Assignment #3

Matthew Langlois - 7731813

Nov 14

Question 1

Using the following properties it becomes trivial to compute $\phi(n)$:

1) $\phi(mn) = \phi(m)\phi(n)$

2) $\phi(p^i) = p^i - p^{i-1}$

3) $\phi(p) = p - 1$ if p is prime

Using the above properties we can solve the for any $\phi(n)$:

a) The divisors of 63 are: 1, 3, 7, 9, 21, 63. With this information, using property 1, it is possible to determine that $\phi(63) = \phi(7)\phi(9)$. Using property 3 we can determine $\phi(7) = 7-1 = 6$. Next, using property 2, we can determine that $\phi(9) = \phi(3^2) = 3^2 - 3 = 6$. Bringing this all back together we get $\phi(63) = \phi(7)\phi(9) = 6 \cdot 6 = 36$.

   $\therefore \phi(63) = 36$

b) The divisors of 246 are: 1, 2, 3, 6, 41, 82, 123, 246. Using property 1 we can break this down into $\phi(246) = \phi(41)\phi(6)$. Since 41 is prime we can use rule 3 to determine $\phi(41) = 41 - 1 = 40$. Using rule 1 followed by rule 3 we can determine $\phi(6) = \phi(2)\phi(3) = (2-1)(3-1) = 2$. Bringing this all back together we get $\phi(246) = \phi(41)\phi(6) = 40 \cdot 2 = 80$.

   $\therefore \phi(246) = 80$.

c) The divisors of 280 are: 1, 2, 4, 5, 7, 8, 10, 14, 20, 28, 35, 40, 56, 70, 140, 280. With this information, using property 1, it is possible to determine that $\phi(280) = \phi(35)\phi(8)$. Repeating with property 1 followed by using property 3, $\phi(35) = \phi(7)\phi(5) = (7-1)(5-1) = 24$. Next, using property 2, we can determine that $\phi(8) = \phi(2^3) = 2^3 - 2^2 = 4$. Bringing this all back together we get $\phi(280) = \phi(35)\phi(8) = 24 \cdot 4 = 96$.

   $\therefore \phi(280) = 96$

d) The divisors of 4220 are: 1, 2, 4, 5, 10, 20, 211, 422, 844, 1055, 2110, 4220. With this information, using property 1, it is possible to determine that $\phi(4220) = \phi(211)\phi(20)$. Using property 3 we can determine that $\phi(211) = 210$. Next, using property 1 we can determine that $\phi(20) = \phi(5)\phi(4)$. From this we can determine that, using property 3,

$\phi(5) = 5 - 1 = 4$ and, using property 2, $\phi(4) = \phi(2^2) = 2^2 - 2 = 2$. Thus $\phi(20) = 2 \cdot 4 = 8$. Bringing this all back together we get $\phi(4220) = \phi(211)\phi(20) = 210 \cdot 8 = 1680$.

$\therefore \phi(4220) = 1680$

Question 2

Node JS implementation of the Elgamal algorithm.

To run:

1. node q2.js
2. genkeys <- generates a private and public key
3. encrypt <YA> <message> <- generates C1 and C2
4. decrypt <XA> <C1> <C2> <- outputs the decrypted number

```
const readline = require('readline');
const process = require('process');
const q = 67;
const alpha = 12;

const commands = {
    encrypt: (Y_A, message) => {

        message = parseInt(message);
        Y_A = parseInt(Y_A);

        if (message > q) {
            console.log(`Message too big`);
            return;
        }

        let k = rand(1, q-1)
        let K = Math.pow(Y_A, k) % q;

        let C_1 = Math.pow(alpha, k) % q;
        let C_2 = (K * message) % q;

        console.log(`Encryption: (${C_1}, ${C_2})`);
        console.log(`----------`);
    },

    decrypt: (X_A, C_1, C_2) => {
```

```
        C_1 = parseInt(C_1);
        C_2 = parseInt(C_2);
        let K = Math.pow(C_1, X_A) % q;
        let M = (C_2 * modInverse(K, q)) % q;
        console.log(`Your message is: ${M}`);
        console.log(`----------`);
    },

    genkeys: () => {
        let X_A = rand(1, q-1);
        let Y_A = Math.pow(alpha, X_A) % q;
        console.log(`Private Key: ${X_A}`)
        console.log(`Public key: (${q}, ${alpha}, ${Y_A})`);
        console.log(`----------`);
    }

};

let rand = (min, max) => {
  min = Math.ceil(min);
  max = Math.floor(max);
  return Math.floor(Math.random() * (max - min + 1)) + min;
}

let modInverse = (num, mod) => {
    num %= mod;
    for (let x = 1; x < mod; x++) {
        if ((num*x)%mod == 1) {
            return x;
        }
    }
}

readline.createInterface({
    input: process.stdin,
    output: process.stdout,
}).on('line', line => {
    const [command, ...args] = line.split(' ');
    const action = commands[command];
    if (!action) {
        console.error(`Invalid command: ${command}`);
        return;
    }
    action.apply(null, args);
});
```

When $X_a = 1$ and $X_A = q - 1$ both of their C_1 values are the same. Furthermore in the case of 1, K is also equal to 1. And in the case of $q - 1$ K is $q - 2$. Thus a range is specified to ensure that these values aren't always used otherwise it would become trivial to derive they keys used in encryption (also why k introduces randomness).

Elgamal's secruity relies difficult logarithms to compute if the values of the keys are unknown. In this case alpha is a primitive root of 67, thus causing brute force calculation to be difficult. Thus since this property is true the algorithm will protect all values of M equally such that $0 \leq M \leq (q - 1)$.

Question 3

a) 277:

Factor: $N = 277 - 1$
Find $N - 1 = 2^k \cdot m$: $276 = 2^2 \cdot 69$
Using $a = 2$ compute $b_k = (a^m)^{2^k} \mod 277$ until b_{k-1} or it is congruent to 1/-1

$b_0 : (2^{69})^{2^0} \equiv 60 \mod 277$

$b_1 : (2^{69})^{2^1} \equiv 276 \mod 277$

$\therefore$ 277 is likely prime since b_1 is congruent to -1.

279:

Factor: $N = 279 - 1$
Find $N - 1 = 2^k \cdot m$: $278 = 2^1 \cdot 139$
Using $a = 2$ compute $b_k = (a^m)^{2^k} \mod 279$ until b_{k-1} or it is congruent to 1/-1

$b_0 : (2^{139})^1 \equiv 47 \mod 279$

$\therefore$ 279 is defiantly not prime since b_0 is composite.

281:

Factor: $N = 281 - 1$
Find $N - 1 = 2^k \cdot m$: $280 = 2^3 \cdot 35$
Using $a = 2$ compute $b_k = (a^m)^{2^k} \mod 281$ until b_{k-1} or it is congruent to 1/-1

$b_0 : (2^{35})^{2^0} \equiv 280 \mod 281$

$\therefore$ 281 is probably prime since b_0 is congruent to -1.

283:

Factor: $N = 283 - 1$
Find $N - 1 = 2^k \cdot m$: $282 = 2^1 \cdot 141$
Using $a = 2$ compute $b_k = (a^m)^{2^k} \mod 283$ until b_{k-1} or it is congruent to 1/-1

$b_0 : (2^{141})^{2^0} \equiv 282 \mod 283$

$\therefore$ 281 is probably prime since b_0 is congruent to -1.

285:

Factor: $N = 285 - 1$
Find $N - 1 = 2^k \cdot m$: $284 = 2^2 \cdot 71$
Using $a = 2$ compute $b_k = (a^m)^{2^k} \mod 285$ until b_{k-1} or it is congruent to 1/-1

$b_0 : (2^{71})^{2^0} \equiv 143 \mod 283$

$b_1 : (2^{71})^{2^1} \equiv 214 \mod 283$

$\therefore$ 285 is defiantly not prime since b_1 is composite.

287:

Factor: $N = 287 - 1$
Find $N - 1 = 2^k \cdot m$: $286 = 2^1 \cdot 143$
Using $a = 2$ compute $b_k = (a^m)^{2^k} \mod 287$ until b_{k-1} or it is congruent to 1/-1

$b_0 : (2^{143})^{2^0} \equiv 172 \mod 283$

$\therefore$ 287 is defiantly not prime since b_0 is composite.

b) Encryption using the following values:
n: 78391
p: 277
q: 283
M: 2
e: 41

$M^e \mod n = c$ so $2^{41} \mod 78391 = 21026$

$\therefore$ the encrypted message is 21026.

c) From the values above we can compute m using the totent function:
m: $\phi(n) = \phi(277)\phi(283) = (277-1)(276-1) = 77832$
d: $41^{-1} \mod 77832 = 72137$

Now that we have solved for d and m we are able to compute the private and public keys.

$\therefore$ the private key is (72137, 277, 283) and the public key is (41, 78391).

d) Decryption (without then with Chinese remainder theorem)

i) Without the Chinese remainder theorem decryption is simply just: $c^d \mod n = M$ or $21026^{72137} \mod 78391 = M = 2$. This can be done on any powerful calculator, though the Chinese remainder theorem makes it much easier to compute (as seen below).

ii) $d_p : d \mod (p-1) = 72137 \mod 276 = 101$
$d_q : d \mod (q-1) = 72137 \mod 282 = 227$
$q_{inv} : q^{-1} \mod p = 277^{-1} \mod 283 = 47$
$c_p : c \mod p = 21026 \mod 277 = 251$
$c_q : c \mod q = 21026 \mod 283 = 84$
$m_1 : (C_p)^{d_p} \mod p = 251^{101} \mod 277 = 2$
$m_2 : (C_q)^{d_p} \mod p = 84^{227} \mod 283 = 2$
$h : q_{inv} \cdot (m_1 - m_2) \mod p = 47 \cdot (2-2) \mod 277 = 0$
$m : m_2 + h \cdot q = 2 + 0 \cdot 283$

$\therefore$ Using the Chinese remainder theorem we get the message as 2.