DiffieHellman

November 23, 2022

[94]:
```python
p = 1117
g = 6
h = 527
```

[117]:
```python
from math import ceil,sqrt

class DiffieHellmanProtocol:
    def __init__(self, p, g):
        self.p=p
        self.g=g

    def potega_m(self, m):
        """use python embedded function pow(a,b,n) === a^b (mod n)"""
        return pow(self.g,m,self.p)

    @staticmethod
    def generator(g, p):
        #Z_p* order = phi(p) = p-1
        group_order = p-1
        for i in range(1,g-1):
            if (pow(g,i,p)==1):
                return False
        return pow(g,group_order,p)==1

    @staticmethod
    def gcdExtended(a, p):
        """
        Find such numbers x,y that: GCD(a,p) = a*x + b*y
        where GCD(a,p) - greatest comon divisor of a and b
        (Extended Euclidean algorithm)
        """
        if a == 0 :
            return p,0,1
        gcd,x1,y1 = DiffieHellmanProtocol.gcdExtended(p%a, a)
        x = y1 - (p//a) * x1
        y = x1
        return gcd,x,y #x = a^-1
```

```
    @staticmethod
    def euklid(a,p):
        """
        Use Euclidean algorithm for finding a^-1:
        GCD(a,p) = 1 = a*a^-1 + p*b <=>
        solve a*a^-1 === 1 (mod p) where a^-1 is a variable in Z_p*
        """
        return DiffieHellmanProtocol.gcdExtended(a,p)[1]%p

    def findPower(self, h):
        """
        Solve the equation h = g^x mod p given a prime number p.

        For more detailed description, of the algorightm, see
        https://en.wikipedia.org/wiki/Baby-step_giant-step#The_algorithm
        """
        # phi(p) is p-1 if p is prime
        N = ceil(sqrt(self.p - 1))

        # Baby step:
        # Store hashmap of g^{1, ... m} (mod p).
        tbl = {pow(self.g, i, self.p): i for i in range(N)}
        c = pow(self.g, N * (self.p - 2), self.p)

        # Giant step:
        # Search for an equivalence in the table.
        for j in range(N):
            y = (h * pow(c, j, self.p)) % self.p
            if y in tbl:
                a = j * N + tbl[y]
                assert pow(self.g,a,self.p) == h
                return a
        return None

if __name__=="__main__":
    dh = DiffieHellmanProtocol(p,g)
    # g^a = h
    a = dh.findPower(h)
    assert pow(g,a,p)==h
    print(a)
```

123

[ ]: