# Intro to Cryptography

Mark Anderson
Homework 2

March 6, 2019

1. Answer the following questions as concisely as you can. You must derive all answers using only the most basic facts about groups and not some very powerful theorems

   - Let G be a group that satisfies the following, prove G is abelian

$$g^2 = e \forall g \in G$$

$$Let\ a, b \in G \rightarrow ab \in G$$

$$\begin{aligned}
(ab)^2 &= e \\
(ab)(ab) &= e \\
abab &= e \\
a * abab &= a * e \\
bab &= a * e && \text{(aa = e by definition)} \\
bab &= a && \text{(a*e = a by identity)} \\
bab * b &= a * b \\
ba &= a * b && \text{(bb = e by definition)} \\
ba &= ab
\end{aligned}$$

   $\therefore$ *G is abelian*

   - Let G be a group that satisfies the following, prove G is abelian

$$Let\ a, b \in G$$

$$(ab)^2 = a^2 * b^2$$

$$\begin{aligned}
(ab)^2 &= a^2 * b^2 \\
(ab)(ab) &= (a^2)(b^2) \\
abab &= aabb \\
a^{-1} * abab &= a * a^{-1}abb \\
ebab * b^{-1} &= eabb * b^{-1} \\
ebae &= eabe \\
ba &= ab
\end{aligned}$$

- If G is a group with only 4 Elements, prove that is must be Abelian We know that any element in the group must divide the order of the group, if our group only has 4 elements then there is only 2 possibilites:
  - G contains an element of order 4 - If G contains an element of order 4 then $\exists g \in G \; s.t. \; |g| = |G|$ which means G is cyclic, and if G is cyclic G is abelian $g^i g^j = g^{i+j\%ord(G)} = g^j g^i$
  - G contains only elements of order 2. If G contains only elements of at least order 2. We can build G iteratively, logically adding one element at a time, such that the order of the element is less than or equal to two. This leads us initially to the set $(e, a_1, a_2)$ we need to add another element, call it $a_1 a_2$, if this element is added then $a_2 a_1$ must also be in the group, and if both of these elements are in the group the current order of the group is 5, so one of these elements must be equal to each other, that is $a_1 a_2 = a_2 a_1$ and is abelian.
- If G is a group of even order, prove that the following is true, you may not assume G is abelian $\exists g \in G s.t. \; g^2 = e$ We will solve this by using a contradictory proof. If we assume that there is no $g \in G \; s.t. \; g^2 = e$ i.e. there is no element in G that is its own inverse. Then we know that $\exists g^{-1} \in G \; s.t. \; g * g^{-1} = e$. We know this by the definition of a group, that there is an inverse element for all elements in the group. If the group is of even order then there is an even amount of these pairs of elements $g * g^{-1}$ and since we know that e is the identity and has no inverse element, the group contains an even amount of these pairs, plus, the identity element which is equivalent to 2n + 1 which is odd. Therefore, we can tell by contradiction that if the group is of even order, it contains the given element.

2. Let G be the set of all 2x2 matrices that satisfy ad-bc != 0
   - List all Such Matrices $(1001), (1101), (0111), (0110), (1011), (1110)$
   - Prove that these matrices form a group In order to prove that these matrices form a group we need to show that the set is closed, the elements have associativity, the set contains an identity element, and the set contains an inverse element.
     - Associativity Matrix Multiplication can be represented as a composition of two linear transformations defined as $F : R^m \to R^k$ and $G : R^m \to R^k$ and the composition represented by $F * G : R^n \to R^k$ and the composition of functions, that is $f(g(x)) = g(f(x))$ is known to be associative.
     - Identity element - trivially, the Identity Matrix denoted by $I_2$ represented by (1001) is the identity element for this set, and $I_2 \in G$
     - Inverse Element - the inverse of the group is defined as $(abcd)^{-1} = \frac{1}{det(abcd)} * (d(-b)(-c)a)$
   - Determine whether the group is abelian let $m_1, m_2 \in G$ and $m_1 = (abcd)$ and $m_2 = (efgh)$
     $m_1 * m_2 = [(ae + bg), (ab + bh), (ca + dg), (cf + dh)]$
     $m_2 * m_1 = [(ea + fc), (eb + bd), (ga + hc), (gb + hd)]$ therefore matrix multiplication is not commutative because $m_1 \neq m_2$

3.

$$a, b \in R \; define \; f_{a,b} : R \to R \; by \; f_{a,b}(r) = a * r + b. \; Let \; G = \{f_{a,b} | a \neq 0\}$$

- Prove that G is a group under the composition of mappings
  - The set is closed under its binary operation, the result remains in the set
    $f_{c,d} * (f_{a,b}) = f_{c,d}(ar + b)$
    $f_{c,d}(ar + b) = a(ar + b) + b$
    $a^2r + ab + bs.t.a \neq 0$
  - The set contains an inverse $f_{a,b}^{-1} = (ar + b)^{-1} = (\frac{r}{a} - \frac{b}{a})$
- Determine whether G is abelian for G to be abelian $f_{a,b} = ar + b == f_{b,a} = br + a$ the restriction on the set is that a is non zero, however there is no such restriction on b. With that we can deduce that there is a case where the 2 functions can never commute, and that is when b = 0 $letb = 0$, $ar + b = ar \; letb = 0 \; br + a = 0 + a = a$ thus, G is not abelian
- H is a subgroup of G if $\forall a, b \in H$
  - if $e \in G, e \in H$ The 2 Groups are trivially different, and the only difference between H and G is the addition of the possibility of a being equal to 0, this does not remove the identity element from G, therefore $e \in H$
  - $\forall a \in H, a^{-1} \in H$ The trivial difference of a being able to be 0 brings on the possibility of the function $f_{a,b}(r) = 0r + b = b$ which still follows with the inverse defined earlier, so H is a Subgroup of G
  - if G is abelian and $H \subset GgHg^{-1} = g * g^{-1}H = H \therefore H$ is a normal subgroup of G

4. Write a python program that takes two polynomials as inputs p(x) and q(x) and outputs two polynomials m(x) and r(x) s.t. $p(x) = m(x) * q(x) + r(x)$ These are the helper functions used in each of the polyGCD and division functions, they are the implementation of polynomial addition and subtraction, as well as a function return the degree of the given polynomial.

---

```
  'degree(poly):
- poly is the input and is a polynomial represented as a list such that
    the first element in the list is the highest degree represented in
    the polynomial
- This function recursively loops through the list and returns the index
    of the first non-zero element in the list, this index reperesents
    the degree of the polynomial
- If no non-zero element in poly is found than the function return 0,
    indicating it is either the [0] polynomial, or the list was
    completely empty
  '
  def degree(poly):
    if poly:
        if poly[0] != 0:
            return len(poly)
```

```
            else:
                return(degree(poly[1:]))
        else:
            return 0
    '
add(P1,P2):
    - P1, and P2 are two polynomials in list notation that are to be added
      together, there are no limitations on size or elements in the
      polynomails
    - The function creates a new polynomials represented as a list that is
      the size of the largest input polynomial, it then loop through all
      the elements of the polynomial on the LHS and assigns them to the
      indeces of the
     newly created polynomial. After this it loops through the second
        polynomial and adds each element to the correct index in the newly
        created polynomial, the result of this a polynomial representing
        the addition
     of the 2 polynomials passed as input
    '

     def add(P1, P2):
         newSize = max(len(P1), len(P2))
         P3 = [0 for i in range(newSize)]
         for i in range(0, len(P1)):
             P3[i] = P1[i]
         for i in range(0, len(P2)):
             P3[i] += P2[i]
         return P3
'
sub(P1,P2):
    - This function takes in 2 polynomials that are the same requirements as
      the polynomials for the add() function
    - This function loops through the elements of the 2 polynomial given,
      and multiplies each element in the list by -1, we do this because A
      - B == A + (-B)
'
     def sub(P1,P2):
         P2 = [i * -1 for i in P2]
         return add(P1, P2)

'
division(dividend, divisor):
    - This function takes in 2 polynomials such that the degree of the
      divisor is not greater than the dividend, because then the result is
      trivial, it also checks to make sure the first element in both
      polynomials is non-zero
     as instructed in the prompt
```

```
    - This function is an implementation of polynomial long division
      similarly to as how it would be done by hand, at each iteration it
      matches the highest degree that can be divided into the dividend,
      and then multiplies the
    result of that operation into a variable called quotient, representing
      the terms successfully divided through, it will then subtract the
      result of this from the dividend and loop again with dividend
      being what wasnt divided
    the function returns 2 lists representing the quotient, and the
      remainder
'

def division(dividend, divisor):
    if not dividend or not divisor:
        print("Error")
        return False

    if degree(divisor) > degree(dividend):
        print("Cannot divide")
        return (0,0)

    degDivisor = degree(divisor)
    degDividend = degree(dividend)
    quotient = []
    counter = 0
    while(degDividend >= degDivisor):
        print(dividend)
        print(divisor)
        newDivisor = copy(divisor)
        shiftValue = degDividend - degDivisor
        for _ in range(shiftValue): newDivisor.append(newDivisor.pop(0))
        val1 = dividend[len(dividend) - degree(dividend)]
        val2 = newDivisor[len(newDivisor) - degree(newDivisor)]
        quotient.append(val1 / val2)
        newDivisor = [elem * quotient[-1] for elem in newDivisor]
        dividend = sub(dividend, newDivisor)
        degDividend = degree(dividend)

    rem = dividend
    return(quotient, rem)

    '
    test():
    is a function that generates X number of cases as specified in the
      function, and generates random polynomials fro degree 10 -> 25 and
      runs them through the division algorithm and through the GCD
```

```
        algorithm.'
def test():
  numCases = 5
  for i in range(numCases):
      a, b = generate_tests()
      print(division(a,b))
      c, d = generate_tests()
      print(poly_gcd(c,d))
```

5. Using your polynomial division algorithm, write a GCD algorithm for polynomials

```
def poly_gcd(a, b):
if all(elem == 0 for elem in b):
    return a
else:
    quot, rem = division(a, b)
    if all(elem == 0 for elem in rem):
        return quot
    else:
        return poly_gcd(b, rem)
```

6. To prove that E is a commutative ring we must first prove that E is a ring. In order for a set to be a ring, it must satisfy the following properties

   - (E,+) is an abelian group closed under addition.
     - (+) is associative: $Let\ a, b, c \in Z \rightarrow 2a, 2b, 2c \in E,\ then(2a + 2b) + 2c = 2(a + b) + 2(c) = 2(a + b + c) = 2(a) + 2(b + c) = 2a + (2b + 2c)$
     - (+) is commutative: $Let\ a, b, c \in Z \rightarrow 2a, 2b, 2c \in E,\ then(2a + 2b) = 2(a + b) = 2(b + a) = 2b + 2a$
     - There is an element: $0 \in E | a + 0 = a\ \forall\ a \in E$ 0 is still 0 in our set of even integers, and functions the same way under the usual addition.
     - Additive Invserse: $Leta \in Z \rightarrow 2a \in E\ 2a + (-2a) = 2a + 2(-a) = 2(a + (-a)) = 2(0) = 0$
     - (E, *) is a monoid under multiplication, except for a multiplicative identity
     - (*) is distributive: $a(b - c) = a(b + (-c)) = ab + a(-c) = ab - ac$

   Therefore E is a commutative ring without identity

7. Let p be a prime, prove that $Z_p$ is a field Let $i \in Z_p\ s.t\ i \neq 0$ if $i \in Z_p$ then we know the gcd(i, p) = 1 because p is prime, and i is non-zero, because of this, p cannot divide i. For i to be in the set $Z_p$ there must exist $x \in Z\ s.t.\ ix = 1\%p$ this is saying that x is a non-zero inverse of i, and since $Z_n$ is already a commutative ring, the only criteria that needs to be fulfilled is that there must exist a non-zero inverse for every element in the set, which we just proved $\therefore$ $Z_p$ is a field

8. Let R and S be two rings and f: R -¿ S a ring homomorphism.

   - Prove that f(0) = 0 by definition of ring homomorphism we know that $f(0_r) = 0_s$
   - Let Ker(f) =
     $$r \in R | f(r) = 0$$
     Prove that Ker(f) is a two-sided ideal of R. let
     $$r \in R \; and \; x \in Ker(f)$$
     Ker(f) is a Left Ideal of R
     $$f(rx) = f(r)f(x) \; because f : R \to S \; is \; a \; ring \; homomorphism$$
     $$f(r) = 0 \; as \; stated \; above \; so \; f(r)f(x) = 0 * f(x) = 0 \therefore rx \in ker(f)$$
     Ker(f) is a Right Ideal of R
     $$f(xr) = f(x)f(r) \; because f : R \to S \; is \; a \; ring \; homomorphism$$
     $$f(r) = 0 \; as \; stated \; above \; so \; f(x)f(r) = f(x) * 0 = 0 \therefore xr \in ker(f)$$
     Ker(f) is a two-sided ideal of R because it is a left and right ideal of R.