RFC 9106 : Argon2 Memory-Hard Function for Password Hashing and Proof-of-Work Applications

Stéphane Bortzmeyer
<stephane+blog@bortzmeyer.org>

Première rédaction de cet article le 8 septembre 2021

Date de publication du RFC : Septembre 2021

https://www.bortzmeyer.org/9106.html

En général, en informatique, on essaie de diminuer la consommation de mémoire et de processeur des programmes, afin de les rendre plus efficaces. Mais il y a aussi des cas où on va tenter de **réduire** délibérement l'efficacité d'un programme. C'est le cas des fonctions de condensation qui sont utilisées en sécurité : on ne veut pas faciliter la tâche de l'attaquant. Ainsi, Argon2, spécifié dans ce RFC est volontairement très consommatrice de mémoire et d'accès à cette mémoire. Elle peut être utilisée pour condenser des mots de passe, ou pour des sytèmes à preuve de travail.

L'article original décrivant Argon2 est « "*Argon2 : the memory-hard function for password hashing and other applications*" <https://www.cryptolux.org/images/0/0d/Argon2.pdf> » et vous pouvez également lire « "*Argon2 : New Generation of Memory-Hard Functions for Password Hashing and Other Applications*" <https://www.cryptolux.org/images/d/d0/Argon2ESP.pdf> ». Notre RFC reprend la description officielle, mais en s'orientant davantage vers les programmeurs et programmeuses qui devront mettre en œuvre cette fonction. Argon2 est conçu pour être "*memory-hard*", c'est-à-dire faire souvent appel à la mémoire de la machine. Cela permet notamment aux ordinateurs classiques de tenir tête aux systèmes à base d'ASIC. (Par exemple, nombreuses sont les chaînes de blocs utilisant des preuves de travail. La fonction utilisée par Bitcoin, SHA-256, conçue pour être simple et rapide, n'est pas "*memory-hard*" et le résultat est qu'il y a belle lurette qu'un ordinateur, même rapide, n'a plus aucune chance dans le minage de Bitcoin. Seules les machines spécialisées peuvent rester en course, ce qui contribue à centraliser le minage dans peu de fermes de minage.) Argon2 est donc **dur pour la mémoire**, comme décrit dans l'article « "*High Parallel Complexity Graphs and Memory-Hard Functions*" <https://eprint.iacr.org/2014/238.pdf> ». À noter qu'Argon2 est très spécifique à l'architecture x86.

Argon2 se décline en trois variantes, Argon2id (celle qui est recommandée par le RFC), Argon2d et Argon2i. Argon2d a des accès mémoire qui dépendent des données, ce qui fait qu'il ne doit pas être employé si un attaquant peut examiner ces accès mémoire (attaque par canal auxiliaire). Il convient donc

pour du minage de cryptomonnaie mais pas pour une carte à puce, que l'attaquant potentiel peut observer en fonctionnement. Argon2i n'a pas ce défaut mais est plus lent, ce qui ne gêne pas pour la condensation de mots de passe mais serait un problème pour le minage. Argon2id, la variante recommandée, combine les deux et est donc à la fois rapide et résistante aux attaques par canal auxiliaire. (Cf. « "*Tradeoff Cryptanalysis of Memory-Hard Functions*" <https://eprint.iacr.org/2015/227.pdf> », pour les compromis à faire en concevant ces fonctions dures pour la mémoire.) Si vous hésitez, prenez donc Argon2id. La section 4 du RFC décrit plus en détail les paramètres des variantes, et les choix recommandés.

Argon repose sur une fonction de condensation existante (qui n'est pas forcément dure pour la mémoire) et le RFC suggère Blake (RFC 7693 [1]).

La section 3 du RFC décrit l'algorithme mais je n'essaierai pas de vous le résumer, voyez le RFC.

La section 5 contient des vecteurs de test si vous voulez programmer Argon2 et vérifier les résultats de votre programme.

La section 7 du RFC revient en détail sur certaines questions de sécurité, notamment l'analyse de la résistance d'Argon2, par exemple aux attaques par force brute.

1. Pour voir le RFC de numéro NNN, https://www.ietf.org/rfc/rfcNNN.txt, par exemple https://www.ietf.org/rfc/rfc7693.txt