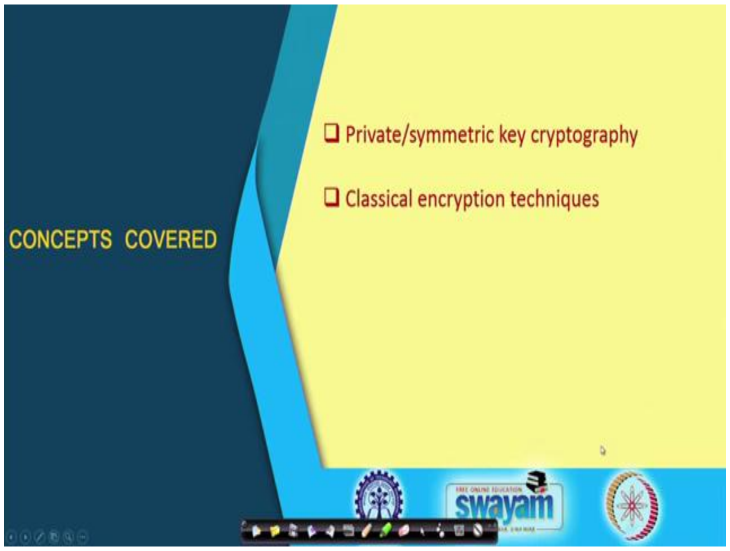
Ethical Hacking
Prof. Indranil Sengupta
Department of Computer Science and Engineering
Indian Institute of Technology, Kharagpur

Lecture – 27
Private – Key Cryptography (Part I)

In the last lecture we talked about some of the security principles and services and we mentioned that encryption/decryption play a very important role in securing some kind of communication or a network in general. So, in this lecture we start with some discussion on private key cryptography techniques, this we shall be discussing in two parts. This is the first part.

(Refer Slide Time: 00:46)

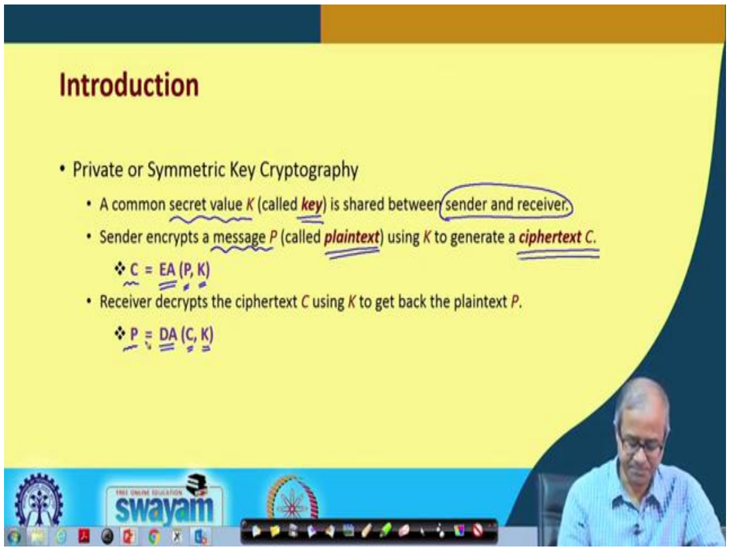


CONCEPTS COVERED

- Private/symmetric key cryptography
- Classical encryption techniques

Now, in this part of the lecture, we shall broadly be discussing some of the private and symmetric key cryptography algorithms, particularly some of the classical techniques. Just recall whatever we will be discussing today in this lecture, the primary objective will be to give you a conceptual idea regarding private key encryption; but the practical algorithms which are really used, that we shall be discussing in our next lecture.

(Refer Slide Time: 01:22)



Introduction

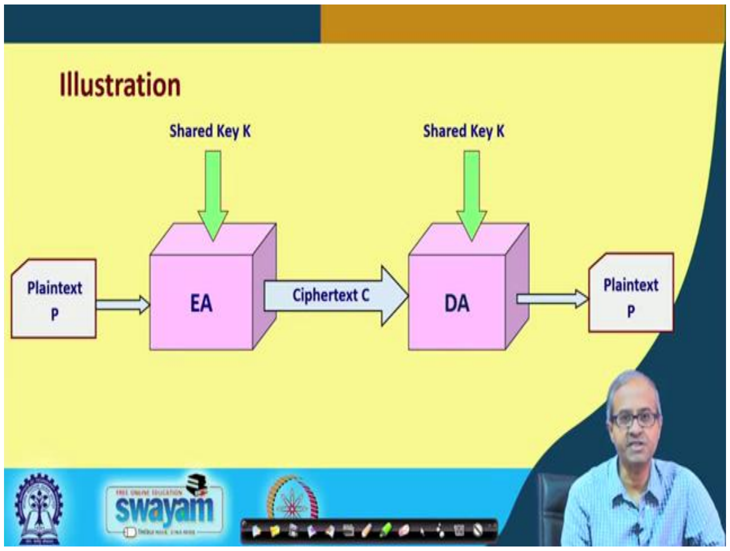
- Private or Symmetric Key Cryptography
    - A common secret value *K* (called ***key***) is shared between sender and receiver.
    - Sender encrypts a message *P* (called ***plaintext***) using *K* to generate a ***ciphertext*** *C*.
        - C = EA (P, K)
    - Receiver decrypts the ciphertext *C* using *K* to get back the plaintext *P*.
        - P = DA (C, K)

Now, talking about private or symmetric key cryptography, the basic concept is like this. Well I have a message P that I want to transmit. This message P in cryptographic terminology is called a plaintext. Now what I try to do, I try to encrypt my plaintext or message into something called ciphertext. Ciphertext is something which is garbled and if an intruder manages to even get hold of it, will not be able to decode what is the actual message that is been flowing through, ok

And in symmetric key cryptography, there is a concept of a secret value called a key and this key is shared between the sender and the receiver, the same key is shared. So, the way encryption and decryption are carried, functionally they are expressed like this; this EA stands for the encryption algorithm, well EA is like a function. It takes two parameters, the plaintext and the key and it generates this ciphertext C, ok. Similarly the decryption algorithm which is run at the receiver end, it will take the ciphertext as input and also the same value of key K and it will get back the original message or plaintext P.

This is conceptually how symmetric key cryptography works. It is called symmetric or private key, because the same key is shared by both the parties, sender and receiver that is why it is called symmetric, or it is private between the two parties that is why it is called private.

(Refer Slide Time: 03:23)

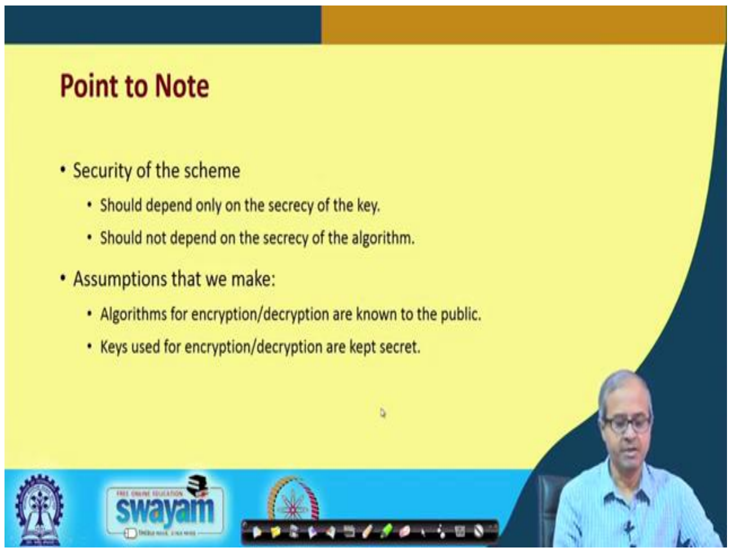


Illustration

Now, pictorially the same process as I have just now said is depicted in this diagram. Now, in this diagram if you see, here is your plaintext, this is the sender side and this is the receiver side. The plaintext goes through the encryption algorithm which also takes as input the shared key K, generates the ciphertext C which flows or is transmitted over the network, it reaches the receiving end. The receiving end runs a decryption algorithm with the same shared key value K, it gets back the plaintext. This is how symmetric key algorithms work.

So, the point to note is that the receiving end unless that party does not have the same value of the key K, will not be able to decode it which means an intruder which will not be having the value of K, well, even if the intruder knows the encryption and decryption algorithms, he or she will not be able to decode the cipher text, because the value of K is not known, ok.

(Refer Slide Time: 04:46)



Point to Note

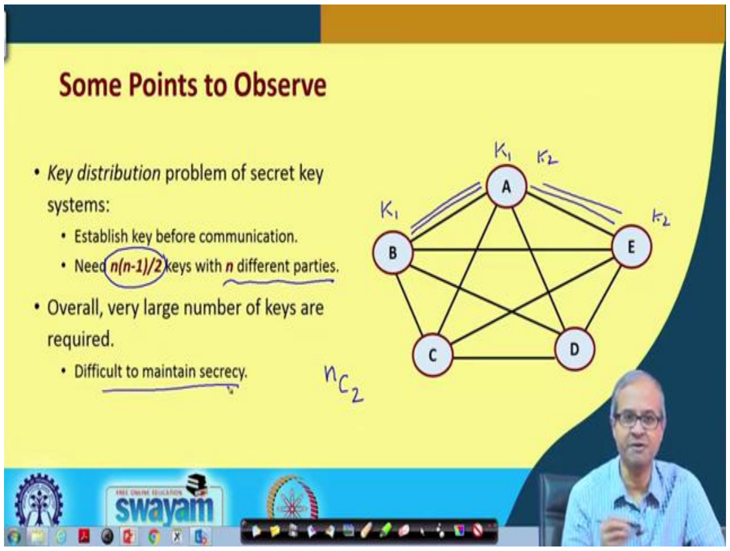
- Security of the scheme
  - Should depend only on the secrecy of the key.
  - Should not depend on the secrecy of the algorithm.
- Assumptions that we make:
  - Algorithms for encryption/decryption are known to the public.
  - Keys used for encryption/decryption are kept secret.

So, the points to note, there are a couple of important points here; first thing is that, for this kind of encryption or decryption techniques it is understood or it is accepted that your method should be such that the overall security of this scheme must not depend on the secrecy of the algorithm. Let the algorithm be known to everybody. The important point is that the security should only depend on the secrecy of the key; as long as the value of K, the key K is only maintained by the sender and receiver means our communication will be very secure, this is the understanding. So, it should not depend on the secrecy of the algorithm.

So, the assumptions in this regard, that are made are, that the algorithms encryption, decryption what we talked about, they are known to the public; but the keys are kept secret. But there are some agencies which are involved in very high security transactions, like for example, the defense forces, they also prefer to keep their algorithm secret; now it is like a double edged sword.

So, if you keep the algorithm secret, maybe the intruder will find it more difficult to try and break your code. But, because you are not making it public. The general experts who are outside that organization will not be getting a chance to analyze that algorithm and identify whether there are any vulnerabilities or weaknesses, ok. It is possible that the algorithm apparently looks very nice, but there are some very important weaknesses in the algorithm, which someone once is known can very easily exploit, fine.

(Refer Slide Time: 06:56)



Some Points to Observe

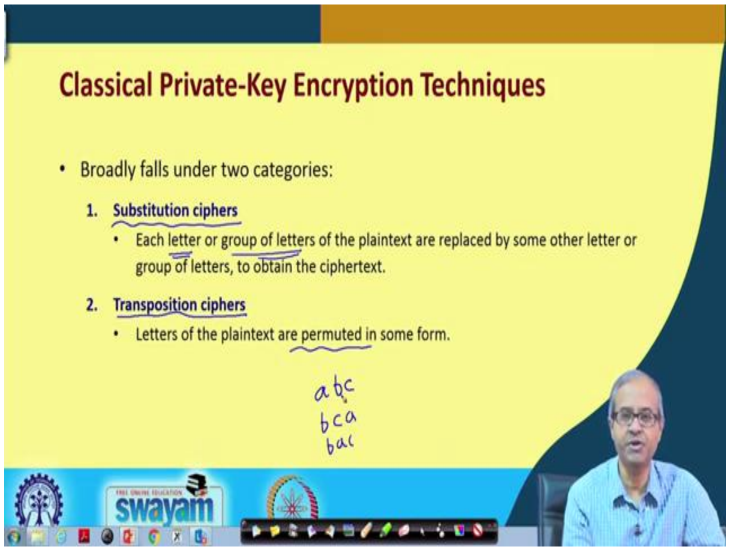
- *Key distribution* problem of secret key systems:
  - Establish key before communication.
  - Need $n(n-1)/2$ keys with n different parties.
- Overall, very large number of keys are required.
  - Difficult to maintain secrecy.

Now, another point here is regarding the number of keys that are being handled. Now you think of a communication system schematic. Here I have shown it as a graph where A B C D E these are five communicating parties and they can communicate among each other, any pair.

So, when A and B are communicating, let us say along this path, so there will be some key which will be shared by A and B. Let us say K_1; when A and E are communicating there will be some other key value K_2, so like this K_3, K_4, K_5, for every pair of sender and receiver there will be a different value of secret key. So, in general for n different parties, the number of distinct keys will be n choose 2, $\binom{n}{2}$ which is nothing but $n \times \frac{n-1}{2}$.

So, as you can understand as the value of n grows, the number of secret keys that the network has to maintain will grow very large; well if n is let us say, 1000, now this value will be $1000 \times \frac{999}{2}$, close to 500000, ok. So, maintaining so many distinct keys in a private way is itself a challenge; this is one problem. So it is rather difficult to maintain secrecy of this large number of keys. This is one problem in this method.

(Refer Slide Time: 08:39)



Classical Private-Key Encryption Techniques

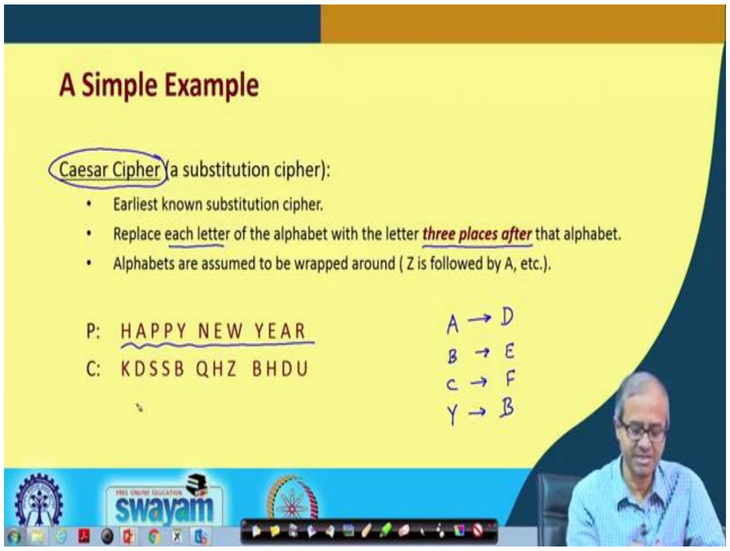
- Broadly falls under two categories:
  1. **Substitution ciphers**
     - Each letter or group of letters of the plaintext are replaced by some other letter or group of letters, to obtain the ciphertext.
  2. **Transposition ciphers**
     - Letters of the plaintext are permuted in some form.

a b c
b c a
b a c

Now let us look at some of the classical private key encryption algorithms. They broadly fall under two categories; well here I am assuming in the classical category that the message that I am transmitting is a simple text message. It is not a non-text binary file. I am transmitting, a text message.

The first category is referred to as substitution cipher, where we substitute each letter or a group of letters in the original plaintext by a different letter or a different group of letters so that my cipher text remains a textual data, but the letters are all garbled up. And in transposition cipher, the letters are not replaced by anything. Just the order of the letters are mixed up; they are permuted, some kind of permutation. Like for example, if my letters are a b c, I can make it b c a or b a c something, this is called permuting the order of the letters.

(Refer Slide Time: 09:56)



A Simple Example

Caesar Cipher (a substitution cipher):

- Earliest known substitution cipher.
- Replace each letter of the alphabet with the letter ***three places after*** that alphabet.
- Alphabets are assumed to be wrapped around (Z is followed by A, etc.).

P: HAPPY NEW YEAR

C: KDSSB QHZ BHDU

$A \rightarrow D$

$B \rightarrow E$

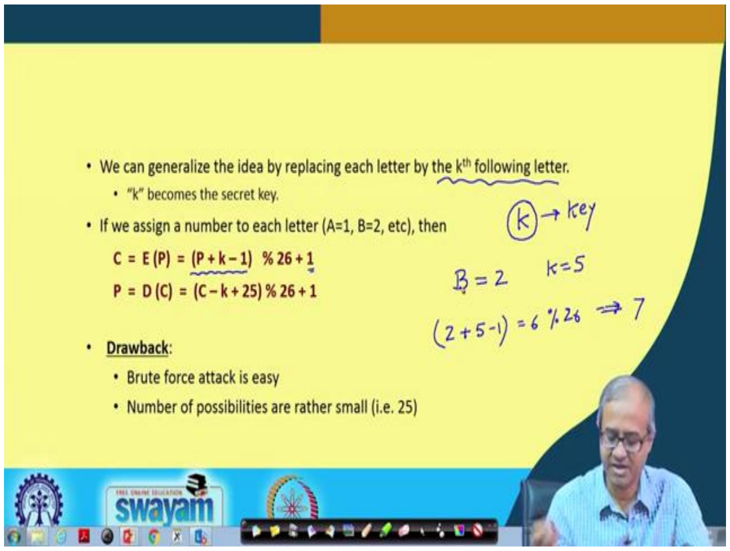
$C \rightarrow F$

$Y \rightarrow B$

So, let us see some of the classical techniques. Well, one of the oldest ciphers or encryption method was referred to as Caesar Cipher. This is essentially a substitution cipher, where every letter or alphabet in the plaintext is replaced by some other alphabet. And the way it works is very simple, it says you replace each letter in your message by the letter which appears three places in the sequence of alphabet; like for example, if your, if you have a letter A, A will be replaced by B C D, A will be replaced by D. Similarly, B will be replaced by E. C will be replaced by F and so on.

Now, here we assume that the alphabets are cyclically rolled or chained like after Z again comes A. So, for example, if my letter in the original plaintext is Y then Z A B this will be replaced by B like this. This is a example is shown here. If my plaintext is HAPPY NEW YEAR then H gets replaced by K, I J K, A by D, P by S, Y by B and so on, ok. Like this, this substitution is carried out, ok. This method is very simple as you can understand, but as such in this method there is no concept of a key. I have a message. I replace every letter by the fourth next letter, third next letter.

(Refer Slide Time: 11:52)



- We can generalize the idea by replacing each letter by the kth following letter.
  - "k" becomes the secret key.
- If we assign a number to each letter (A=1, B=2, etc), then

  C = E (P) = (P + k − 1) % 26 + 1

  P = D (C) = (C − k + 25) % 26 + 1
- **Drawback**:
  - Brute force attack is easy
  - Number of possibilities are rather small (i.e. 25)

$\boxed{k} \rightarrow$ key

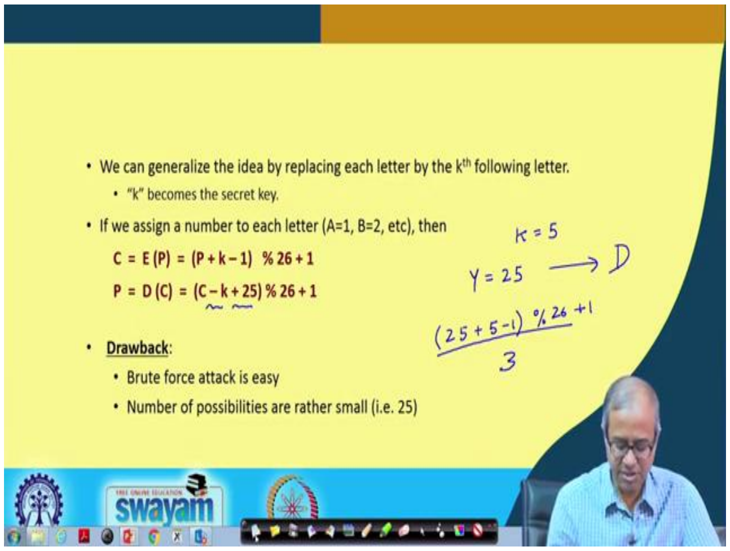
$B = 2 \quad k = 5$

$(2 + 5 - 1) = 6 \% 26 \Rightarrow 7$

Now, you can generalize this Caesar cipher concept to make it dependent on a key as well. So, what we do, we do not replace a letter by the next third letter, but we replace it by the kth next letter. So, now, this value of k, this becomes our key. So, if we number the alphabets A 1, B 2, C 3 up to Z 26; then for each letter the encryption and decryption process will work like this; encryption will be $(P + k - 1)\%26$; % means divided by 26 and take the remainder $+ 1$.

Let us take an example, suppose I want to encrypt B, which in terms of the code, it is 2 and my key value is let us say 5. So, my encryption how would, what will happen? $P + k - 1$, now here P is this B 2, $2 + 5 - 1$, so this is how much 6. So, if you take modulo 26, divided by 26 and take the remainder, so it remains 6; $6 + 1$ is 7, it becomes 7; 7 is A B C D E F G, so, B will be replaced by G.

(Refer Slide Time: 13:36)



- We can generalize the idea by replacing each letter by the kth following letter.
  - "k" becomes the secret key.
- If we assign a number to each letter (A=1, B=2, etc), then

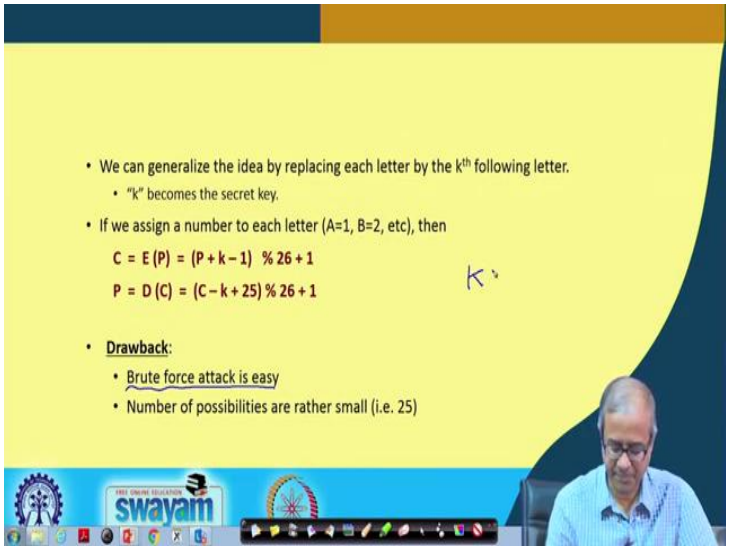
  C = E (P) = (P + k − 1) % 26 + 1

  P = D (C) = (C − k + 25) % 26 + 1
- **Drawback**:
  - Brute force attack is easy
  - Number of possibilities are rather small (i.e. 25)

Let us take another example, suppose again $k = 5$ and I am trying to encrypt the letter Y which is 25. So, now, this will be $(25 + 5 - 1)\% \ 26 + 1$. So, this is $30 - 1$, this is 29; $229\%26$ is 3, divided by 26 and take the remainder, $3 + 1$ is 4, so A B C D; this Y will be replaced by the letter D, ok.

So, decryption is very similar, where you do $-k$ and you do $+25$ here. Now you can check, you can get back the original letter once you have the value of k.

(Refer Slide Time: 14:43)



- We can generalize the idea by replacing each letter by the kth following letter.
  - "k" becomes the secret key.
- If we assign a number to each letter (A=1, B=2, etc), then

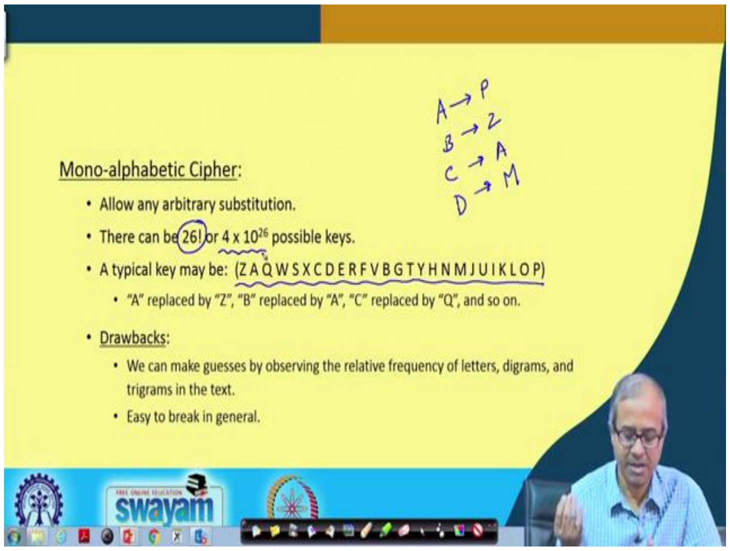
  C = E (P) = (P + k − 1) % 26 + 1

  P = D (C) = (C − k + 25) % 26 + 1
- **Drawback**:
  - Brute force attack is easy
  - Number of possibilities are rather small (i.e. 25)

Now this method apparently is very simple, it should work; but the problem is that, here Brute Force attack is very easy, because we are talking about alphabets; the value of the key, the different values the possibilities are limited only there are 26 possible values, 1 upto 26.

So, you can exhaustively try all values of k and see for which value you get back a meaningful text that will be your key. So, it is very easy to break, right. So, this is not a practical method.

(Refer Slide Time: 15:07)



Mono-alphabetic Cipher:

- Allow any arbitrary substitution.
- There can be 26! or 4 x 10^{26} possible keys.
- A typical key may be: (Z A Q W S X C D E R F V B G T Y H N M J U I K L O P)
    - "A" replaced by "Z", "B" replaced by "A", "C" replaced by "Q", and so on.
- Drawbacks:
    - We can make guesses by observing the relative frequency of letters, digrams, and trigrams in the text.
    - Easy to break in general.

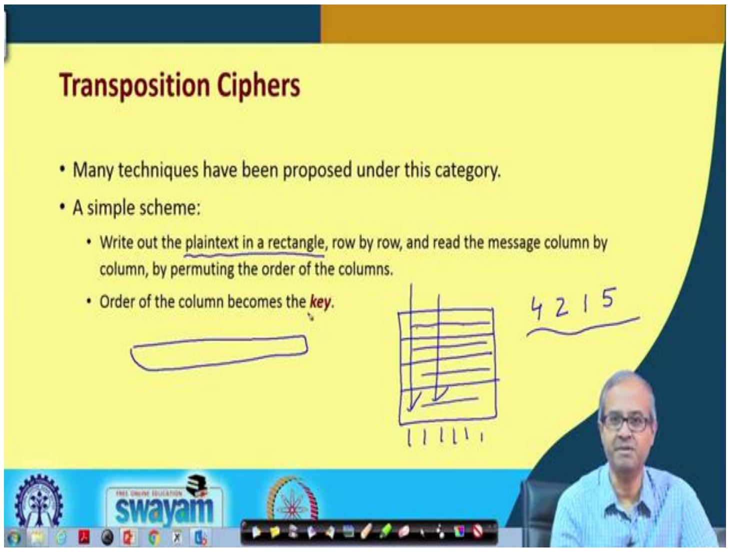
$A \rightarrow P$
$B \rightarrow Z$
$C \rightarrow A$
$D \rightarrow M$

So, to make the process more complex and to increase the number of possibilities in the key, you can go for something called Mono-alphabetic cipher. Mono-alphabetic cipher says, you allow any arbitrary substitution of the letters, not necessarily by the kth next letter. For example, I can say that I will replace A by P, B by Z, C by A, D by M and so on, any arbitrary.

So, if I have this A to Z the string of 26 letters so, I can define a sequence like this. This will mean the first letter A will be replaced by Z, second letter B will be replaced by A, third letter C will be replaced by Q and so on; the last letter Z will be replaced by P. So, I am actually writing down a permutation of the 26 alphabets and that permutation becomes my key, ok. Now; obviously, this will be much more difficult to break, because there is no easy way. And the other thing is that for 26 things, 26 letters in the alphabet, the numbers of possibilities are $26!$. So many permutations are possible, which comes to about 4×10^{26}; well which seems to be a very large number.

But the truth is that, this code is also not that difficult to break; because you can make some guesses. Like you know in normal English text you know some typical frequencies of occurrence, which letter occurs the most frequent and pairs of letters T followed by H is very frequent; and similarly trigrams, three letters coming together, so how frequent are there. So, if you make such a frequency count of all the letters, diagrams and trigrams that appear in your ciphertex, then you can guess which is your A, which is your B, which is your C and like that you can break it. So, using the relative frequencies it is rather easy to break this code.

(Refer Slide Time: 17:45)



Transposition Ciphers

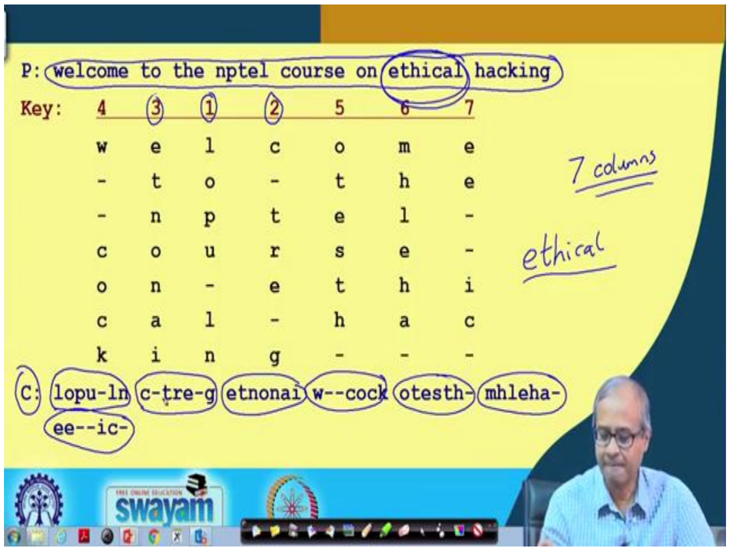
- Many techniques have been proposed under this category.
- A simple scheme:
  - Write out the plaintext in a rectangle, row by row, and read the message column by column, by permuting the order of the columns.
  - Order of the column becomes the ***key***.

Then you have another class of ciphers, where you do not replace a letter by other letter, but change their order. So, one such commonly talked about cipher is called Transposition cipher, ok. The transmission cipher, the idea is like this, suppose I have some message to be encrypted. First thing is that I write out the plaintext in a rectangle; rectangle means, number of letters in a row is fixed. So, I write certain number of letters in a row, then in the next row, then in the next row like this, row wise I write down the letter in a rectangle.

Now, the number of columns in the rectangle that is part of the key, that I will not tell anybody, I share it with the receiver, number of column. And not only that, once I write down the text, for reading I will be reading it column by column; but not in the order of first column followed by second column followed by third column, not like that, maybe I will be reading first the fourth column, then the second column, then the first column, then the fifth column. So, this will also be part of the key. So, if the receiver knows in which order the columns have to be read, then the plaintext can be recovered.

So, the order of the column and of course, the number of columns becomes the key in this case.

(Refer Slide Time: 19:27)



P: welcome to the nptel course on ethical hacking

Key:	4	3	1	2	5	6	7
	w	e	l	c	o	m	e
	-	t	o	-	t	h	e
	-	n	p	t	e	l	-
	c	o	u	r	s	e	-
	o	n	-	e	t	h	i
	c	a	l	-	h	a	c
	k	i	n	g	-	-	-

C: lopu-ln c-tre-g etnonai w--cock otesth- mhleha- ee--ic-

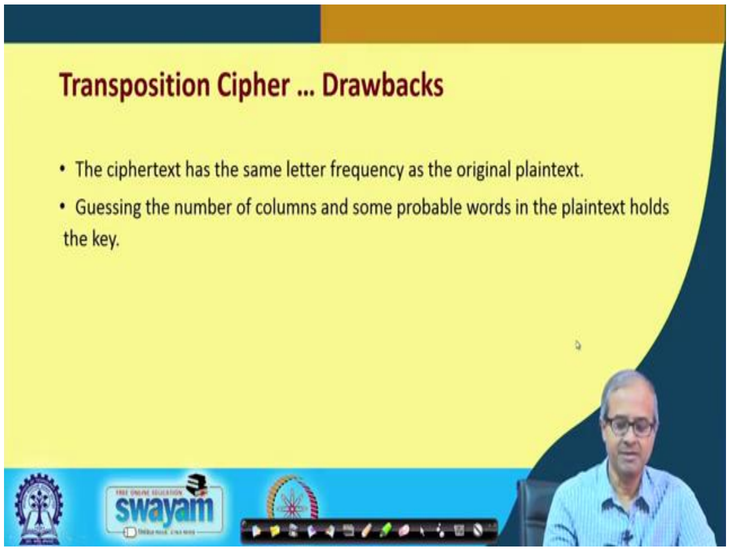
Let us take an example, fine. So, here I worked out an example, where my plaintext is something like this; welcome to the NPTEL course on ethical hacking, this is my plaintext, let us say. And I have decided that I will be using 7 columns. This is one I have decided. So, I write down this text, well here these spaces I am showing as dash; welcome to the NPTEL course on ethical hacking and just to fill up, the last three are $-, -, -$.

Now, I choose some order of the columns as my key, now you see on top it is mentioned this is 1, this is 2, this is 3, this is 4 like this. So, when I generate the cipher text, I will be reading out the columns in that order; first I will be reading out column number one. It is $l\, o\, p\, u - l\, n$, you see this is one, this is the first column. Then the second column $c - t\, r\, e - g$ this, then the third, this one, then the fourth like this, then fifth, then sixth and finally 7th. So, you see the cipher text is apparently all jumbled up. You cannot get any information from here apparently, right. So, this is how transposition cipher works.

But let us see how we can break this kind of a cipher. Let us take a very simple example. Suppose well I know the context under which this message transfer is taking place. So, I can guess, some probable word that is appearing in my cipher text or my message. Let us say, I know that this ethical is a word which is likely to appear. So, in this transposition what will happen, these alphabets in the word ethical they will be ordered in some arbitrary way. Now here, we will try to find out how this $e\ t\ h\ i\ c\ a\ l$ letters are coming. They will be coming with some gaps; here I have 7 columns, so they should come with a gap of 7.

So, you have to see whether these letters are all coming in gaps of 7, but their orders are slightly different. So, just you can write a computer program to do this automatically, it is not so difficult to do. But once you do it, you will not only get this word; you will also have an idea regarding the order of the columns where is e, where is t, where is h like that. So, in that way you can decode the message in pretty easily.

(Refer Slide Time: 22:50)

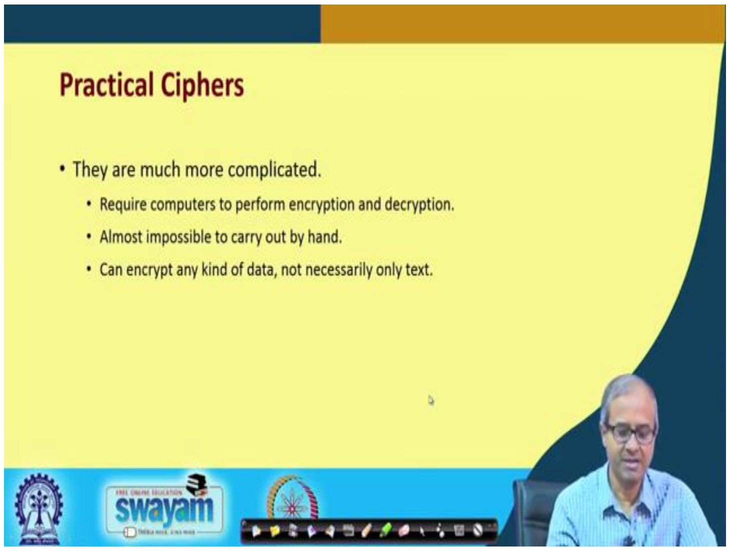


Transposition Cipher ... Drawbacks

- The ciphertext has the same letter frequency as the original plaintext.
- Guessing the number of columns and some probable words in the plaintext holds the key.

So, these are the drawback as it said. So, we are not modifying or changing any letters. So, it will have the same letter frequency as the original plaintext, guessing the number of columns as and I said some probable words will make breaking this code very simple ok. So, these are not practical ciphers; but if you combined many of them these are typically used; let us say by defense people in some places, they have some kind of codebook using that code books they do some kind of encryption ok; there some combination of these methods are used.

(Refer Slide Time: 23:35)



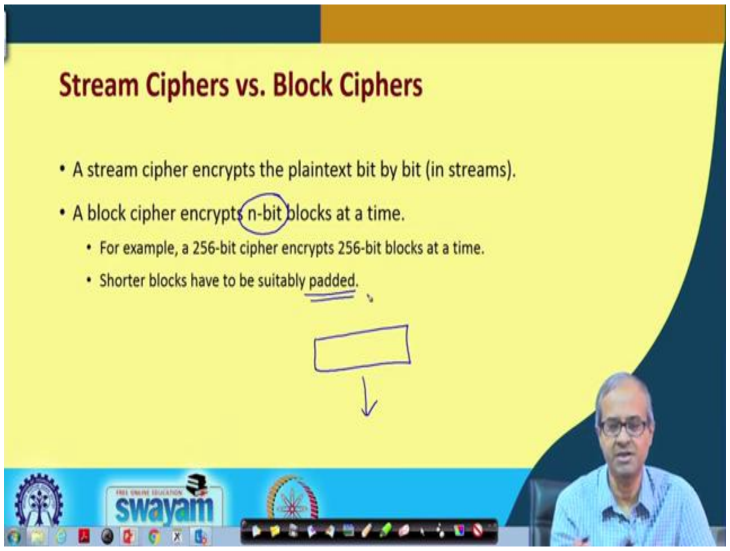
Practical Ciphers

- They are much more complicated.
  - Require computers to perform encryption and decryption.
  - Almost impossible to carry out by hand.
  - Can encrypt any kind of data, not necessarily only text.

Talking about practical ciphers, they are much more complicated not so simple; and there the message need not be a text, it can be a file, it can be an image, it can be anything, it can be an audio clip, it will be a binary file in general.

So, these algorithms are much more complicated and they will be requiring a computer to perform encryption and decryption; doing it by hand is simply out of the question, because the process is so complicated, ok. These methods as I said can encrypt any kind of data, not necessarily only text.

(Refer Slide Time: 24:17)



Stream Ciphers vs. Block Ciphers

- A stream cipher encrypts the plaintext bit by bit (in streams).
- A block cipher encrypts n-bit blocks at a time.
  - For example, a 256-bit cipher encrypts 256-bit blocks at a time.
  - Shorter blocks have to be suitably padded.

So, another classification let me talk about; there is something called stream ciphers, something called block ciphers. Stream ciphers carry out encryption and decryption of a continuous stream of data that is coming bit by bit. They encrypts the plaintext bit by bit, it is coming continuously. Let us say a streaming video is being played, where the bits that are coming I am doing some encryption and sending out the encrypted bit stream in the output channel.

And at the receiving end, the similar thing happens. Bit by bit they are getting decrypted. But in a block cipher, we consider n bits of the block at a time; it can be 64 bits, 128 bits or 256 bits whatever is the value of n. So, I take a certain number of bits of the cipher text, I convert it or decode it into plaintext or for encryption the reverse process. I take certain number of bits of my original message, I encrypt it, receiving and I will decrypt it, this is called block cipher. So, if my block size is smaller; then I will have to do some kind of padding, padding with zeros or ones to bring the block size to the required value.

So, with this we come to the end of this lecture; where we discussed some of the classical private key encryption algorithms. Now in our next lecture we shall be looking at, some of the algorithms which have been practically used; this we shall be seeing in the next lecture.

Thank you.