ignorant p ne peut pas faire la différence entre $qp+e$ et un nombre quelconque de même taille.

De ce fait, chiffrer un bit b (b=0 ou 1) par $\mathcal{C}(b)=pq+b+2r$ où r a une vingtaine de chiffres et q plusieurs millions est une bonne méthode : celui qui connaît p n'a aucun mal à retrouver q en faisant la division par p.

Le reste de la division est $b+2r$, qui est pair si b=0 et impair si b=1. Ainsi, celui qui connaît p saura sans mal déchiffrer $\mathcal{C}(b)$. Et celui qui ne connaît pas p ne voit dans $\mathcal{C}(b)$ qu'un nombre quelconque dont il ne tire rien. Ce système de chiffrement est homomorphe pour le XOR et le ET et est donc pleinement homomorphe. Le vérifier est une simple question d'arithmétique. En voici les détails, que vous pouvez passer si vous trouvez cela fastidieux.

Soient b_1 et b_2 deux bits. Chiffrons-les : $\mathcal{C}(b_1)=c_1=q_1p+2r_1+b_1$ et $\mathcal{C}(b_2)=c_2=q_2p+2r_2+b_2$ avec des entiers r_1 et r_2 d'une vingtaine de chiffres, p d'environ 500 chiffres et q_1 et q_2 de plusieurs millions de chiffres. Celui qui connaît p peut calculer le reste de la division par p de : $c_1+c_2=(q_1+q_2)p+2(r_1+r_2)+(b_1+b_2)$. Ce reste est $2(r_1+r_2)+(b_1+b_2)$; puisque $2(r_1+r_2)$ est pair, connaître le reste donne la parité de (b_1+b_2) c'est-à-dire le XOR entre b_1 et b_2. Additionner $\mathcal{C}(b_1)$ et $\mathcal{C}(b_2)$ donne donc, si l'on connaît p, le XOR entre b_1 et b_2 (b_1+b_2 n'est impair que si l'un ou l'autre est impair, mais pas les deux, et est pair dans les deux autres cas). On peut donc faire calculer à un opérateur extérieur des XOR sans qu'il connaisse ni les données b_1 et b_2, ni le résultat qu'il calcule. C'est la propriété d'homomorphisme additif. Notons que le «bruit» sur c_1+c_2 est passé de r_1 et r_2 à $2(r_1+r_2)$, il a donc à peu près doublé.

Pour la multiplication, l'opérateur extérieur calcule : $c_1c_2=(q_1q_2p+2q_1r_2+q_1b_2+2q_2r_1+q_2b_1)p+2(2r_1r_2+r_1b_2+r_2b_1)+b_1b_2$.

La division par p donne le reste $2(2r_1r_2+r_1b_2+r_2b_1)+b_1b_2$ et donc la parité de b_1b_2, c'est-à-dire le ET entre b_1 et b_2 (b_1b_2 est impair si b_1 et b_2 le sont). C'est la propriété d'homomorphisme multiplicatif. Notons cependant que le bruit sur le résultat est maintenant de l'ordre de $4r_1r_2$.

NETTOYER LE « BRUIT »

Cette méthode est donc parfaite à un détail près. Plus on fera de calculs successifs de XOR et de ET, plus le bruit introduit par les $2r$ augmentera en taille et cela en particulier à cause des ET. Ce n'est pas grave si l'on n'effectue que quelques opérations, mais cela devient très ennuyeux si le bruit atteint la taille de p, car alors on ne peut plus retrouver (b_1+b_2) et b_1b_2.

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 >

VERS DES MACHINES À VOTER SÛRES ?

La question de savoir s'il faut développer l'utilisation de machines à voter est particulièrement délicate, et elle exige que des cryptologues compétents et des techniciens très qualifiés s'en occupent… ce qui n'est pas toujours le cas. Des appels à la prudence ont été émis par de nombreuses personnalités et mouvements. En France, un rapport du Sénat datant de 2014 préconisait le maintien du moratoire sur les machines à voter décidé en 2007. La Cnil a aussi exprimé des réserves sur le vote électronique.

Les incidents et problèmes semblent trop nombreux pour qu'on envisage aujourd'hui de remplacer au niveau national le bulletin de vote, l'isoloir et l'urne fermée à clé par des ordinateurs s'échangeant des messages et que les citoyens devraient manipuler pour effectuer leurs devoirs électoraux.

Le site Ordinateurs de vote (www.ordinateurs-de-vote.org/) et le récent rapport *Ethics and electronic voting* de Chantal Enguehard (https://hal.archives-ouvertes.fr/hal-01016256/document) détaillent les raisons et les faits montrant qu'il faut être prudent et patient.

Cependant, rien n'interdit de chercher à concevoir des méthodes de vote aussi rigoureuses que possible, permettant des contrôles *a posteriori* tout en garantissant l'anonymat des votes. Le calcul homomorphe présenté dans le présent article semble une voie sérieuse : il donne des garanties convenables concernant la protection du secret des votes et autorise le contrôle et le recalcul des résultats, tout en permettant à chacun d'être certain que son bulletin est pris en compte.

Le système de vote par Internet Helios (https ://vote.heliosvoting.org/), fondé sur le calcul homomorphe et développé par l'université Harvard et l'université catholique de Louvain, est disponible gratuitement et librement. L'association internationale des chercheurs en cryptographie (IACR) l'utilise pour choisir les membres de son bureau… ce qui est un argument sérieux pour croire en ses qualités.

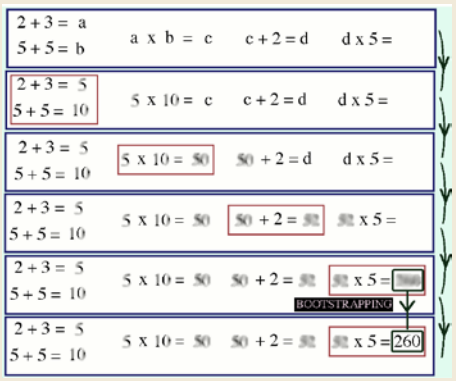
Il faut se méfier de toutes les solutions précipitées en la matière, et il ne faut renoncer à aucune des garanties données par les procédures classiques de vote. Il ne fait cependant pas de doute qu'il y aurait des avantages à pouvoir voter de chez soi, et que si c'était possible de manière sûre, le coût de l'organisation d'élections en serait, à terme, considérablement réduit.

Machine à voter électronique utilisée à Aulnay-sous-Bois en 2007.

DR

CALCUL FLOU ET *BOOTSTRAPPING*

Le système de calcul homomorphe initialement conçu par Craig Gentry se fonde sur deux idées que l'on retrouve dans tous les systèmes de calcul homomorphe inventés et perfectionnés depuis.
La première idée est que les données doivent être chiffrées d'une manière floue, ce flou ne pouvant être effacé que par celui qui connaît la clé de déchiffrement. À mesure que les calculs homomorphes s'effectuent, le flou s'accroît – comme l'illustration ci-contre le symbolise. Après un certain nombre d'opérations, le flou devient si important que même avec la clé de déchiffrement, il est impossible d'accéder aux résultats.
La seconde idée est qu'en s'y prenant bien (en demandant au système homomorphe d'exécuter l'opération de déchiffrement sur ses données, donc sans en prendre connaissance, et sans prendre connaissance des résultats), on réussit à effacer le flou qui risquait de devenir irrécupérable. C'est le *bootstrapping*, opération miraculeuse analogue à celle du baron de Münchhausen qui s'élevait dans les airs en tirant sur ses bottes.
Soigneusement conçu et réalisé, ce nettoyage périodique des données est réellement possible et permet alors de mener des calculs homomorphes sans limitations de complexité : dès que le flou devient trop grand, on le réduit par *bootstrapping*.



> 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 clé 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. ■

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.