hw3

November 18, 2023

Q1. Compute the following Jacobi symbols $\left(\frac{1923}{3761}\right)$ and $\left(\frac{10002}{9765}\right)$

[62]:
```python
def modpow(x: int, n: int, m: int) -> int:
    if n == 0:
        return 1

    current = x
    result = 1
    while n > 0:
        if (n & 1):
            result = (result * current) % m
        current = (current ** 2) % m
        n = n >> 1

    return result

def legendre(a: int, p: int):
    result = modpow(a, (p-1)//2, p)
    if result == p - 1:
        return -1
    else:
        return result

def jr1(m: int, n: int) -> tuple[int, int]:
    assert m % 2 == 1 and n % 2 == 1, "This rule requires m and n be odd."
    if m % 4 == n % 4 and m % 4 == 3:
        return (-n, m)
    else:
        return (n, m)

def jr2(m: int, n: int) -> tuple[int, int]:
    assert n % 2 == 1, "This rule requires n is odd."
    return (m % n, n)

def jr3(m: int, n: int) -> tuple[int, int]:
    assert n % 2 == 1, "This rule requires n is odd."

    k = 0
```

```
    while m & 1 == 0:
        m >>= 1
        k += 1

    if m != 1:
        return ((jr4(2, n) ** k) * m, n)
    else:
        return jr4(2, n) ** k

def jr4(m: int, n: int) -> int:
    assert m == 2, "This rule requires m == 2."
    assert n % 2 == 1, "This rule requires n is odd."
    if n % 8 == 1 or n % 8 == 7:
        return 1
    elif n % 8 == 3 or n % 8 == 5:
        return -1

func_names = {
    jr1: "rule 1",
    jr2: "rule 2",
    jr3: "rule 3",
    jr4: "rule 4",
    legendre: "legendre"
}

# i'm sure i could be writing an algorithm for this...
def do_p1():
    p1funcs = [jr1, jr2, jr3, jr2, jr3, jr1, jr2, jr1, jr2, jr1, jr2, jr3]
    p1 = (1923, 3761)
    print(p1)
    for func in p1funcs:
        p1 = func(*p1)
        print(p1, func_names[func])

def do_p2():
    p2funcs = [jr2, jr1, jr2, jr3, jr1, jr2, legendre]
    p2 = (10002, 9765)
    print(p2)
    for func in p2funcs:
        p2 = func(*p2)
        print(p2, func_names[func])

do_p1()
do_p2()
```

```
(1923, 3761)
(3761, 1923) rule 1
(1838, 1923) rule 2
```

```
(-919, 1923) rule 3
(1004, 1923) rule 2
(251, 1923) rule 3
(-1923, 251) rule 1
(85, 251) rule 2
(251, 85) rule 1
(81, 85) rule 2
(85, 81) rule 1
(4, 81) rule 2
1 rule 3
(10002, 9765)
(237, 9765) rule 2
(9765, 237) rule 1
(48, 237) rule 2
(3, 237) rule 3
(237, 3) rule 1
(0, 3) rule 2
0 legendre
```

A1.

$(\frac{1923}{3761}) \implies (\frac{3761}{1923}) \implies (\frac{1838}{1923}) \implies (\frac{2}{1923})(\frac{919}{1923}) \implies (\frac{-919}{1923}) \implies (\frac{1004}{1923}) \implies (\frac{2}{1923})^2(\frac{251}{1923}) \implies (\frac{251}{1923}) \implies (\frac{-1923}{251}) \implies (\frac{85}{251}) \implies (\frac{251}{85}) \implies (\frac{81}{85}) \implies (\frac{9}{85})^2 = 1$

$(\frac{10002}{9765}) \implies (\frac{237}{9765}) \implies (\frac{9765}{237}) \implies (\frac{48}{237}) \implies (\frac{2}{237})^4(\frac{3}{237}) \implies (\frac{3}{237}) \implies (\frac{237}{3}) \implies (\frac{0}{3}) = 0$

Q2. Devise an RSA scheme based on $p = 11279$ and $q = 11491$. Identify all parameters, choose any three plaintext numbers and encrypt and decrypt them.

[63]:
```python
import random

def extended_gcd(a, b):
    if a == 0:
        return b, 0, 1
    else:
        g, x, y = extended_gcd(b % a, a)
        return int(g), int(y - (b // a) * x), int(x)

p = 11279
q = 11491
n = p*q
tot_n = (p-1) * (q-1)

def rsa_encrypt(x: int, b: int, n: int):
    return modpow(x, b, n)

def rsa_decrypt(y: int, a: int, n: int):
    return modpow(y, a, n)
```

```python
def generate_random_b(b: int, tot_n: int) -> tuple[int, int]:
    b = 2
    while extended_gcd(b, tot_n)[0] != 1:
        b = random.randint(4, 50*int(tot_n ** 0.5))
    _, a, _ = extended_gcd(b, tot_n)
    return b, a

# got these variables from doing generate_random_b
b = 520381
beta = 3007141

xs = [7, 49, 343]
ys = [rsa_encrypt(x, b, n) for x in xs]

print(n, tot_n)
xs, ys, [rsa_decrypt(y, beta, n) for y in ys]
```

```
129606989 129584220
```

[63]:
```
([7, 49, 343], [39152966, 89090032, 47873123], [7, 49, 343])
```

A2. Final parameters:

- $n = 129606989$
- $\phi(n) = 129584220$
- $b = 520381$
- $a = 3007141$

Results:

- $e(7) = 7^{520381} \mod 129606989 = 39152966$
- $e(49) = 49^{520381} \mod 129606989 = 89090032$
- $e(343) = 343^{520381} \mod 129606989 = 47873123$
- $d(39152966) = 39152966^{3007141} \mod 129606989 = 7$
- $d(89090032) = 89090032^{3007141} \mod 129606989 = 49$
- $d(47873123) = 47873123^{3007141} \mod 129606989 = 343$

Q3. Devise an ElGamal scheme with $p = 11279$. Choose any two plaintext numbers and encrypt and decrypt them.

[64]:
```python
from math import gcd

def is_coprime(a: int, b: int):
    return gcd(a, b) == 1

def get_z_s(z: int) -> set[int]:
    return set(filter(lambda x: is_coprime(x, z), range(z)))
```

```python
def is_generator(x: int, z: int, z_s: set[int] | None = None, debug: bool =␣
↪False):
    if z_s is None:
        z_s = get_z_s(z)

    if x not in z_s:
        return False

    n: int = x
    generated: set[int] = set([x])
    for _ in range(len(z_s)):
        n = (n * x) % z
        if n in generated:
            break
        else:
            generated.add(n)

    if debug:
        print(x, generated)
    return z_s == generated

def elgamal_encrypt(x: int, alpha: int, beta: int, k: int, p: int) ->␣
↪tuple[int, int]:
    return (modpow(alpha, k, p), (x*modpow(beta, k, p)) % p)

def elgamal_decrypt(y1: int, y2: int, a: int, p: int) -> int:
    _, y1_inv, _ = extended_gcd(modpow(y1, a, p), p) # calculate the inverse of␣
↪y1
    return (y2 * y1_inv) % p

p = 11279

alpha = 311 # verified through is_generator and modpow
a = 2401 # i like powers of 7
k = 3125 # i also like 5^5
beta = modpow(alpha, a, p)

xs = [1, 2, 3]
ys = list(map(lambda x: elgamal_encrypt(x, alpha, beta, k, p), xs))
txs = list(map(lambda y: elgamal_decrypt(*y, a, p), ys))

xs, ys, txs
```

[64]: ([1, 2, 3], [(7066, 7134), (7066, 2989), (7066, 10123)], [1, 2, 3])

A3. Final Parameters:

- $\alpha = 311$

- $\beta = 5233$
- $a = 2401$
- $k = 3125$

Results: - $e(1) = (311^{3125}, 1(5233^{2401})) = (7066, 7134)$ - $e(2) = (311^{3125}, 2(5233^{2401})) = (7066, 2989)$ - $e(3) = (311^{3125}, 3(5233^{2401})) = (7066, 10123)$ - $d((7066, 7134)) = (7134)(7066^{2401})^{-1} = 1$ - $d((7066, 2989)) = (2989)(7066^{2401})^{-1} = 2$ - $d((7066, 10123)) = (10123)(7066^{2401})^{-1} = 3$