ichunqiu的Round Rabins!的writeup

原创

> John gave up on RSA and moved to Rabin. ...he still did it wrong though flag.txt What a box!

这题是Rabin密码体制，该密码详细介绍：https://en.wikipedia.org/wiki/Rabin_cryptosystem

通过yafu可以将pq分解出来，yafu的使用方法见：https://blog.csdn.net/dchua123/article/details/105444230：

```
P154 = 868357428980839855168069059631251918871234401992999056369686301440381835665240313935930358309462389 3
P154 = 868357428980839855168069059631251918871234401992999056369686301440381835665240313935930358309462389 3
```

将pq转16进制后丢下面脚本求解：

```
#!/usr/bin/env python
'''
Rabin cryptosystem challenge:
N=0x6b612825bd7972986b4c0ccb8ccb2fbcd25fffbadd57350d713f73b1e51ba9fc4a6ae862475efa3c9fe7dfb4c89b4f92e925ce8



c=0xd9d6345f4f961790abb7830d367bede431f91112d11aabe1ed311c7710f43b9b0d5331f71a1fccbfca71f739ee5be42c16c6b4d
'''
# some functions from http://codereview.stackexchange.com/questions/43210/tonelli-shanks-algorithm-implemen
def legendre_symbol(a, p):
    """
    Legendre symbol
    Define if a is a quadratic residue modulo odd prime
    http://en.wikipedia.org/wiki/Legendre_symbol
    """
    ls = pow(a, (p - 1)/2, p)
    if ls == p - 1:
        return -1
    return ls

def prime_mod_sqrt(a, p):
```

```
    """
    Square root modulo prime number
    Solve the equation
        x^2 = a mod p
    and return list of x solution
    http://en.wikipedia.org/wiki/Tonelli-Shanks_algorithm
    """
    a %= p

    # Simple case
    if a == 0:
        return [0]
    if p == 2:
        return [a]

    # Check solution existence on odd prime
    if legendre_symbol(a, p) != 1:
        return []

    # Simple case
    if p % 4 == 3:
        x = pow(a, (p + 1)/4, p)
        return [x, p-x]

    # Factor p-1 on the form q * 2^s (with Q odd)
    q, s = p - 1, 0
    while q % 2 == 0:
        s += 1
        q //= 2

    # Select a z which is a quadratic non resudue modulo p
    z = 1
    while legendre_symbol(z, p) != -1:
        z += 1
    c = pow(z, q, p)

    # Search for a solution
    x = pow(a, (q + 1)/2, p)
    t = pow(a, q, p)
    m = s
    while t != 1:
        # Find the lowest i such that t^(2^i) = 1
        i, e = 0, 2
        for i in xrange(1, m):
            if pow(t, e, p) == 1:
                break
            e *= 2

        # Update next value to iterate
        b = pow(c, 2**(m - i - 1), p)
        x = (x * b) % p
        t = (t * b * b) % p
        c = (b * b) % p
        m = i

    return [x, p-x]

def egcd(a, b):
    if a == 0:
```

```
        return (b, 0, 1)
    else:
        g, y, x = egcd(b % a, a)
        return (g, x - (b // a) * y, y)

def modinv(a, m):
    g, x, y = egcd(a, m)
    if g != 1:
        raise Exception('modular inverse does not exist')
    else:
        return x % m


# This finds a solution for c = x^2 (mod p^2)
def find_solution(c, p):
    '''
    Hensel lifting is fairly simple.  In one sense, the idea is to use
    Newton's method to get a better result.  That is, if p is an odd
    prime, and

                            r^2 = n (mod p),

    then you can find the root mod p^2 by changing your first
    "approximation" r to

                        r - (r^2 - n)/(2r) (mod p^2).

    http://mathforum.org/library/drmath/view/70474.html
    '''
    n = p ** 2
    # Get square roots for x^2 (mod p)
    r = prime_mod_sqrt(c,p)[0]

    inverse_2_mod_n = modinv(2, n)
    inverse_r_mod_n = modinv(r, n)

    new_r = r - inverse_2_mod_n * (r - c * inverse_r_mod_n)

    return new_r % n

if __name__ == "__main__":
    # These are the given values
    n = 0x6b612825bd7972986b4c0ccb8ccb2fbcd25fffbadd57350d713f73b1e51ba9fc4a6ae862475efa3c9fe7dfb4c89b4f92e
    # n is a perfect square: n = p * p
    p = 0xa5cc6d4e9f6a893c148c6993e1956968c93d9609ed70d8366e3bdf300b78d712e79c5425ffd8d480afcefc71b50d85e09
    # encrypted message
    c = 0xd9d6345f4f961790abb7830d367bede431f91112d11aabe1ed311c7710f43b9b0d5331f71a1fccbfca71f739ee5be42c1

    solution = find_solution(c, p)
    print hex(solution)[2:-1].decode("hex")
```

flag为： `IceCTF{john_needs_to_get_his_stuff_together_and_do_things_correctly}`

参考：https://github.com/WCSC/writeups/tree/master/icectf-2016/Round-Rabins