Lecture - 5

Thursday, 4 August 2016 (17:10- 18:00)

Shannon's One Time Pad- A Perfectly Secure Encryption Technique

We have studied three encryption techniques and their respective cryptanalysis in lecture 3, namely Caesar cipher, substitution cipher and Vigenere cipher. All of these ciphers were seen to be easily cryptanalysed using some intelligence applied in the correct direction. Today we will study an encryption scheme which guarantees perfect secrecy.

Perfect Secrecy: An encryption scheme is said to be perfectly secure, if given a ciphetext, the malicious third party(the party trying to decode the ciphertext without having the key) could not say anything about the plaintext. In other words, given a ciphertext, the probability of every plaintext is equally likely.

One Time Pad: In this lecture, we will discuss one such encryption scheme, called one time pad proposed by Shannon.
In this encryption, the plain text is XORed with the Key to generate the cipher text. An example is shown in Table 1.

Plain Text	1	1	0	1	0	1	1	0	0	0	1	0	1	0	1
Key	0	0	1	0	1	1	1	0	0	1	0	0	0	1	1
Cipher Text	1	1	1	1	1	0	0	0	0	1	1	0	1	1	0

Table 1: One Time Pad Encryption Technique

Given any amount of computation power, this cipher can not be broken, unlike Caesar cipher which can be broken in 26 attempts and substitution cipher which can be broken in 26! steps. Let us see how.

Question: Given that the ciphertext is β, what is the probability of plaintext being α?

Answer: We know that the XOR operation is reversible, i.e. if $A \oplus B = C$, then, $B \oplus C = A$. Therefore, when given a ciphertext β, and any plaintext α, we can generate a key which when applied on α generates β. Hence, given a ciphertext β, the probability that the plaintext is $\alpha = \frac{1}{2^n}$, where n is the length of the plaintext = length of the ciphertext= length of the key.

The problem with this cipher is the key length, which should be equal to the length of the plaintext.
Why can not we use the same key for all the messages?

Consider Bob sending messages to Alice. Bob and Alice have already fixed upon a key k and are using this key for encryption and decryption. Tom is the malicious third party who is getting all the ciphertexts Bob is sending to Alice. Tom have all the m messages Bob has sent to Alice. If the same key was used for the encryption of all the messages, Bob can make out the key as well as the plaintext. So, the process demands a new key for every new plaintext which Bob wants to send. Sending a new key securely everytime when Bob wants to send a message to Alice does not make sense, because in that case, Bob can just send the plaintext securely. There is no need for the key.

How to solve this problem?
This problem can be solved with the help of a protocol that generates a large random number when provided with a seed. Anyone having the seed can then generate the random numbers sequentially. Let us see how.

Linear Congruential generator:n^{th} random number is dependent on previous numbers based on this relation $x_n = a \times x_{n-1} + b \; mod \; c$
where x_0 is a seed value,
a is multiplier, $0 < a < c$
b is increment , $0 \leq b < c$
c is modulus, $0 < m$

When provided with efficient values of a, b and c, this formula generates random numbers ranging from 0 to $c - 1$. Consider the following :
$x_n = 2 \times x_{n-1} + 1 \; mod \; 11$

When seeded with x_0= 3, this pseudo random number generator generates: x_1= 7, x_2= 4, x_3= 9, x_4= 8, x_5= 6, x_6= 2, x_7= 5, x_8= 0, x_9= 1.

What Bob needs to share with Alice is just the parameters of this equation, with a seed value. They can go about sharing a large number of messaged with just one seed. They find out the next large number to be used a key with the help of the above equation.