**Mathematics 3159A** Introduction to Cryptography

**Group Assignment 2** Due Date: October 13, 2020

## Report

**Group members.**

- Manager: John Fischer, jfisch25@uwo.ca
- Programmer: Peter Beaulieu, pbeauli2@uwo.ca
- Reporter: Dat Vo, dvo9@uwo.ca
- Scribe: Jay Shah, jshah48@uwo.ca

**Group meetings.** The group met for three meetings that were held on Friday, Sunday, and Monday at 4pm, 10am, and 7pm respectively. Each zoom meeting featured members joining the room within a minute after the scheduled time and the meetings lasted for 20 minutes with the host John closing the room for everyone after it was over.

**Timeline.** The meetings were scheduled at the beginning of the first meeting on Friday and the problems were worked on individually thereafter. The notes for each problem were compiled using google drive and the document was kept constantly up to date with the latest notes from each group member. Most communication was done using Facebook messenger to address concerns over notes before tentative solutions were uploaded to the google doc. A tentative solution for Q1.1 was uploaded on Friday at 5:45pm after the meeting and was later rewritten on Sunday during the meeting. Tentative solutions for Q1.2 and Q1.3 were added on Saturday at 9pm and finalized on Sunday during the meeting. The program was written on Monday for part 1 at 3pm and part 2 was completed at 7pm on Monday. The report was: written on an ongoing basis starting Friday after the meeting, completed on Monday at 9pm and, edited by each group member using overleaf.com.

**Alternative approaches.** 1.1 and 1.3 were solved quickly. For 1.2, we considered how the solutions in $\mathbb{Z}/p$ and $\mathbb{Z}/q$ would look, but did not make much progress until applying the Chinese Remainder Theorem. Tried various methods to solve for 1.4 such as breaking all $a_i$ into their various decompositions in $\mathbb{Z}/p$ and $\mathbb{Z}/q$.

## Problem 1

If p is an odd prime and a is an integer not divisible by p, we must show that if the congruence

$$x^2 \equiv a \pmod{p}$$

has a solution, then it will have two solutions.

## Solution

*Proof.* Suppose $f \in \mathbb{Z}/p$ is a solution. Then, since $\mathbb{Z}/p$ is a field, we know $-f \in \mathbb{Z}/p$ and of course $(-f)^2 \equiv f^2 \pmod{p}$. Now, towards a contradiction.
Suppose $f \equiv -f \pmod{p}$, then,

$$\begin{aligned} 0 &\equiv f + (-f) \pmod{p} \\ &\equiv f + f \pmod{p} \\ &\equiv 2f \pmod{p} \end{aligned}$$

However, since $p \nmid 2$ and $p \nmid f$, then $f \not\equiv -f \pmod{p}$. So, we have found 2 distinct solutions.
Now, suppose a third solution, $j \in \mathbb{Z}/p$ exists.
Then, using difference of squares

$$j^2 \equiv a \equiv f^2 \pmod{p} \iff 0 \equiv j^2 - f^2 \equiv (j-f)(j+f) \pmod{p}$$

Since $\mathbb{Z}/p$ is a field and all elements are units, either

$$j - f, j + f = 0 \pmod{p} \iff j \equiv \pm f \pmod{p}$$

Thus, only 2 solutions in $\mathbb{Z}/p$ exist.

□

### Difference of Squares proof

Claim: $x^2 - y^2 \equiv (x-y)(x+y) \pmod{p}$

*Proof.* Notice,

$$x^2 - y^2 \equiv (x-y)(x+y) \pmod{p} \iff m \in \mathbb{Z}, mp = x^2 - y^2 - (x-y)(x+y)$$

Now,

$$x^2 - y^2 + (x^2 - xy + xy + y^2) = 0$$

So, taking $m = 0$ we have,

$$x^2 - y^2 \equiv (x-y)(x+y) \pmod{p}$$

as desired. □

## Problem 2

If p is an odd prime and a is an integer not divisible by p, we must show that if the congruence

$$x^2 \equiv a \pmod{p}$$

has a solution, then it will have two solutions.

## Solution

*Proof.* Suppose a solution $x^2 \equiv a \pmod{pq}$ exists. Then, since $p \mid pq$ and $q \mid pq$ we must have

$$x^2 \equiv a \pmod{p} \text{ and } x^2 \equiv a \pmod{q}$$

Now, through the proof for problem 1.1 above we know there exists distinct

$$j, -j \in \mathbb{Z}/p \text{ and } f, -f \in \mathbb{Z}/q$$

satisfying the above respective equations. Moreover, we know these are the only solutions. So, $x \equiv \pm j \pmod{p}$ and $x \equiv \pm f \pmod{q}$. Using Chinese Remainder Theorem, we see

$$x_1 = \begin{cases} x_1 \equiv j \pmod{p} \\ x_1 \equiv f \pmod{q} \end{cases} \qquad x_2 = \begin{cases} x_2 \equiv -j \pmod{p} \\ x_2 \equiv f \pmod{q} \end{cases}$$

$$x_3 = \begin{cases} x_3 \equiv j \pmod{p} \\ x_3 \equiv -f \pmod{q} \end{cases} \qquad x_4 = \begin{cases} x_4 \equiv -j \pmod{p} \\ x_4 \equiv -f \pmod{q} \end{cases}$$

are all possible solutions and in fact the only solutions by (1.1).
Notice, again since $p \mid pq$ and $q \mid pq$

$$\begin{aligned} x_1 \not\equiv x_2 \pmod{p} &\implies x_1 \not\equiv x_2 \pmod{pq} \\ x_2 \not\equiv x_3 \pmod{q} &\implies x_2 \not\equiv x_3 \pmod{pq} \\ x_3 \not\equiv x_4 \pmod{p} &\implies x_3 \not\equiv x_4 \pmod{pq} \end{aligned}$$

so

$$x_1 \not\equiv x_2 \not\equiv x_3 \not\equiv x_4$$

and our four solutions are distinct. □

## Problem 3

Describe a polynomial-time algorithm that finds all solutions to the congruence

$$x^2 \equiv 1 \pmod{pq}$$

## Solution

*Proof.* To solve for the four solutions to $x^2 \equiv 1 \pmod{pq}$, let us first consider

$$x^2 \equiv 1 \iff x^2 - 1 \equiv 0 \iff (x-1)(x+1) \equiv 0 \pmod{n}, n \in p, q$$

It is clear to see 1, $p-1 \in \mathbb{Z}/p$ and 1, $q-1 \in \mathbb{Z}/q$ are solutions.

Notice,

$$1^2 \equiv 1 \pmod{pq} \text{ and } (pq-1)^2 \equiv (-1)^2 \equiv 1 \pmod{pq}$$

So only two pairs remain.

Let $CRT(x_1, x_2)$ define a function executing the Chinese Remainder Theorem on the equivalences $x \equiv x_1 \mod q$ and $x \equiv x_2 \mod p$.

Tt is clear to see

$$CRT(1,1) = 1 \text{ and } CRT(q-1, p-1) = pq - 1$$

so, only $CRT(q-1, 1)$ and $CRT(1, p-1)$ remain. Of course, both of these operations terminate and give the correct answers.

Notice, the CRT consists of:

The Extended Euclidean Algorithm (O(logn))
And several operations (O(1))

So the time complexity of CRT is O(logn).
Thus, the time complexity for our algorithm is 3 operations (for calculating $p-1, q-1$, and, $pq-1$) + O(logn) (for CRT(1,p-1) and CRT(1,q-1)) $\leq$ polynomial time. □

## Problem 4

Describe a polynomial-time algorithm that given a natural number $N$ of the form $N = pq$ (where $p$ and $q$ are primes) and four elements of $a_1, a_2, a_3, a_4 \in \mathbb{Z}/N$ such that $a_i^2 \equiv 1 \pmod{pq}$.

## Solution

*Proof.* Since we know $1 \equiv 1^2 \equiv (-1)^2 \equiv (pq-1)^2 \pmod{pq}$, we can easily check and get rid of the $a_i$ equal to 1 and pq-1.
Then, without loss of generality, we let

$$a_1 \equiv -1 \equiv (p-1) \pmod{p}$$

We know this is one of the solutions because of our work in 1.3. Now $a_1 \equiv (p-1) \pmod{pq}$ since $p|pq$. Then take

$$gcd(pq, a+1 = mp - 1 + 1 = mp) = p$$

Finally divide pq by p to get q. We have now found our two primes.

Checking $a_i = 1$ and $a_i = pq - 1 = N - 1$ are $\leq$ O(4) each and hence O(1) time complexity. Then, the operation $a_1 + 1$ is O(1) time complexity. Finally, computing gcd($pq$, $a+1$) has O(logn) time complexity and dividing pq by q is O(1) time complexity. Therefore the total time complexity is

$$O(1) + O(1) + O(1) + O(logn) + O(1) = O(logn)$$

so this algorithm runs in polynomial time. Of course, this algorithm terminates because the gcd algorithm terminates. □

## Program 1

```
#returns gcd(N,a), as well as x,y such that a*x + N*y = gcd(N,a)
def eea(N,a):
    #when remainder of previous call is zero, then a*x will always be 0 mod N
    if a == 0:
        return (N, 0, 1)
    #when remainder
    else:
        # find N/a and remainder
        d = N // a
        r = N % a

        #recursive call
        gcd_result, x, y = eea(a, r)

        x_next,y_next = y-(d*x), x
        #mod inverse is (x_next % N)
        return (gcd_result, x_next, y_next)

# a and m are lists [a_1,a_2,...,a_i], and [m_1,m_2,...,m_i]
# finds x that is congruent to a_1 mod m_1, a_2 mod m_2,..., a_i mod m_i
def crt(a, m):
    sum = 0
    product_m = 1

    # multiply all elements of m together
    for m_i in m:
        product_m *= m_i

    for m_i, a_i in zip(m, a):
        p = product_m // m_i
        gcd, mul_inverse,y = eea(m_i,p)
        sum += a_i * (mul_inverse % m_i) * p
    return sum % product_m

def solve1(p, q):
    pq = p*q
    sol_1 = crt([1, q-1], [p,q])
    sol_2 = crt([p-1, 1], [p,q])
    solutions = (1, pq-1, sol_1, sol_2)
    print(solutions)
```

**Output**

input: solve1(32416187893, 4294968199)
output: (1, 13922649613324381 4706, 107209790882217346430, 32016705251026468277)

input: solve1(32416189381, 4294969207)
output: (1, 13922653519967539 0866, 390812463894029 01520, 100145288810272489347)

input: solve1(32416188223, 4294967357)
output: (1, 13922647025615283 6610, 527101929999709 95072, 865162772561818 41539)

## Program 2

```
#returns gcd(N,a), as well as x,y such that a*x + N*y = gcd(N,a)
def eea(N,a):
    #when remainder of previous call is zero, then a*x will always be 0 mod N
    if a == 0:
        return (N, 0, 1)
    #when remainder
    else:
        # find N/a and remainder
        d = N // a
        r = N % a

        #recursive call
        gcd_result, x, y = eea(a, r)

        x_next,y_next = y-(d*x), x
        #mod inverse is (x_next % N)
        return (gcd_result, x_next, y_next)

def solve2(N,a):
    #this will be elements of a that aren't 1 or pq-1
    a_new = []
    for i in a:
        if not (i==1 or i==(N-1)): a_new.append(i)
    gcd,x,y = eea(N,a_new[0]+1)
    solutions = ((N//gcd), gcd)
    print(solutions)
```

### Output

input: solve2(147573981718145935357, (1, 147573981718145935356, 65452501736166652927, 82121479981979282430))
output: (17179870397, 8589935681)

input: solve2(14757400717871546326 3, (1, 14757400717871546326 2, 7174367118247926980, 140399640060467536283))
output: (17179871549, 8589936587)

input: solve2(14757400887952279437 7, (1, 14757400887952279437 6, 90093287053582930296, 57480721825939864081))
output: (17179871603, 8589936659)