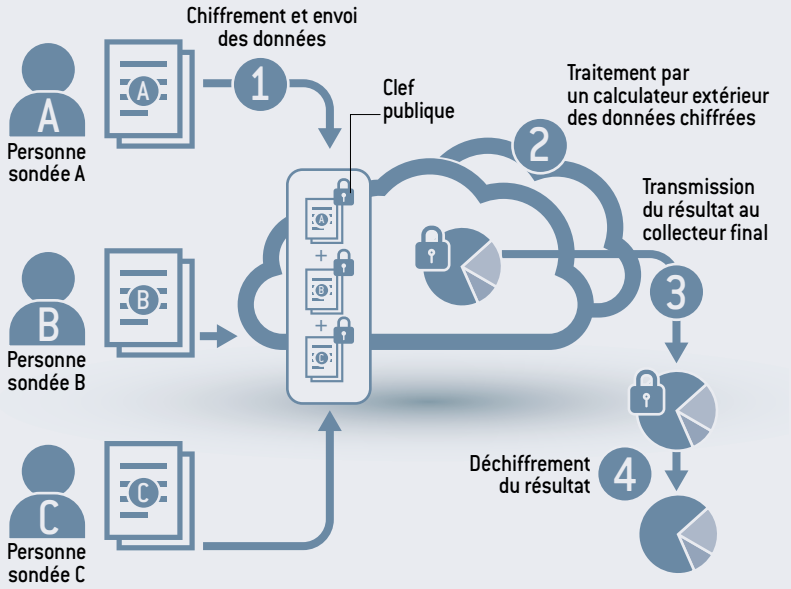
$\mathscr{C}'(R) = \mathscr{C}'[\mathscr{C}(d_1) *_1 (\mathscr{C}(d_2) *_2 \mathscr{C}(d_3))] = \mathscr{C}'(\mathscr{C}(d_1)) + \mathscr{C}'(\mathscr{C}(d_2) *_2 \mathscr{C}(d_3)) = d_1 + \mathscr{C}'(\mathscr{C}(d_2) \times \mathscr{C}'(\mathscr{C}(d_3))) = d_1 + d_2d_3$.

La méthode fonctionnera pour toute expression, aussi compliquée soit-elle, constituée d'additions et de multiplications. Puisque dès qu'on sait mener des additions

Préserver la confidentialité

Imaginons que l'on souhaite établir une statistique sur un sujet délicat en utilisant Internet et en garantissant à chacune des personnes sondées que les données individuelles contenues dans les réponses ne seront pas divulguées. La procédure est la suivante :

Chaque personne chiffre ses réponses avec la clef publique du collecteur final et les envoie à un calculateur extérieur *(1)*. Ce calculateur traite ces données cryptées et envoie le résultat au collecteur final *(2, 3)*. Ce dernier déchiffre le résultat et obtient la statistique souhaitée *(4)*. Le calculateur extérieur ne peut pas accéder aux informations privées des sujets de l'enquête car il ne dispose pas de la clef privée de déchiffrement, et le collecteur final ne peut pas accéder aux données individuelles car il n'a pas eu accès aux informations chiffrées individuelles transmises.



Craig Gentry, le pionnier

En proposant en 2009 le premier système de cryptage complètement homomorphe, Craig Gentry, aujourd'hui chercheur chez IBM, a révolutionné le domaine. Il a déclenché une multitude de travaux qui, petit à petit, ouvrent des voies pour rendre utilisable son idée, initialement théorique pour l'essentiel. Confier de façon massive des calculs à des opérateurs extérieurs qui ne peuvent pas accéder aux données n'est pas aujourd'hui envisageable. En revanche, peu à peu, il est devenu possible de résoudre des problèmes particuliers n'exigeant qu'une petite quantité de calculs homomorphes (comme pour un vote). Les applications de ces méthodes devraient rapidement s'étendre et aider à accroître la sécurité des réseaux et la protection des données... ce dont on a un grand besoin !

© John D. & Catherine T. MacArthur Foundation

et des multiplications de cette façon, on peut exécuter tout algorithme, le problème des calculs secrets par un tiers se ramène à celui de trouver $\mathscr{C}$, $\mathscr{C}'$, $*_1$, $*_2$, vérifiant les propriétés 1 et 2.

Les méthodes de cryptage vérifiant les propriétés 1 et 2 sont dites « pleinement homomorphes ». Le terme vient du langage mathématique où l'on dénomme homomorphisme les fonctions qui transforment une opération en une autre, comme l'indique par exemple la propriété 1.

Ces homomorphismes jouent un rôle très important en mathématiques et chacun connaît le logarithme qui vérifie $\log(a \times b) = \log(a) + \log(b)$ et qui transforme donc des multiplications en additions (ici, $*_2$ est l'addition habituelle). Avant l'avènement des ordinateurs, c'était le moyen le plus efficace pour mener à la main des calculs numériques, et c'est pourquoi on dressait des tables de logarithmes contenant des millions de données.

En mathématiques, on connaît de nombreux homomorphismes, mais aucun des homomorphismes classiques ne peut servir à concevoir un cryptage $\mathscr{C}$ homomorphe car l'opération de décryptage $\mathscr{C}'$ est alors trop simple ; or une bonne méthode cryptographique est une méthode où il est difficile de trouver $\mathscr{C}'$. Le problème des systèmes de chiffrement pleinement homomorphes n'était donc pas évident au moment où, en 1978, l'idée en a été envisagée par les Américains Ron Rivest et Leonard Adleman et le Grec Michael Dertouzos. On s'est longtemps demandé si de telles méthodes existaient. Le problème a parfois été qualifié de graal de la cryptographie.

Une percée décisive a été réalisée en 2009 par Craig Gentry, de l'université Stanford et aujourd'hui chercheur au centre Thomas Watson d'IBM à New York. Craig Gentry a proposé une méthode qui résout théoriquement le problème et a pour cela reçu une bourse de 625 000 dollars de la fondation MacArthur. La méthode qu'il a initialement proposée n'est pas applicable, car elle exige des calculs trop volumineux au moment du cryptage (la transformation des données avant de les envoyer) et du décryptage. De plus, la taille des données que doit manipuler l'opérateur extérieur $\mathbb{O}$, et donc la quantité de calculs qu'il doit mener, est des millions de fois celle nécessaire quand les données ne sont pas cryptées.

Craig Gentry a fait une percée décisive, pour laquelle il s'est vu attribuer une bourse de 625 000 dollars.

Depuis 2009, une multitude de recherches ont été faites pour trouver des schémas plus économiques. En quelques années, les progrès ont rendu envisageables certaines applications, et l'espoir de méthodes encore plus économiques et faciles à mettre en œuvre suscite de nombreux travaux parmi les spécialistes.

Avant de donner des précisions sur les idées de la percée décisive de Craig Gentry, revenons sur les applications des cryptages pleinement homomorphes ou même partiellement homomorphes.

L'algorithme RSA (conçu en 1977 par Ronald Rivest, Adi Shamir et Leonard Adleman) est une méthode de chiffrement à clef publique : pour crypter un message, on utilise la clef publique de son correspondant, et lui seul peut déchiffrer le message grâce à sa clef secrète. C'est un bon algorithme, aujourd'hui très utilisé. Il a la propriété d'être homomorphe pour la multiplication : on en déduit une méthode de chiffrement homomorphe pour l'addition, comme celle évoquée au début de l'article, méthode ayant cette fois l'avantage que les données sont correctement masquées à l'opérateur extérieur.

Une application : les votes secrets

Avec un système additivement homomorphe de ce type, on peut opérer des votes secrets sur le réseau sans se rencontrer. Décrivons un tel procédé.

Il y a trois types d'intervenants : les votants (au plus $N-1$), l'opérateur calculateur, et l'opérateur de dépouillement. L'opérateur de dépouillement choisit un couple clef publique/clef privée et communique la clef publique à chaque votant. Chaque votant choisit alors un nombre entier V de la forme Nk (k entier) s'il veut voter OUI, ou $Nk+1$ (k entier) s'il veut voter NON ; puis il chiffre le nombre V choisi avec la clef publique et transmet le nombre crypté $\mathscr{C}(V)$ à l'opérateur de calcul. Ce dernier fait la somme S de tous les nombres qu'il reçoit et communique S à l'opérateur de dépouillement, lequel déchiffre le résultat grâce à la clef privée, qu'il est seul à connaître. Il trouve ainsi un nombre de la forme $Mk+p$, où p est le nombre de personnes ayant voté NON. Le résultat du vote est alors communiqué à tous.