**TNE week 8 to 12 useful formulas: (as no lect in 7)**

**Week8:**

1. If there are **“K ”** bits in a key, no of possible keys is **2k**
   1. So, if key bit increased from **“n” to “m”,** find **(m-n)** value, and then find **2(m-n)**to get increase in key strength
2. For symmetric key,

**EncryptAlgokey (Message) = Cipher** and **DecryptAlgokey (Cipher) = Message**.

BUT, **DecryptAlgokey (EncryptAlgokey (Message))** may not be equal to **Message**

1. How many keys does each person need to keep secret? N(N-1)/2

And how many keys in total? N(N)(N-1)/2

1. For Public Key,

**EncryptAlgokey1 (Message) = Cipher** and **DecryptAlgokey2 (Cipher) = Message**.

The key used for encrypt and decrypt must be modulo inverses of one another! They cant be same!

But Key1 and Key2 positions can be interchanged!

**DecryptAlgokey1 (EncryptAlgokey2 (Message))** == **Message**

1. One time pad:

**a XOR b = c**

and **c XOR b = a**

Here a, b,c are binary( or 01) forms of **message** (for a), **OTP** (for b) and **cipher** (for c).

FOR XOR: **0 XOR 0 = 0**

**1 XOR 1 = 0**

**1 XOR 0 = 1**

**0 XOR 1 = 1**

1. S-BOX:

Convert **the 6 bit input** (eg 100001) into **2 outer bits** (eg 11) and **4 inner bits**(eg 0000)

Find the **row that match outer bits** and **Column that match inner bit**

(if needed convert column bits to decimal )

The place the **outputted 4 bit value from the table** as output of S-BOX

**Week9:**

1. Modulo arithmetic breakdown in case calc doesn’t work as too big power:

**Ap+q+rmod N = (Ap mod N \* Aq mod N \* Armod N ) mod N**

1. Diffie Hellman:

Alice secret number: **a**

Alice sends “A” to bob where **A= ga mod p**

Bob secret number: **b**

Bob sends “B” to bob where **B= gb mod p**

Their overall shared secret key S is **S= Ab mod p = Ba mod p**

Or **S = (ga mod p)b mod p**

1. RSA:

Two large **prime numbers**: **p** and **q**

**N = p\*q**

**X = (p-1)\*(q-1)**

Find **an integer e that is relatively prime to X**

**Public key [n,e]**

Choose d, so that **(d\*e)mod X = 1**

**Private key [n,d]**

So, to **encrypt, c = (me) mod N**

To **decrypt, m= (cd) mod N**

Where c= cipher and m = message

To reverse it, or find respective d or e,

factorise n value to get p and q. Then just perform the steps

1. Lehmann’s test for Primality:

Need to find **p is prime or not**

Choose a where **a <p**

Find **a(p-1)/2 mod p**

If **ans is 1 or p-1**, confidence of prime increases

**Confidence of prime = 1 -0.5n**where **n is no of times this has been checked.**

Confidence can be 0.5, 0.75, 0.875,…..etc.

1. Tute week 9: same as week 8 stuff
2. Week 10: No calc
3. Tute week 10: same as week 9 stuff
4. Week11: No calc
5. Tute week 11: same as week 8 and 9 stuff