Pollard's Algorithm for Discrete Logarithm Problem

Professor Dr. D. J. Guan *

December 8, 2013

Let p be a prime. The *discrete logarithm problem* over Z_p is:

Given a and b, find x such that $a^x \equiv b \pmod{p}$.

The security of many cryptosystems is based on the difficulty of solving the discrete logarithm problem. In this lecture note, we shall discuss the Pollard's algorithm to solve the discrete logarithm problem. It is also called *square-root attack*.

Pollard's algorithm is a Monte Carlo type probabilistic algorithm. The algorithm first finds two numbers u and v such that

$$a^u \equiv b^v \pmod{p}. \tag{1}$$

The method used in this step is to find a sequence of numbers $x_0, x_1, \ldots$ until two equal ones are found. The sequence is defined to be:

$$x_0 = 1 \tag{2}$$

$$x_{i+1} = \begin{cases} bx_i & \text{if } 0 < x_i \le p/3 \\ x_i^2 & \text{if } p/3 < x_i \le 2p/3 \\ ax_i & \text{if } 2p/3 < x_i < p \end{cases} \tag{3}$$

Note that x_i can also be expressed as

$$x_i \equiv a^{\alpha_i} b^{\beta_i} \pmod{p}.$$

To compute the sequence, the algorithm computes

$$\alpha_0 = 0$$

$$\alpha_{i+1} = \begin{cases} \alpha_i & \text{if } 0 < x_i \le p/3 \\ 2\alpha_i & \text{if } p/3 < x_i \le 2p/3 \\ \alpha_i + 1 & \text{if } 2p/3 < x_i < p \end{cases}$$

*Department of Computer Science, National Sun Yar-Sen University, Kaohsiung, Taiwan 80424 (guan@cse.nsysu.edu.tw).

$$\begin{aligned} \beta_0 &= 0 \\ \beta_{i+1} &= \begin{cases} \beta_i + 1 & \text{if } 0 < x_i \le p/3 \\ 2\beta_i & \text{if } p/3 < x_i \le 2p/3 \\ \beta_i & \text{if } 2p/3 < x_i < p \end{cases} \end{aligned}$$

The algorithm needs not store all the numbers to find an equal pair. The algorithm run through the sets

$$(x_i, \alpha_i, \beta_i; x_{2i}, \alpha_{2i}, \beta_{2i}), \qquad i = 1, 2, \ldots,$$

generating each one from the previous one, until the one with $x_i = x_{2i}$ is generated. Then we have $a^u \equiv b^v \pmod{p}$, where

$$\begin{aligned} u &= \alpha_{2i} - \alpha_i \pmod{p-1} \\ v &= \beta_i - \beta_{2i} \pmod{p-1}. \end{aligned}$$

Note that the sequence $\{x_i\}$ can be regarded as a random sequence. It is estimated that the period of the random sequence is

$$\sqrt{\frac{\pi^5 p}{288}} \approx 1.0308\sqrt{p}.$$

That is, the algorithm may need to try $O(\sqrt{p})$ iterations to obtain the right pair x_i and x_{2i}.

The second step is to find x. If $v \equiv 0 \pmod{p-1}$, then the algorithms fails. Assume that $v \not\equiv 0 \pmod{p-1}$. The algorithm then finds d, the greatest common divisor of v and $p-1$, by the Extended Euclid algorithm. That is,

$$d = (v, p-1) = v\nu + (p-1)\mu.$$

Raising both sides of $a^u \equiv b^v \pmod{p}$ to the power ν gives

$$a^{u\nu} \equiv b^{v\nu} \equiv b^{d-(p-1)\mu} \equiv b^d \equiv (a^x)^d \pmod{p}.$$

Therefore, $xd \equiv u\nu \pmod{p-1}$, which implies that

$$xd = u\nu + w(p-1).$$

Since $d \mid |(p-1)$, hence $d \mid |n\nu$. We obtain

$$x = (u\nu + w(p-1))/d. \tag{4}$$

The value of w is between 0 and d. In the case that d is small, we can do an exhaustive search. If d is large, then the search will take a long time to finish.

If $d = 1$, that is $(v, p-1) = 1$, then equation 4 can be reduced to

$$x = u\nu = uv^{-1} \pmod{p-1}.$$

The above algorithm can also be adapted to work on the factors of $p-1$. This is called the multistage method. Let $p-1 = st$. The method is to use a^s and b^s to replace a and b. This leads to find u and v such that

$$(a^s)^u \equiv (b^s)^v \pmod{p}.$$

The other steps are similar.

Note that there are more efficient algorithms to solve the discrete logarithm problem, such as the Pohlig-Hellman algorithm and the index calculus method. All these algorithms can work on smaller subgroups if $p-1$ can be factored. This is why $p-1$ should have large prime factors to resist the square-root attack.

References

[1] J. M. Pollard. Monte Carlo method for index computation (mod p). *Mathematics of Computation*, 32(143):918–924, July 1978.