En effet, si le bruit devient supérieur à p, le reste n'est plus comme il faut (car le quotient augmente de 1 ou plus) pour connaître la somme et le produit des deux bits codés. La méthode n'est bonne qu'à la condition d'opérer un nombre limité d'opérations successives. Pour un calcul dépendant de plusieurs dizaines de bits $b_1, b_2, ..., b_k$ présents dans une expression complexe utilisant beaucoup de XOR et de ET, le résultat obtenu par l'opérateur extérieur risque d'être devenu illisible à celui qui lui a confié le calcul. La méthode sans le perfectionnement qui va venir ne fonctionne qu'avec des calculs assez limités. C'est là qu'intervient la deuxième idée majeure de Craig Gentry. Elle consiste à nettoyer périodiquement le bruit pour éviter qu'il ne devienne trop important.

Pour cela, on fait déchiffrer le résultat du calcul partiel avant qu'il ne soit trop brouillé. Ce nettoyage enlève tout le bruit. On l'effectue en demandant à l'opérateur extérieur d'appliquer la fonction de déchiffrement sur le résultat partiel, c'est-à-dire de diviser par p et examiner la parité du reste obtenu. Suprême astuce : puisqu'on ne fait pas confiance à l'opérateur extérieur, on ne lui communique pas la clef de déchiffrement (le nombre p), mais on lui demande de faire ce déchiffrement par un calcul homomorphe utilisant la méthode qu'on vient de décrire ! Ce n'est possible bien sûr que si ce calcul homomorphe du déchiffrement n'est pas trop complexe, c'est-à-dire n'exige pas, une fois traduit en XOR et en ET, une succession trop longue de calculs ; on s'en assurera.

Cela semble assez fou et cela ressemble à l'opération consistant à s'élever dans les airs en tirant sur ses bottes… c'est d'ailleurs pourquoi le nom donné à cette phase du calcul est *bootstrapping*. Pour que la phase de *bootstrapping* soit possible, il faut qu'un minimum d'opérations soient faisables sans que le bruit qui s'introduit soit devenu trop grand, et cela oblige à prendre des nombres entiers très longs pour p et les q_i, mais cela marche, et on dispose donc d'une méthode de calcul pleinement homomorphe comme on en rêvait. L'opération de *bootstrapping*, en nettoyant autant de fois que nécessaire les calculs partiels, rend possible des séries d'opérations aussi longues qu'on le souhaite de XOR et de ET ; et on peut donc faire exécuter tout calcul par un opérateur extérieur qui ignorera ce qu'il calcule.

Puisque la taille des bits chiffrés $\mathcal{C}(b)$ est nécessairement grande, les calculs effectués par l'opérateur extérieur sont très lourds, énormes comparés à ce qu'il faudrait faire si on ne souhaitait rien cacher, mais le but est atteint : on sait déléguer des calculs sans dévoiler ni les données ni le résultat.

Encore quelques ordres de grandeur à gagner

Les travaux très nombreux sur ce sujet menés depuis 2009 ont visé à réduire les surcoûts en taille et en lourdeur de calcul des systèmes homomorphes. On progresse et des réalisations logicielles utilisant les idées mathématiques de ces travaux tentent de rendre les systèmes utilisables en pratique. La série d'opérations nécessaires au chiffrement de données par l'algorithme AES (*Advanced Encryption Standard*), très largement utilisé dans le monde entier, sert de repère pour mesurer les progrès des différents programmes de chiffrement homomorphe. On est ainsi passé, pour l'exécution de ce calcul test par un système homomorphe, de plusieurs heures (sur une machine courante) à quelques secondes.

C'est remarquable, mais pour que réellement on puisse envisager de confier massivement les calculs qu'on veut faire sur des données confidentielles à des opérateurs extérieurs qui les exécuteraient sans rien y voir, il faudra encore attendre un long moment, que le surcoût ait diminué de plusieurs ordres de grandeur.

Certains experts tels que l'Américain Bruce Schneier restent réservés tant les surcoûts sont encore élevés. Aujourd'hui, la formidable idée de la cryptographie homomorphe est utile, par exemple pour la conception de systèmes de vote. Cependant, malgré une série d'idées intéressantes qui ont prouvé qu'elle était possible dans sa forme la plus générale, sa mise en œuvre pour tous, permettant plus de sécurité et une meilleure confidentialité sur les réseaux, devra encore attendre un peu. ■

L'AUTEUR

J.-P. DELAHAYE est professeur émérite à l'Université de Lille et chercheur au Centre de recherche en informatique, signal et automatique de Lille (CRISTAL).

BIBLIOGRAPHIE

X. Yi *et al.*, **Homomorphic Encryption and Applications**, Springer, 2014.

V. Cortier et S. Kremer, **Vote par Internet**, *Interstices*, 2013 (https://interstices.info/jcms/int_68258/vote-par-internet).

M. Van Dijk *et al.*, **Fully homomorphic encryption over the integers**, *Advances in cryptology - EUROCRYPT 2010*, Springer, pp. 24-43, 2010.

C. Gentry, **Fully homomorphic encryption using ideal lattices**, *Proceedings of the 41st ACM Symposium on Theory of Computing*, pp. 169-178, 2009.

R. Rivest *et al.*, **On data banks and privacy homomorphisms**, *Foundations of Secure Computation*, Academic Press, pp. 169-180, 1978.

Retrouvez la rubrique Logique & calcul sur www.pourlascience.fr