Wizardry and Studies

To the spirit that lives in the computer.

2005-11-07

Pollard rho discrete logarithm in Python

It is one of Pollard's rho methods, and one of famous random algorithm. A limited form of the method is presented in Python. The limitations are 1) P = 2N + 1, for P, N are prime; 2) g generates a sub group with order N.

```
#!/usr/bin/env python

# A simple Pollard rho discrete logarithm
# implementation and has some limitations:
# 1. P must be a prime that equals 2N + 1
# 2. N must be a prime, too
# 3. G generates a sub group with order N
# 4. A belongs to <G>, the sub group generated by G
# these four limitations made this program simpler

# x = log_g(a) in Zp

P = long(raw_input())
G = long(raw_input())
A = long(raw_input())
N = (P - 1) / 2

# assert: classify(1) != 1
def classify(x):
        # 3n + 2 -> S0
        # 3n     -> S1
        # 3n + 1 -> S2
        return (x + 1) % 3

def succssor(x, s, t):
        c = classify(x)
        if c == 0:
                return A * x % P, s + 1 % N, t
        elif c == 1:
                return x * x % P, 2 * s % N, 2 * t % N
        else: # c == 2
                return G * x % P, s, t + 1 % N

def ext_euclid(a, b) :
        if b == 0 :
                return (a, 1, 0)
        else :
                (d, xx, yy) = ext_euclid(b, a % b)
                x = yy
                y = xx - (a / b) * yy
                return (d, x, y)

def inverse(a, n):
        return ext_euclid(a, n)[1]

def discrete_log():
        # Pollard rho discrete log method

        xa, sa, ta = 1, 0, 0
        xb, sb, tb = succssor(1, 0, 0)
        while xa != xb:
                xa, sa, ta = succssor(xa, sa, ta)
                xb, sb, tb = succssor(xb, sb, tb)
                xb, sb, tb = succssor(xb, sb, tb)
```

Search This Blog

Search

Labels

- Algorithm (10)
- snippets (1)
- Takeaways from Books (1)

Archive

► 2012 (2)
► 2006 (4)
▼ 2005 (10)
► 12 (5)
▼ 11 (5)
LZW Implementation
Pollard rho discrete logarithm (2)
Pollard rho discrete logarithm in Python
Rabin-Miller Primality Test in C
Chinese Remainder Theorem in Python

Subscribe To

Posts
Comments

Your Friendly Neighborhood

Che-Liang Chiou

View my complete profile

```
        s, t = sa - sb, tb - ta
        if s == 0:
                return 'fail'
        if s < 0:
                s = s + N
        if t < 0:
                t = t + N

        return inverse(s, N) * t % N

print discrete_log()
```


No comments:

Post a Comment

Links to this post

Create a Link

Newer Post Home Older Post

Subscribe to: Post Comments (Atom)