MATH 314 Spring 2024 - Class Notes

2/14/2024

Scribe: Rayquan Williams

Summary: Hill Cipher Decryption, Moving Beyond mod 26, Extended Euclid's Algorithm, Encode in Binary

Moving beyond (mod 26)

- Residue - A residue modulo n, is the collection of all integers that have the same remainder (mod n)
- Remainder is between 0 and n
  Ex: The residue 5 (mod 8) is all the numbers [-3,5,13,21,28]
- Z_m For the set of all residues (mod m) Z_6 = [0,1,2,3,4,5]
- Ring - Any collection of things we can add, subtract and multiply with regular arithmetic rules. To divide we must use inverse principles $a^{-1}(a \times a^{-1}) \equiv 1 mod m$ if $gcd(a, m) \equiv 1$

How to compute Euclid's Algorithm (a is dividend, b is divisor, q is quotient, r is remainder)

1. Long divide a by b, always get r smaller than b (a = bq + r)
2. If r divides a and b, d divides (a-bq = r) so d divides r
3. Iterate these steps until r = 0, the previous remainder is the gcd.

Example: gcd(19,62)

62 / 19 = 3 R(5), so 62 = 3(19) + 5
19 / 5 = 3 R(4), so 19 = 3(5) + 4
5 / 4 = 1 R(1), 5= 1(4) + 1
4 / 1 = 4 R(0), so 4 = 1(4) + 0
gcd(19,62) = 1

Extended Euclid's Algorithm (a is dividend, b is divisor, q is quotient, r is remainder)
If gcd(a,b) = d, then there exists x and y so that ax + by = d Steps for Extended Euclid's Algorithm

1. Substitute

2. Distribute

3. Combine like terms

4. Repeat

Example: Find integers x and y so that $19x + 62y = 1$. What is $19^{-1} \pmod{62}$?

Equations:
5 = 62 - 19(3)
4 = 19 - 3(5)
1 = 5 - 4(1)
1= 5-4(1)=5-[19-5(3)](1)=5(4)-19=[62-19(3)](4)-19=62(4)-19(13) x = -13 y = 4
1 = 62(4)-19(13) (mod 62) 1 = 19(49)
19^{-1}= 49

<u>Encode Messages in Binary</u>

ASCII converts strings to numbers A - 65 a - 97
Assume plaintexts are binary strings
2 kinds:
Stream (Like Affine) and Block (Like Hill) Ciphers
For Stream ciphers, setup is to generate a key (binary string) and encrypt it by xoring the key with plaintext
Mod 2 Addition and Subtraction are the same, $1 \equiv -1(mod 2)$ D(x) = $x \oplus k$
Xor results are $0 \oplus 0 = 0, 1 \oplus 0 = 1, 0 \oplus 1 = 1, 1 \oplus 1 = 0$
Example: E(x) = $x \oplus k$, P = 011010 and K = 101010
$x \oplus K = 110000$
Decrypt 110000 using key 101010: $110000 \oplus 101010 = 011010$

<u>One Time Pad</u>

Key is a randomly generated string of 1's and 0's. This key is only used one time E(x) = $x \oplus k$ D(y) = $y \oplus k$
Has "perfect security", every plaintext is equally likely to correspond to any ciphertext
Disadvantages: The key is as long as the plaintext and can only be used one time, transmitting the key securely is as hard as transmitting the message
Random
Random number generators: Linear congruential random number generator (LCRNG) and Linear Feedback Shift Register (LFSR)
LCRNG is used by java.random, Pick a modulus and 2 numbers a and b, use equation $r_i \equiv r_{-1} + b$ (mod m) to generate a string of random numbers
Always start with random seed r_0 Ex: m = 11, a = 5, b = 7, $r_0 = 2$
$r_0 = 2$ $r_1 \equiv 5(2) + 7 = 10 + 7 = 17 = 5$(mod 11)
$r_2 \equiv 5(5) + 7 = 25 + 7 = 32 = 10$(mod 11)

$r_3 \equiv 5(10) + 7 = 50 + 7 = 57 = 2(\text{mod } 11)$ $r_4 = 5$
LFSR is better RNG, seed is a string of k bits and generate new bits using previous k bits, is recursion relation
$b_i \equiv a_1 b_{i-1} + a_2 b_{i-2} + \ldots\ldots\ a_k b_{i-k}$ (mod 2) where $a_1, a_2, \ldots.a_k$ are fixed numbers (mod 2)