RFC 9923 : The FNV Non-Cryptographic Hash Algorithm

Stéphane Bortzmeyer
<stephane+blog@bortzmeyer.org>

Première rédaction de cet article le 28 février 2026

Date de publication du RFC : Février 2026

https://www.bortzmeyer.org/9923.html

L'algorithme de condensation FNV, présenté dans ce RFC n'est pas neuf (et certains disent même qu'il est dépassé). Mais il est utilisé par de nombreux logiciels alors qu'il n'avait jamais été spécifié « proprement ». Voilà qui est fait.

FNV a comme principal avantage ses performances : il est rapide et le code est court. Comme le précise le titre du RFC, il n'a pas les caractéristiques des algorithmes de condensation utilisés pour des services de sécurité. Mais cela ne l'empêche pas d'être déployé depuis de nombreuses années pour bien d'autres usages.

Le RFC fait 137 pages mais ne vous affolez pas, FNV est simple et l'essentiel de ces pages est composé de code mettant en œuvre FNV de diverses manières. J'emprunte au RFC, section 2 (avec les modifications de l'article de Wikipédia pour que ce soit plus clair) l'essentiel de l'algorithme, en pseudo-code :

```
algorithm fnv-1 is
    hash := FNV_offset_basis

    for each byte_of_data to be hashed do
        hash := hash × FNV_prime
        hash := hash XOR byte_of_data

    return hash
```

C'est court et rapide, non ? Mais, et le RFC le rappelle plusieurs fois (car des critiques sérieuses du projet de RFC avait insisté sur cette question) : cet algorithme ne doit pas être utilisé pour le cas où on veut de la résistance aux attaques comme la pré-image (section 1.1 du RFC). Ainsi, Python utilisait FNV mais a arrêté <https://peps.python.org/pep-0456/>.

Qu'est-ce qui fait que FNV se retrouve qualifié de « non-cryptographique » et pas utilisable pour la sécurité ? La section 1.3 liste les limites de FNV (voir aussi la section 6) comme le fait que chaque bit du condensat ne dépend pas de **tous** les bits de la donnée d'entrée ou comme le fait que FNV est trop rapide. Oui, pour la sécurité, cela peut être un problème : si on utilise une fonction pour condenser des mots de passe, on ne veut pas faciliter la tâche de l'attaquant qui va chercher un mot de passe ayant ce condensat, en essayant beaucoup de mots de passe possibles. Une fonction de condensation a donc intérêt à être lente, comme Argon 2 (RFC 9106 [1]).

Vous trouverez par contre du FNV un peu partout (la section 1.2 vous donne une liste), dès que ces questions de sécurité ne sont pas pertinentes. FNV est même cité dans des RFC, le RFC 7357 ou le RFC 7873 (les "*cookies*" du DNS) ou bien dans la norme IEEE 802.1Qbp <https://standards.ieee.org/ieee/802.1Qbp/5217/>.

FNV a une longue histoire (section 7 du RFC), ayant commencé en 1991. On notera que la définition de FNV inclut quelques valeurs largement arbitraires comme, par exemple, la chaine de caractères « chongo ¡Landon Curt Noll¿ /\../\ », dont le condensat fournit le paramètre `offset_basis` (section 2.2). Cette chaine était dans le .signature d'un des auteurs de FNV mais, c'est amusant, avait été mal recopiée.

Le gros (en nombre de pages) du RFC étant constitué de code source en C, avec des variantes selon le nombre de bits produits, plutôt que de tenter de l'extraire du RFC <https://github.com/martinthomson/rfc-extract>, utilisons cette archive tar (en ligne sur https://www.bortzmeyer.org/files/FNV-RFC.tar.gz) que j'ai constituée :

```
% tar xzvf FNV-RFC.tar.gz

% cd FNV-RFC

% echo -n toto > tmp

% make FNVhash

% ./FNVhash -t 64 -f tmp
FNV-64 test of return values passed.
FNV-64 Hash of contents of file 'tmp': ADC7B038EFF1F42F
```

Et voilà, FNV marche. Il existe d'autres mises en œuvre. Celle de référence est présentée ici <http://www.isthe.com/chongo/tech/comp/fnv/index.html#FNV-reference-source> et est également sur Github <https://github.com/lcn2/fnv> :

```
% git  clone https://github.com/lcn2/fnv

% cd fnv

% make

% ./fnv1a64 -s toto
0x2ff4f1ef38b0c7ad

[Oui, les octets ne sont pas affichés dans le même ordre qu'avec le code du RFC.]
```

1. Pour voir le RFC de numéro NNN, https://www.ietf.org/rfc/rfcNNN.txt, par exemple https://www.ietf.org/rfc/rfc9106.txt

Vous avez aussi d'autres mises en œuvre <http://www.isthe.com/chongo/tech/comp/fnv/index.html#alt-FNV-source> (une des plus rigolotes est en Fortran <http://www.isthe.com/chongo/tech/comp/fnv/fnv32.f>). Et elle n'est pas listée mais il y a une mise en œuvre de FNV <https://github.com/ziglang/zig/blob/master/lib/std/hash/fnv.zig> dans le langage Zig <https://www.bortzmeyer.org/mes-debuts-en-zig.html> (notez son utilisation de comptime <https://ziggit.dev/t/outreach-looking-for-a-simple-example-of-using-comptime-instead-2106>).

Et pour finir, si vous vous demandez quelle fonction de condensation utiliser, il y a une comparaison de FNV avec SHA-1 (RFC 3174) et SHA-256 (RFC 6234) dans l'annexe A.