ÉLÉMENT DE PORTFOLIO 02

Autre

1 DÉFINITION DE CET ÉLÉMENT

Titre de l'élément : Théorie, pratique, bidouille et supercalculateurs

URL de l'élément :

- https://doi.org/10.1016/j.parco.2021.102804 ou https://hal.inria.fr/hal-02306904
- https://doi.org/10.13154/tosc.v2020.i3.175-196 ou https://hal.archives-ouvertes.fr/hal-02700791

2 MOTIVATIONS DU CHOIX DE CET ÉLÉMENT

Une partie essentielle des travaux menés au sein de l'équipe ALMASTY concerne la *cryptanalyse*, c'est-à-dire la conception d'attaques qui visent à briser les propriétés de sécurités censées être offertes par les mécanismes cryptographiques. Dans la littérature publiée, beaucoup d'attaques sont "théoriques" seulement : un algorithme est décrit, sa complexité est déterminée dans un modèle calculatoire abstrait, puis il en est conclu qu'il "casse" un mécanisme cryptographique car il s'exécute "trop vite".

La communauté cryptographique est très portée sur les mathématiques et est friande de "théorie". *A contrario*, nous faisons le choix volontariste, militant pourrait-on dire, de ne pas nous confiner à la théorie pure et de chercher à mettre en pratique les algorithmes que nous inventons, à réaliser concrètement les attaques que nous concevons. Nous considérons que cela apporte un autre point de vue, très différent (ce qui est théoriquement supérieur est bien souvent pratiquement inférieur et vice-versa). Cet élément du portfolio présente notre trajectoire intellectuelle à ce propos.

Nous estimons que cette confrontation de point de vue est fructueuse. C'est d'ailleurs l'un des axes essentiels du projet ANR JCJC "GORILLA" porté par l'équipe. Nous nous donnons donc pour objectif d'une part de produire des implantations logicielles de bonne qualité des attaques cryptographiques, et d'autre part de démontrer leur faisabilité en tentant de les exécuter à grande échelle.

L'équipe ALMASTY, depuis sa création, a possédé une expertise significative dans le calcul parallèle et la réalisation de records de calculs.

3 PRÉSENTATION DE CET ÉLÉMENT

3.1 *Computational records with aging hardware : Controlling half the output of SHA-256*

Cette publication est l'aboutissement d'un travail commencé à l'été 2016, qui a donc mis 5 ans à aboutir (c'est de la *slow science*). Tout cela a commencé par une réflexion sur les algorithmes dédiés au problème $3\mathrm{XOR}$: étant donné trois fonctions aléatoires $f, g, h : \{0,1\}^* \to \{0,1\}^n$, trouver un triplet (x, y, z) tel que $f(x) \oplus g(y) \oplus h(z) = 0$ — ici $\oplus$ désigne le XOR des chaînes de n bits. Plusieurs attaques cryptographiques doivent résoudre de tels problèmes [2, 3]. Nous nous sommes notamment posés la question : "*si quelqu'un voulait résoudre une instance du problème en pratique, comment devrait-il faire ?*". Essayer d'y répondre nous a ouvert les yeux sur le fait que toutes les "améliorations" théoriques, qui visent à réduire le nombre total d'opérations, rendaient les algorithmes de moins en moins utilisables en pratique : cela ignore les contraintes d'espace, de parallélisme et de complexité de communication. Pour avancer, nous nous sommes donc donné un défi : calculer un triplet 3XOR sur la fonction de hachage cryptographique $\mathrm{SHA256}$ tronquée à n bits, pour la plus grande valeur possible de n.

Il nous est vite apparu clairement que les "meilleurs" algorithmes plafonnaient à $n \approx 80$ pour cause de manque de mémoire. En 2017, nous parvenions à $n = 96$ avec une méthode naïve sur un petit cluster. En 2018, nous avons publié l'article [1] qui décrivait un nouvel algorithme concrètement plus efficace que l'état de l'art. Pour disposer de plus de puissance de calcul, nous avons acheté sur `leboncoin.fr` une machine à miner des bitcoins de seconde main. Quelques semaines de *hacking* plus tard, nous l'avions convertie en accélérateur de calcul pour la cryptanalyse. Nous l'avons fait fonctionner 8 mois sans interruption. Après cela, en 2019, nous avons soumis

FIGURE 1 – Une machine à miner des bitcoins reprogrammée pour la cryptanalyse, en pleine action (gauche). Les supercalculateurs `turing` (milieu) et `jean-zay` (droite) sur lesquels nous avons mené nos travaux.

un dossier au GENCI pour réaliser 10 millions d'heures de calcul sur la machine `turing` (une IBM Bluegene/Q) de l'IDRIS. Nous avons exécuté une version ultra-optimisée de notre algorithme sur 65536 cœurs simultanément pendant 80 heures pour parvenir à calculer un triplet 3XOR sur $n = 128$ bits. Nous avons donc réussi à "contrôler" la moitié de la sortie de $\mathrm{SHA256}$, ce que personne d'autre n'a fait. En termes de nombre d'opérations, il s'agit du plus "gros" calcul de nature cryptographique jamais réalisé. Il s'agit d'ailleurs du seul et unique résultat de cryptanalyse accéléré par des machines à miner des bitcoins (cf. figure 1).

3.2 *Predicting the PCG Pseudo-Random Number Generator In Practice*

Cette publication décrit une "attaque" pratique sur le générateur pseudo-aléatoire PCG, qui est utilisé par défaut dans la populaire bibliothèque de calcul scientifique `numpy`. L'article décrit une procédure qui permet de calculer la suite du flux pseudo-aléatoire (qui est censé être imprévisible) à partir d'un préfixe de quelques kilo-octets.

Ce travail a été motivé par le site web qui fait la promotion de l'algorithme PCG (cf. fig. 2). Il nous semblait évident que l'algorithme ne peut pas être cryptographiquement sûr, mais l'auteur y affirme de manière ambigüe qu'il pourrait pourtant offrir une certaine sécurité. Pour l'édification de la communauté, nous avons donc entrepris de réfuter cette affirmation.

L'algorithme PCG utilise une combinaison de techniques classiques dans les générateurs pseudo-aléatoires (générateurs linéaires congruentiels tronqués) mais emprunte aussi à la cryptographie symétrique (*data-dependant rotation* comme dans RC5). En venir à bout nécessite tout l'arsenal des techniques de cryptanalyse (réseaux euclidiens, etc.). Après avoir conçu l'attaque, nous avons implanté l'attaque dans un programme C ultra-optimisé. Nous avons demandé à l'auteur de PCG de nous fournir quelques kilo-octets de flux pseudo-aléatoire. Nous avons exécuté notre programme sur 20480 coeurs de `jean-zay` pendant une petite demi-heure et nous lui avons renvoyé la graine utilisée. Ceci a fourni la démonstration irréfutable que l'algorithme est faible.

À notre grande surprise, cette publication a fait l'objet d'une discussion sur le réseau social Reddit [1]. Le code de l'attaque étant fourni, certains se sont amusés à la reproduire. Sur HAL, l'article a par conséquent été téléchargé 3600 fois, alors que nos autres publications plafonnent à quelques centaines de téléchargement.

Bref : justiciers du cyberespace, *buzz* sur les réseaux sociaux, et supercalculateurs pour mettre tout le monde d'accord, voilà qui résume bien l'équipe ALMASTY.

4 RÉFÉRENCES BIBLIOGRAPHIQUES

[1] Charles Bouillaguet, Claire Delaplace, and Pierre-Alain Fouque. Revisiting and improving algorithms for the 3XOR problem. *IACR Trans. Symmetric Cryptol.*, 2018(1) :254–276, 2018.

[2] Gaëtan Leurent and Ferdinand Sibleyras. Low-memory attacks against two-round even-mansour using the 3-XOR problem. In Alexandra Boldyreva and Daniele Micciancio, editors, *Advances in Cryptology - CRYPTO 2019 - 39th Annual International Cryptology Conference, Santa Barbara, CA, USA, August 18-22, 2019, Proceedings, Part II*, volume 11693 of *Lecture Notes in Computer Science*, pages 210–235. Springer, 2019.

[3] Mridul Nandi. Revisiting Security Claims of XLS and COPA. *IACR Cryptology ePrint Archive*, 2015 :444, 2015.

1. `https://www.reddit.com/r/programming/comments/ja0tji/predicting_the_pcg_pseudorandom_number_generator/`

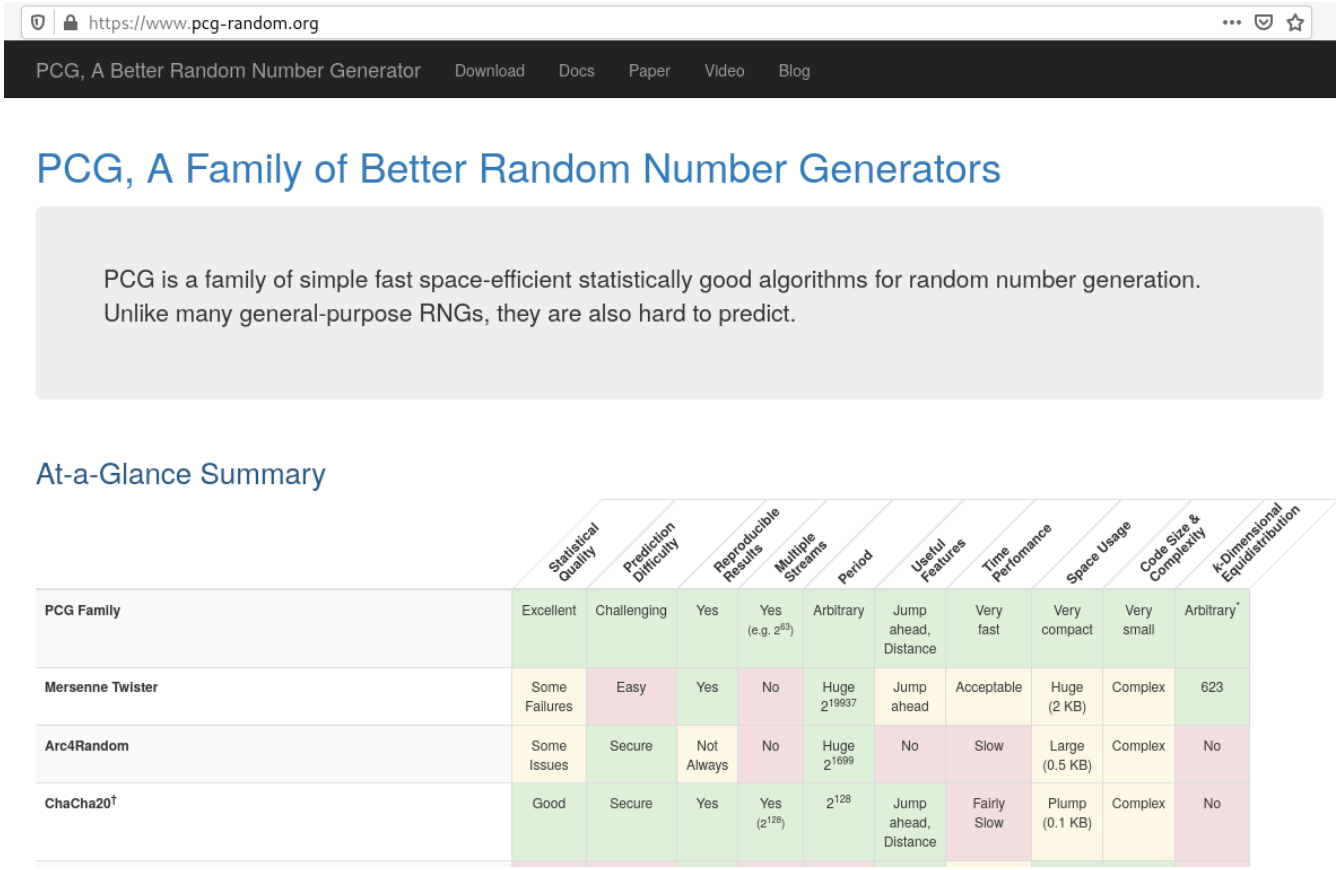

FIGURE 2 – Le site web faisant une promotion non méritée du générateur pseudo-aléatoire PCG.