RFC 7748 : Elliptic Curves for Security

Stéphane Bortzmeyer
<stephane+blog@bortzmeyer.org>

Première rédaction de cet article le 24 janvier 2016. Dernière mise à jour le 29 janvier 2016

Date de publication du RFC : Janvier 2016

https://www.bortzmeyer.org/7748.html

Ce nouveau RFC est la description de deux courbes elliptiques largement utilisées mais qui n'avaient pas fait l'objet d'une description formelle complète : curve25519 et curve448. Maintenant que c'est fait, elles pourront être utilisées dans des normes Internet utilisant la cryptographie comme TLS ou DNSSEC.

La cryptographie sur les courbes elliptiques est une alternative aux algorithmes traditionnels fondés sur la décomposition en facteurs premiers, comme le classique RSA. Il est toujours bon de ne pas mettre tous ses œufs dans le même panier, car on ne sait jamais quelle sera la prochaine victime de la cryptanalyse (d'où le RFC 7696 [1], sur l'importance de ne pas être prisonnier d'un seul algorithme). Grâce aux courbes elliptiques, si RSA est cassé complètement, on aura une solution de rechange. En outre, la cryptographie sur les courbes elliptiques a des propriétés intéressantes, comme des clés de taille plus petite. Dans les deux cas, RSA ou courbes elliptiques, je n'y connais pas grand'chose donc n'espérez pas des explications plus pointues.

Le schéma général de la cryptographie sur courbes elliptiques est décrit dans le RFC 6090 (voir aussi le cours « "*SEC 1 : Elliptic Curve Cryptography*" <http://www.secg.org/sec1-v2.pdf> »). Ce schéma permet l'utilisation d'une infinité de courbes elliptiques mais attention, toutes n'ont pas les propriétés requises. La plus connue est la courbe P-256 normalisée par le NIST. Comme il y a de très bonnes raisons de se méfier du NIST (leur rôle dans l'affaiblissement délibéré de Dual EC DRBG, cf. l'article de Schneier <https://www.schneier.com/blog/archives/2007/11/the_strange_sto.html> et celui de Greenemeier <http://www.scientificamerican.com/article/nsa-nist-encryption-scandal/>), et comme les raisons des choix des paramètres de P-256 n'ont jamais été rendues publiques, beaucoup de gens se méfient de P-256 (voir une bonne discussion sur StackExchange <https://crypto.

1. Pour voir le RFC de numéro NNN, https://www.ietf.org/rfc/rfcNNN.txt, par exemple https://www.ietf.org/rfc/rfc7696.txt

stackexchange.com/questions/10263/should-we-trust-the-nist-recommended-ecc-parameters>
Il est donc important, là encore, d'avoir plusieurs fers au feu, et de disposer d'autres courbes.

Les deux courbes décrites dans ce RFC sont curve25519 et curve448. Elles sont de la famille des courbes de Montgomery (et donc ont une équivalence avec une courbe d'Edwards).

Il n'est pas facile de faire des maths dans un RFC, avec leur format actuel. La section 3 décrit la notation utilisée dans le RFC, mais je ne vais pas reproduire les équations ici (je suis complètement ignorant de MathML et ce blog ne le gère pas). Des formules comme (section 4.1) :

```
(u, v) = ((1+y)/(1-y), sqrt(-486664)*u/x)
(x, y) = (sqrt(-486664)*u/v, (u-1)/(u+1))
```

sont nettement moins jolies dans le texte brut des RFC.

La section 4 du RFC décrit les deux courbes. Curve25519 doit son nom au fait qu'un des nombres premiers utilisés pour la génération des paramètres de la courbe est 2{}255-19. Sa description originale est dans l'article de Bernstein « "*Curve25519 – new Diffie-Hellman speed records*" <http://www.iacr.org/cryptodb/archive/2006/PKC/3351/3351.pdf> », et son équivalent en courbe d'Edwards, ed25519 dans « "*High-speed high-security signatures*" <http://link.springer.com/chapter/10.1007/978-3-642-23951-9_9> ».

L'autre courbe, curve448, utilise le nombre premier 2{}448-2{}224-1. La courbe d'Edwards équivalente est nommée "*Goldilocks*" et est décrite dans « "*Ed448-Goldilocks, a new elliptic curve*" <http://eprint.iacr.org/2015/625.pdf> ».

La section 5 du RFC présente les fonctions X25519 et X448, utilisées pour faire du Diffie-Hellman avec ces courbes. Des vecteurs de test sont également fournis, si vous voulez vérifier votre implémentation (rappel au passage : il y a **plein** de pièges quand on programme de la crypto.)

En parlant de Diffie-Hellman, la section 6 est la description de l'utilisation de ces deux courbes pour un échange de clés ECDH, utilisant les deux fonctions de la section précédente.

Quelle est la sécurité de ces deux courbes (section 7)? Le niveau de sécurité de curve25519 lors d'une attaque par force brute est d'un peu moins de 128 bits <https://safecurves.cr.yp.to/rho.html>. Une attaque par force brute est normalement l'essai de toutes les clés possibles. Toutefois, le terme est souvent utilisé en cryptographie (avec un abus de langage) pour désigner des méthodes où on fait beaucoup d'essais, sans forcément tester toutes les clés (comme l'algorithme rho). D'autre part, le RFC rappelle qu'il n'est pas facile de comparer des algorithmes de cryptographie, surtout entre algorithmes symétriques et asymétriques, cf. l'article de Bernstein « "*Understanding brute force*" <http://cr.yp.to/snuffle/bruteforce-20050425.pdf> ».)

curve448 est normalement « meilleure », avec 224 bits de sécurité (mais plus lente : on n'a rien sans rien). Le RFC note que, pour des ordinateurs quantiques (si on en a un jour...), les deux courbes seront aussi faciles à casser l'une que l'autre. En attendant, curve25519 est parfaitement suffisant.

Notez que, si vous aimez la cryptographie quantique et les discussions sans fin, la NSA a publié en août 2015 <https://www.nsa.gov/ia/programs/suiteb_cryptography/index.shtml> un texte

qui a stupéfié beaucoup de monde car il affirmait que, vu l'arrivée supposée proche des calculateurs quantiques, il n'était pas recommandé de faire des efforts pour migrer depuis RSA vers les courbes elliptiques. Certains en ont déduit que la NSA avait déjà des calculateurs quantiques et savaient donc que les courbes elliptiques étaient cassées ou proches de l'être, d'autres ont supposé que, si la NSA essayait de décourager les utilisateurs de se servir des courbes elliptiques, c'était au contraire une raison supplémentaire pour migrer vers ces courbes au plus vite. La polémique et l'état des informations disponibles sont très bien expliqués dans l'article « "*A riddle wrapped in an enigma*" <https://eprint.iacr.org/2015/1018.pdf> ».

L'annexe A du RFC décrit comment les paramètres de ces deux courbes ont été générés. Des courbes comme la P-256 du NIST ou comme la courbe elliptique souveraine FRP256 <http://www.legifrance.gouv.fr/affichTexte.do?cidTexte=JORFTEXT000024668816> de l'ANSSI ont été très critiquées car les différentes constantes utilisées sont juste annoncées, sans explication. Il n'y a pas besoin d'être excessivement paranoïaque, ou d'avoir lu les documents publiés par Snowden, pour se demander « Pourquoi ces valeurs plutôt que d'autres ? Ne serait-ce pas parce que ces constantes ont des propriétés qui facilitent leur décryptage par la NSA ou un organisme similaire ? ». Il y a consensus aujourd'hui pour dire qu'on ne doit utiliser que des courbes pour lesquelles le choix des paramètres a été fait « objectivement », selon une méthodologie expliquée et vérifiable. C'est ce que fait l'annexe A.

Notez que les paramètres des deux courbes de ces RFC n'ont pas été choisis complètement « objectivement ». Ils ont fait l'objet d'« optimisations », notamment pour des raisons de performance. Si on veut une courbe générée entièrement par un algorithme objectif et publiquement vérifiable, il faudra se tourner (elle est très récente) vers des courbes « publiquement vérifiables » <https://eprint.iacr.org/2014/130> comme la « courbe d'un million de dollars <https://cryptoexperts.github.io/million-dollar-curve/> », qui proclame être « "*a procedure, more than a curve*" ».

Maintenant que ces courbes font l'objet d'une documentation complète, il sera plus facile de les utiliser depuis un protocole IETF (ceci dit, elles sont déjà dans au moins une norme, voir RFC 7479). Plusieurs travaux sont en cours pour cela ; il y avait un "*Internet-Draft*" pour permettre curve25519 dans TLS, `draft-ietf-tls-curve25519` mais il a été remplacé par le plus général RFC 8422. Un autre "*Internet-Draft*" traite le cas de DNSSEC, `draft-sury-dnskey-ed25519`.

Notez que les deux courbes de ce RFC sont décrites selon un formalisme qui ne permet pas de les utiliser telles quelles dans un algorithme comme ECDSA (explications sur StackOverflow <http://stackoverflow.com/a/2517052/15625>). Mais on peut avec EdDSA.

OpenSSH dispose de curve25519 depuis la version 6.5 <http://www.openssh.com/txt/release-6.5>. (« "*Add support for key exchange using elliptic-curve Diffie Hellman in Daniel Bernstein's Curve25519. This key exchange method is the default when both the client and server support it. [...] Add support for Ed25519 as a public key type. Ed25519 is a elliptic curve signature scheme that offers better security than ECDSA and DSA and good performance. It may be used for both user and host keys.*" »). Dans les sources d'OpenSSH, ce sont les fichiers ayant 25519 dans leur nom :

```
/tmp/openssh-7.1p2 %   ls *25519*
ed25519.c  fe25519.c  fe25519.h  ge25519_base.data  ge25519.c  ge25519.h  kexc25519.c  kexc25519c.c  kexc25519s.
```

Pour générer des clés utilisant cette courbe, on utilise le classique `ssh-keygen` :

```
% ssh-keygen -t ed25519
Generating public/private ed25519 key pair.
...
Your identification has been saved in /home/stephane/.ssh/id_ed25519.
Your public key has been saved in /home/stephane/.ssh/id_ed25519.pub.
...
```

Par contre, ces courbes ne sont pas encore dans OpenSSL (voir la discussion #309 <https://github.com/openssl/openssl/issues/309>). Autrement, on trouve des mises en œuvre de curve25519 à plein d'endroits : <https://github.com/msotoodeh/curve25519>, <https://code.google.com/p/curve25519-donna/> et bien sûr <http://cr.yp.to/ecdh.html> et <https://nacl.cr.yp.to/>.

Merci à Patrick Mevzek pour son exploration d'OpenSSH et OpenSSL, à Émilien Gaspar pour les discussions et à Manuel Pégourié-Gonnard pour des corrections, notamment sur la force brute.