Shweta Prasad April 2, 2021

CS 2362 Problem Set 3

Collaborators: *Saloni Mehta*

Problem 3-1

(a) When p is prime, all elements in $\mathbb{Z}_p - \{0\} = \{1, 2, \cdots, p-1\}$ are non-divisors of p except for 1. Consequently, they are all relatively prime to p. Division by 1 does not give a zero remainder, and for prime numbers, the first non-zero number that will give a zero remainder when used as divisor, is itself.

(b) A zero divisor in $\mathbb{Z}_{39}$ is 3, as $3 \cdot 13 = 39 \equiv 0 \mod 39$.

(c) The value of the first element that repeats in the given sequence is 5, as both $5^1 \mod 39$ and $5^5 \mod 39$ give the value 5.

(d) No it is not possible to find a non-zero number x in Z_39 and a number $k \geq 0$ that satisfy the given condition.

We can write

$$x^k \equiv 0 \mod 39 = 39 \times m = 3 \cdot 13 \times m$$

This is true only if $x \equiv 0 \mod 39$ is true. Therefore, from the contrapositive, we get that no nonzero number x that satisfies the given condition can exist.

The answer wouldn't change for an RSA modulus, as has been shown by the decomposition above.

(e) Yes, such numbers can be found. Take the example where $n = 8, x = 2, k = 3$. $2^3 \equiv 0 \mod 8$.

Problem 3-2

In the next iteration, the remainder becomes 0, so we look at iteration 11 for our answer. We get one solution as

$$539 \cdot 597 - 1387.232 = -1$$

i	r_i	u_i	v_i	q_i
1	539	1	0	
2	-1387	0	1	
3	-848	1	1	-1
4	-539	-1	0	1
5	-309	2	1	1
6	-230	-3	-1	1
7	-79	5	2	1
8	-72	-13	-5	2
9	-7	18	7	1
10	-2	-193	-75	10
11	-1	597	232	3

Where $x = 597$ and $y = 232$

Problem 3-3

1. To show that G_n is a multiplicative subgroup, elements in the subgroup must be closed under modular multiplication, must have the property of associativity, must have an identity element and every element must have a modular multiplicative inverse.

   - The property of associativity is transitive, so any subgroup of Z_n^* will also inherit the property of associativity.
   - An identity element exists: for $a = 0$, $an + 1 \mod n^2 \equiv 1$ and any element multiplied with this will give itself.
   - Closure: Consider two elements $an + 1, bn + 1$ for some $a, b \epsilon \{0, 1, ..., n - 1\}$ which belong to the subgroup.

     $$(an + 1)(bn + 1) \equiv (a + b)n + 1 \mod n^2$$

     We have two cases for $(a + b)$:
     - If $(a + b)$ falls inside $\{0, 1, 2, ...n - 1\}$: Then closure is satisfied.
     - If $(a + b) > n$, then $(a + b) = (n + i)$ for some $i \epsilon \{0, 1, 2, .., n - 1\}$. Then,

       $$(a + b)n + 1 \mod n^2 = (n + i)n + 1 \mod n^2 = (in + 1) \mod n^2$$

       which clearly falls in the group.
       Thus, closure is satisfied.

- Existence of inverse: For every $(an+1)$ in the group, let's assume that a multiplicative inverse $(a^{-1}n+1)$ exists. We have

$$(an+1)(a^{-1}n+1) \equiv (a+a^{-1})n+1 \equiv 1 \mod n^2$$

$$\implies (a+a^{-1})n \equiv 0 \mod n^2$$

For an inverse to exist, they need to belong in the range of possible values for a.

$$\implies (a+a^{-1}) \equiv n \mod n^2$$

Therefore the inverse elements are $a^{-1} = n - a$.

2. Using trial and error, it can be found that all elements in the subgroup such that a, n are coprime are generators. Generators are those elements whose powers are able to 'generate' all the elements of the group. In other words, the generator has an order of n.

Let $an+1$ be a generator.
then $(an+1)^2 = (2an+1)$
$(an+1)^3 = (3an+1)$
$(an+1)^4 = (4an+1)$
$(an+1)^5 = (5an+1)$ and so on till $(an+1)^k = (kan+1)$

For this to be true, ka must belong to $[0, n-1]$ which is equivalent to saying that k is a generator of Z_n. This implies that $k\epsilon\ Z_n^*$. Therefore, all elements of the subgroup such that it belongs to Z_n^* will be generators.

Problem 3-4

Very briefly, this is the test we will be doing:
g is a primitive root of $p = 761$ if $g^{\frac{s}{f_i}} \mod p \neq 1\ \forall\ i\ \epsilon\{1, 2..\}$, where $s = p-1$ (since p is prime), f_i are the prime factors of s.
For $p = 761$, $s = 761 - 1 = 760$. The prime factors of s are $2, 5, 19$. Thus, the powers we need to raise g to to test whether they are primitive roots or not can be reduced to the set $\{760/2 = 380,\ 760/5 = 152,\ 760/19 = 40\}$.
Now we try by hand $g = 1, 2, ...$ till we find a g that satisfies the conditions listed above. All calculations below are in mod 761

- Let $g = 2$.
  $2^{380} = 1$
  No point looking further with $g = 2$.

- Let $g = 3$
  $3^{380} = 760$
  $3^{152} = 1$
  No point looking further with $g = 3$.
- Let $g = 4$
  $4^{380} = 1$
  No point looking further with $g = 4$.
- Let $g = 5$
  $5^{380} = 1$
  No point looking further with $g = 5$.
- Let $g = 6$
  $6^{380} = 760$
  $6^{152} = 67$
  $6^{40} = 498$
  We have found a primitive root in $g = 6$.

Therefore, we found $g = 6$ such that it is a primitive root for $p = 761$. Additionally, we also found $g = 2, 3, 4, 5 \; \epsilon \; \mathbb{Z}_p^*$ such that they are not primitive roots of p.

Problem 3-5

1. Alice's public key is $(N, e) = (2038667, 103)$.

   Bob can use this public key to encrypt his message to Alice. The encryption is done as follows. If c is the ciphertext and m is the message, $c = m^e \mod N$.

   Substituting the values accordingly, we get $c = 892383^{103} \mod 2038667 = 45293$.

2. Alice knows that $p = 1301$ is one of the factors of her modulus N. To decrypt received messages, Alice must know her private key (N, d) such that $e \cdot d \equiv 1 \mod \Phi(N)$. Dividing N by p, we get the other factor $q = 1567$. Therefore, we can calculate $\Phi(N) = (p-1)(q-1) = 1300 \times 1566 = 2035800$.

   The modular inverse of e is $103^{-1} \mod 2035800 = 810367 = d$.

3. Alice receives $c = 317730$ from Bob. To decrypt, Alice uses her private key as follows. Decryption $D = c^d \mod N = 317730^{810367} \mod 2038667 = 514407$.

Problem 3-6

The program corresponding to this question is named `factor.py` and can be found in the submitted folder. Run it on the terminal using the command `python factor.py` and you will get the output listing p, q and the confirmation of correctness. In case you wish to test the program with other (n, e, d), please make the changes within the program itself, where the call to the `factor()` function is being made.

The factors of n in this particular case is (WLOG) $p = 79$ and $q = 19$.

Problem 3-7

README:

To run the program, you must run the following commands depending on what you wish to do:

```
python keygenerator.py
python encrypt.py
python decrypt.py
```

Prime generation has been done by randomly producing strings of desired bit length and putting them through the Miller Rabin Primality test.

Although the requirement is to run correctly with 512 bit primes, I have been unable to debug the code to be able to do that. Consequently, I have generated 25 bit primes to show that the program works for small numbers. For larger numbers, only some key pairs seem to be working in properly decrypting the file, so the problem must be with the key generation part, however, on much agonizing exploration, I have still not been able to figure it out.

The extended Euclidean algorithm has been used to calculate modular inverses, and the Square and Multiply algorithm has been used to do modular exponentiation.

For small prime generation, the program ran at 207 ms, 210 ms, 192 ms, 219 ms in the four trials, and ran in an avg 207 ms.

The program did not run properly for large primes :(.