Cryptanalyse

June 2025

1 Entretien Galactique 2

Solution On prend 3 entiers premiers p_1, p_2 et p_3 s'écrivant sur 64 bits (donc plus petits que 2^{64}).

On te donne accès à S_1 et S_3 tels que :

$$S_i = p_1^i + p_2^i + p_3^i \tag{1}$$

Tu peux vérifier (en développant bêtement) que :

$$\frac{S_1^3 - S_3}{3} = (\prod_{i=1}^{3} S1 - p_i) \tag{2}$$

Maintenant tu itères sur tous les diviseurs de la partie de gauche (puisque tu peux la calculer), tu enlèves S_1 et tu vérifies si ça te donnes un nombre premier négatif supérieur à -2^{64}. Dans ce cas, tu stockes son opposé. Généralement tu n'en trouves que 3, mais si tu en trouves plus, et bien tu testes tous les triplets possibles...

2 Lunettes 2

Mise en équations Je suppose que seule la première partie (i.e. récupérer a et b) t'intéresse, car la seconde est la même que pour la lunette partie 1.

Tu as 3 nombres inconnus $x_i, 1 <= i < 4$ tels que :

$$x_{i+1} = a * x_i + b[2^{2028}] \tag{3}$$

Tu as 3 nombres premiers $p_i, 1 <= i < 4$ tels que :

$$p_i = x_i + \epsilon_i \tag{4}$$

Tu sais que statistiquement, les ϵ_i sont petits (disons inférieurs à 1000).

Enfin on te donne 3 nombres $y_i, 1 <= i < 4$ tels que :

$$p_i = y_i * 2^{64} + l_i \tag{5}$$

$$0 <= l_i < 2^{64} \tag{6}$$

Voilà ce qu'on a, et on cherche a et b.

Quand tu veux utiliser l'algorithme lll, tu cherches à composer des nombres "petits" à partir de nombres "grands". Ici, ce qui est "petit", c'est a, b, les l_i et les ϵ.

Si on mixe tout ce qu'on a en haut, on peut faire :

$$x_2 - x_1 = p_2 - p_1 + \epsilon_1 - \epsilon_2 = 2^{64} * (y_2 - y_1) + o(2^{512}) \tag{7}$$

De même :

$$x_3 - x_2 = p_3 - p_2 + \epsilon_2 - \epsilon_3 = 2^{64} * (y_3 - y_2) + o(2^{512}) \tag{8}$$

Or tu sais que :

$$x_3 - x_2 = a * (x_2 - x_1) + K * 2^{2048} \tag{9}$$

Du coup tu as :

$$2^{64} * (y_3 - y_2) = a * 2^{64} * (Y_2 - Y_1) + K * 2^{2048} + \epsilon \tag{10}$$

Avec ϵ petit. Donc si tu poses les vecteurs suivants :

$$v_1 = [1, 0, 0, 2^{64} * (y_2 - y_1)] \tag{11}$$

$$v_2 = [0, 1, 0, 2^{64} * (y_3 - y_2)] \tag{12}$$

$$v_3 = [0, 0, 1, 2^{2048}] \tag{13}$$

Alors tu sais que le vecteur $a * v_1 - v_2 + K * v_3$ est "petit". Donc si tu appliques l'algorithme lll sur ce jeu de 3 vecteurs, et si parmi ce que que ça te retourne, le coefficient associé à v_2 est 1 (ou -1), alors tu as trouvé a (ou -a) !

C'est la moitié du travail de fait.

Trouver b Sauf qu'on n'a pas utilisé toutes les informations à notre portée !!! On a considéré que les l_i et les ϵ_i étaient négligeables (ce qui est vrai par rapport aux grandeurs considérées), mais dans leur petitesse, il y a aussi une hiérarchie.

En effet les l_i sont bien plus grands que les ϵ_i ! On va utiliser cela.

Tout d'abord, on sait que :

$$(a * v_1 - v_2 + K * v_3)[3] = a * (l_2 - l_1) + o(2^{512+64}) \tag{14}$$

Or, on a accès à a et K, donc on trouve directement $l_2 - l_1$ en divisant par a (vu qu'on ne connaît pas le signe de l'erreur, on obtient la bonne valeur à +/- 1 près).

On peut également développer p_2 en faisant :

$$p_2 = x_2 + \epsilon_2 = a * x_1 + b + \epsilon_2 + K^* * 2^{2048} \tag{15}$$

Donc si on note :

$$A = 2^{64} * y_2 - a * 2^{64} * y_1 + l_2 - l_1 \tag{16}$$

Alors :

$$A - l_1 * a - K^* * 2^{2048} = o(2^{512+64}) \tag{17}$$

Bon ben on connaît la musique désormais :

$$w_1 = [1, 0, 0, A] \tag{18}$$

$$w_2 = [0, 1, 0, a] \tag{19}$$

$$w_3 = [0, 0, 1, 2^{2048}] \tag{20}$$

On fait un lll sur ces 3 vecteurs. Si le premier terme d'un des vecteurs trouvés vaut 1 (ou -1), alors le deuxième vaut l_1 (ou $-l_1$).

On connait l_1 et $l_2 - l_1$ (à 1 près), donc on peut trouver les valeurs l_1 et l_2 telles que $(y_2 * 2^{64} + l_2 - a * (y_1 * 2^{64} + l_1)) \bmod 2^{2048}$ soit un nombre premier de 512 bits. Si oui, c'est que c'est b !