RFC 9861 : KangarooTwelve and TurboSHAKE

Stéphane Bortzmeyer
<stephane+blog@bortzmeyer.org>

Première rédaction de cet article le 13 octobre 2025

Date de publication du RFC : Octobre 2025

https://www.bortzmeyer.org/9861.html

Vous trouvez qu'il n'y a pas assez d'algorithmes en cryptographie ? En voici quatre de plus, ici pour condenser cryptographiquement des données, plus rapidement que les algorithmes existants.

Il s'agit de quatre fonctions, TurboSHAKE128 et TurboSHAKE256 (variantes de TurboSHAKE <http://eprint.iacr.org/2023/342>) ainsi que de KT128 et KT256 (variantes de Kangaroo Twelve qui est décrit dans cet article <https://link.springer.com/chapter/10.1007/978-3-319-93387-0_21>). Toutes sont des XOF ("*eXtendable Output Function*"), des fonctions dont le résultat peut être infiniment long (comme ce que produit un générateur pseudo-aléatoire), mais qu'on tronque à la valeur souhaitée (cf. section 2.1). Kangaroo Twelve (section 3) est bâti sur TurboSHAKE et apporte le parallélisme : il permet de répartir le travail sur plusieurs processeurs alors que SHA-3 et TurboSHAKE sont strictement séquentiels. TurboSHAKE (section 2), lui, est bâti sur Keccak (qui est aussi la base de SHA-3, la cryptographie, c'est compliqué). D'ailleurs, le logiciel qui met en œuvre les algorithmes de ce RFC est développé par l'équipe de Keccak.

Ne comptez pas sur moi pour des explications cryptographiques détaillées, je n'y connais rien, lisez les sections 2 et 3 du RFC. Mais Sakura <https://link.springer.com/chapter/10.1007/978-3-319-07536-5_14>, le système utilisé par Kangaroo Twelve pour répartir le travail semble intéressant (cf. l'article original <http://eprint.iacr.org/2013/231.pdf>).

En section 4 du RFC, vous trouverez comment utiliser Kangaroo Twelve pour faire un MAC (on condense le message puis on condense la concaténation de la clé et du résultat de la première condensation).

Et si vous voulez mettre en œuvre Kangaroo Twelve vous-même, la section 5 vous offre de nombreux vecteurs de test. Mais, avant, regardez l'annexe A qui vous offre du pseudo-code et rappelle qu'il y a une implémentation en C, XKCP <https://github.com/XKCP/XKCP> (par l'équipe Keccak) et une en Python, décrite dans l'article « " *KangarooTwelve : fast hashing based on Keccak-p*" <http://eprint.iacr.org/2016/770.pdf> » (les vecteurs de test du RFC viennent de là, normal, ce sont les mêmes auteurs). Commençons en Python (j'ai fait une copie du code de l'article dans et le programme de test est). Si on prend ce vecteur de test dans le RFC :

```
KT128(M=ptn(17 bytes), C=‘00‘^0, 32):
  ‘6B F7 5F A2 23 91 98 DB 47 72 E3 64 78 F8 E1 9B
   0F 37 12 05 F6 A9 A9 3A 27 3F 51 DF 37 12 28 88‘
```

Le programme affiche :

```
% python test-kangarootwelve.py
6b f7 5f a2 23 91 98 db 47 72 e3 64 78 f8 e1 9b 0f 37 12 05 f6 a9 a9 3a 27 3f 51 df 37 12 28 88
```

Ce qui est bien la bonne valeur.

Et avec XKCP, compilons la bibliothèque et le programme de test :

```
git submodule update --init
make generic64/libXKCP.so
gcc -I ./bin/generic64/libXKCP.so.headers  -L ./bin/generic64   test-kangarootwelve.c -lXKCP
# ou
# gcc -I ./bin/generic64/libXKCP.so.headers test-kangarootwelve.c ./bin/generic64/libXKCP.so
LD_LIBRARY_PATH=./bin/generic64 ./a.out
```

Et le programme affiche bien le vecteur de test indiqué dans le RFC.

Kangaroo Twelve a été ajouté au registre IANA <https://www.iana.org/assignments/named-informati
named-information.xml#hash-alg>, et les quatre algorithmes sont dans le registre de COSE <https:
//www.iana.org/assignments/cose/cose.xml#algorithms> (RFC 8152[1]).

1. Pour voir le RFC de numéro NNN, https://www.ietf.org/rfc/rfcNNN.txt, par exemple https://www.ietf.org/rfc/rfc8152.txt