BabyPRNG writeup

This writeup was generated by Google Translate from the original Chinese version, sorry for inconvenience.

This task mainly examines the truncated bit sequence prediction problem of Elliptic Curve Power Generator (ECPG) [1].

Description

First select a regular elliptic curve on a prime field $\mathbb{F}_p$

$$E_{\mathbb{F}_p} : y^2 = x^3 + ax + b,$$

Where a, b are unknown secret parameters.

Next, randomly select a point $G \in E_{\mathbb{F}_p}$ on the curve, and each random number r is generated according to the following process:

$$\begin{gathered} G \leftarrow 1337 \cdot G \\ r_i \leftarrow G.x >> 32 \\ r_{i+1} \leftarrow G.y >> 32 \end{gathered}$$

In this question, the above steps are carried out three times, that is, to provide 6 consecutive random numbers $r_0, r_1, \ldots, r_5$ output by the generator. At the same time, the RSA encryption result of `flag` $c = (a^{3371} * \texttt{flag} + b^{3713})^{1337} \mod p$, as well as the 1024-bit prime number p are given.

Problem Solving Ideas

Obviously, in order to solve `flag`, it is enough to restore a, b. After some searching, you can find a paper [2] on ECPG attack based on the Coppersmith method. The attack is given in Section 5 of the article, but the attack is applicable to the situation where a, b are known, which is not scene of this topic. In fact, Section 4 of the article gives an attack against another type of random number generator ECLCG when a, b are unknown, but this attack is also applicable to the ECPG generator in this question: It may be assumed that the corresponding elliptic curve points for three iterations are $G_0 = (x_0, y_0), G_1 = (x_1, y_1), G_2 = (x_2, y_2)$, then the following formulas hold:

$$y_0^2 = x_0^3 + ax_0 + b$$
$$y_1^2 = x_1^3 + ax_1 + b$$
$$y_2^2 = x_2^3 + ax_2 + b$$

Although a, b are unknown, they can be eliminated, and finally a six-element quaternary equation about $x_0, x_1, x_2, y_0, y_1, y_2$ is obtained

$$f = (y_0^2 - y_2^2 - (x_0^3 - x_2^3)) * (x_0 - x_1) - (y_0^2 - y_1^2 - (x_0^3 - x_1^3)) * (x_0 - x_2)$$

In this question, most of the bits of these six variables are given, only the lower 32 bits are discarded, so we can write them in the form of $x_0 = x_0^* + x_0'$, where x_0^* means The part given by the title, and x_0' represents the unknown part. In this way, f can be regarded as an equation about $x_0', x_1', x_2', y_0', y_1', y_2'$, and notice that the values of these six variables are very small, no more than 2^{32}. The paper [2] uses the Coppersmith method to solve this equation, but the actual complexity is so high that it cannot be done on a laptop. Therefore, this question requires players to have a certain understanding of the principle of the attack, so as to optimize it.

In fact, a simpler method is to use the LLL reduction algorithm directly. For ease of understanding, a simple example is used to describe it below. Consider the polynomial $h = Ax^2y + Bxy + Cy + Dx + E \in \mathbb{F}_p[x, y]$, now we want to find a set of small-valued roots (x', y'), which satisfies $x' < U, y' < U$, and S is a number much larger than p. Consider constructing the following lattice:

$$\mathbf{L} = \begin{bmatrix} S*p & & & & & \\ S*A & 1 & & & & \\ S*B & & U & & & \\ S*C & & & U^2 & & \\ S*D & & & & U^2 & \\ S*E & & & & & U^3 \end{bmatrix}$$

Obviously, this grid contains vector $\mathbf{v} = (0, x'^2y', Ux'y', U^2y', U^2x', U^3)$, that is, there is a certain vector $\mathbf{u}$ satisfies $\mathbf{uL} = \mathbf{v}$. And each component in $\mathbf{v}$ is very small, which means that $\mathbf{v}$ can be found out by lattice reduction method, and then (x', y') can be restored.

Back to this question, construct a lattice corresponding to f, and then use the lattice reduction algorithm to restore $x_0', x_1', x_2', y_0', y_1', y_2'$, and then calculate a, b decrypt `flag`. 32 bits are truncated in the title, and the LLL algorithm may not be able to solve it, so you can consider enumerating 2 bits for each variable, and there are 2^{12} possibilities in total.

References

[1] Lange T, Shparlinski I E. Certain exponential sums and random walks on elliptic curves[J].

Canadian Journal of Mathematics, 2005, 57(2): 338-350.

[2] Mefenza T, Vergnaud D. Inferring sequences produced by elliptic curve generators using Coppersmith's methods[J]. Theoretical Computer Science, 2020, 830: 20-42.