RFC 6090 : Fundamental Elliptic Curve Cryptography Algorithms

Stéphane Bortzmeyer
<stephane+blog@bortzmeyer.org>

Première rédaction de cet article le 4 février 2011

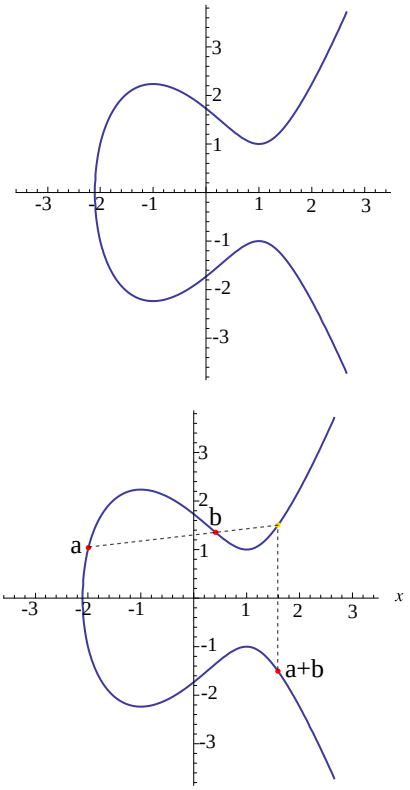
Date de publication du RFC : Février 2011

https://www.bortzmeyer.org/6090.html

Dans l'arsenal des algorithmes de cryptographie utilisés sur l'Internet, on trouve de plus en plus d'algorithmes basées sur les courbes elliptiques (par exemple les RFC 4754 [1], RFC 5289, RFC 5656 et RFC 5753, sans compter les normes d'autres SDO). Mais il n'existait apparemment pas de texte de synthèse sur cette famille d'algorithmes, utilisable comme référence pour des RFC. Ce manque est désormais comblé avec ce RFC 6090 qui résume ce que tout participant à l'IETF devrait connaître des courbes elliptiques et de leur utilisation en cryptographie. Ainsi, RSA et la factorisation en nombres premiers ne restera pas la seule méthode de cryptographie asymétrique sur laquelle faire reposer la sécurité des communications à travers l'Internet.

Ce RFC nécessite donc un peu plus de connaissances en mathématiques que la moyenne. Le débutant pourra commencer par l'excellent article d'Adam Langley, sur son blog Imperial Violet, « "*Elliptic curves and their implementation*" <https://www.imperialviolet.org/2010/12/04/ecc.html> ». Il apprendra que les courbes elliptiques n'ont rien à voir avec les ellipses (en mathématiques, lorsqu'on classe des objets en deux types on appelle souvent l'un le type pair, le type impair ; lorsqu'on tombe sur trois types, on qualifie souvent les trois types de parabolique, elliptique et hyperbolique, en référence aux coniques), il verra de jolies images : (toutes les images de cet article ont été volées à l'article sur Imperial Violet). Il devra aussi réviser quelques notions de maths (cf. section 2 du RFC) comme celle de groupe (section 2.2 du RFC). En effet, les points de la courbe elliptique, plus une opération d'addition sur la courbe (qu'on peut aussi représenter visuellement, voir l'image plus loin), qui a un élément neutre (un zéro), forment un groupe. (À noter que le RFC, lui, sans donner de nom à cette opération, la note * qui est d'habitude réservé, en informatique, à la multiplication, voir annexe E pour une discussion de ces deux notations possibles.)

1. Pour voir le RFC de numéro NNN, https://www.ietf.org/rfc/rfcNNN.txt, par exemple https://www.ietf.org/rfc/rfc4754.txt



Une fois qu'on a l'addition, on définit la multiplication par un nombre entier positif (qui consiste juste à répéter l'addition). Pour tout point de la courbe, on peut donc relativement facilement le multiplier par un nombre quelconque (ce nombre sera la **clé** privée). Mais l'inverse est extrêmement dur : étant donné un point résultat, et le point de départ, comment trouver le nombre qui a servi à la multiplication ? Dans le groupe formé par les nombres entiers et l'addition usuelle, c'est trivial (c'est une simple division euclidienne). Sur la courbe elliptique, c'est au contraire une tâche herculéenne. Et c'est là son intérêt pour la cryptographie. De méme que RSA reposait sur le fait qu'une composition des facteurs premiers est triviale mais que la décomposition est **très** difficile, la cryptographie par courbes elliptiques repose sur le fait que la multiplication sur la courbe est simple, la division très difficile. On a donc la base de la cryptographie asymétrique, une opération mathématique très difficile à inverser.

J'ai dit que la multiplication sur la courbe elliptique était simple mais tout est relatif. L'essentiel de l'article d'Imperial Violet est consacré à la mise en œuvre logicielle et beaucoup d'optimisations sont nécessaires, pour pouvoir multiplier un point en un temps raisonnable, sachant que le facteur de multiplication est un très grand nombre. L'article devient nettement plus chevelu ici, à réserver aux fanas de l'algorithmique.

Après ce petit détour par les maths, revenons au RFC. Il utilise le sigle **ECC** pour parler de l'ensemble des techniques de "*Elliptic Curve Cryptography*". Ce sont donc des techniques de clé publique, qui peuvent servir à mettre en œuvre des techniques comme Diffie-Hellman ou ElGamal. Bien qu'assez ancienne, ECC est toujours relativement peu déployé, alors qu'elle fournit en général une meilleure sécurité et des performances plus importantes. Notre RFC note que c'est sans doute en partie par manque de documents normatifs facilement disponibles (un problème que notre RFC vise justement à traiter) mais aussi en raison de problèmes d'appropriation intellectuelle (la plupart des techniques ECC sont pourries de brevets jusqu'au trognon, cf. section 9).

Le RFC fournit d'abord un arrière-plan mathématique nécessaire, en section 2. Comme il n'est pas facile de lire des formules mathématiques dans le texte ASCII auquel sont contraints les RFC `<https://`

www.bortzmeyer.org/rfc-en-texte-brut.html>, la lecture n'en est pas évidente et je suggère de partir plutôt de l'article d'Imperial Violet.

Le RFC donne ensuite la définition d'une courbe elliptique en section 3 et introduit la notion de transformation d'un système de coordonnées dans un autre, afin de faciliter les calculs (cf. annexe F pour du pseudo-code). Surtout, la section 3.3 liste les **paramètres** dont dépend une courbe elliptique particulière. Des propriétés de ces paramètres dépend la sécurité de la courbe et ils ne doivent donc pas être choisis au hasard (la section 3.3.2 donne les critères qu'ils doivent satisfaire). Un exemple d'une courbe et de ses paramètres, la courbe P-256, normalisée par le NIST (dans FIPS 186-2) et utilisée dans le RFC 4753, est donné en annexe D. Une autre courbe elliptique connue est Curve25519, utilisée dans DNScurve <https://www.bortzmeyer.org/dnscurve.html>.

Une fois ECC définie, on peut l'utiliser dans des algorithmes de crypto classique. La section 4 décrit ECDH, courbe elliptique + Diffie-Hellman, pour permettre à ces chers Alice et Bob de se mettre d'accord sur un secret (qui servira au chiffrement ultérieur) bien qu'ils communiquent via un canal non sûr. Le principe est simple mais génial. Alice et Bob partent du même point G de la courbe, Alice choisit au hasard un nombre j et met G à la puissance j (je rappelle que le RFC utilise la notation multiplicative, pas l'additive comme dans l'article sur Imperial Violet, et note donc cette opération G{}j). Bob en fait autant avec son nombre k pris au hasard. Chacun envoie le résultat de son calcul à l'autre, qui doit alors calculer G{}(j*k). Par exemple, Alice reçoit donc G{}k et calcule donc G{}k{}j (qui est équivalent à G{}(k*j)). Alice et Bob connaissent donc le secret, G{}(j*k), alors que l'écoutant éventuel n'a pu le déterminer, car il ignore j et k.

De même, on peut faire une version « courbe elliptique » de ElGamal (section 5). Cet algorithme de chiffrement était basé sur un autre groupe mathématique, celui des entiers modulo N mais on peut se servir d'un autre groupe, comme celui fourni par une courbe elliptique. C'est ainsi qu'est créé ECDSA, version à courbe elliptique de DSA (cet algorithme est disponible pour DNSSEC, cf. RFC 6605 ; il est déjà mis en œuvre dans PowerDNS <http://doc.powerdns.com/dnssec-supported.html> ; DNSSEC a déjà un algorithme à courbe elliptique, GOST, cf. RFC 7091).

L'IETF se préoccupant de faire des protocoles qui marchent dans le monde réel, il faut se poser la question de l'interopérabilité. La section 7 décrit donc des détails pratiques nécessaires. Toujours sur le côté concret, la section 8 est consacrée aux détails d'implémentation et aux tests validant celle-ci. Par exemple, un test courant est de vérifier qu'un sous-programme produit, pour des paramètres d'entrée donnés, le résultat attendu. Comme le mécanisme de signature de KT-I (cf. section 5.4) est non déterministe, ce genre de tests ne va pas marcher, la signature étant différente à chaque fois. Par contre, la validation de la signature est, elle, déterministe (heureusement...) et peut donc être testée. De même, ECDH peut être testé avec les résultats des RFC 4753 et RFC 5114.

Reprenons de la hauteur : si les sections précédentes passionneront les mathématiciens et les programmeurs, la section 9 est consacrée aux problèmes que posent les brevets pour le déploiement des courbes elliptiques. Si les bases des courbes elliptiques semblent libres (en théorie, on ne peut pas breveter un théorème mathématique : mais, en pratique, les organismes comme l'Office Européen des Brevets violent régulièrement leurs propres règles, et semblent agir en dehors de tout contrôle démocratique), les optimisations nécessaires sont souvent plombées par un brevet. L'implémenteur d'une courbe elliptique doit donc, après avoir pris connaissance des règles de l'IETF sur le sujet (RFC 8179), consulter la base des appropriations intellectuelles publiées <https://datatracker.ietf.org/ipr/about/>.

L'unique but de la cryptographie à courbes elliptiques étant la sécurité, la section Sécurité (la 10), revient donc en détail sur les précautions à prendre pour que les courbes elliptiques remplissent leur rôle. Il existe plusieurs attaques documentées (comme celle de Pohlig-Hellman), l'algorithme de Shanks

ou celui de Pollard qui sont des algorithmes génériques, marchant avec n'importe quel groupe, comme des attaques spécifiques d'une courbe elliptique particulière. Le RFC insiste notamment sur l'importance de ne prendre que des courbes dont les paramètres ont été soigneusement choisis (voir aussi le RFC 4086, et l'annexe B, puisque plusieurs opérations dépendent de nombres aléatoires).

D'autre part, la cryptographie ne fait pas de miracles : si Diffie-Hellman protège contre un attaquant passif, il ne peut rien contre un intermédiaire actif.

Merci à Michael Le Barbier Grünewald pour sa relecture mathématique.