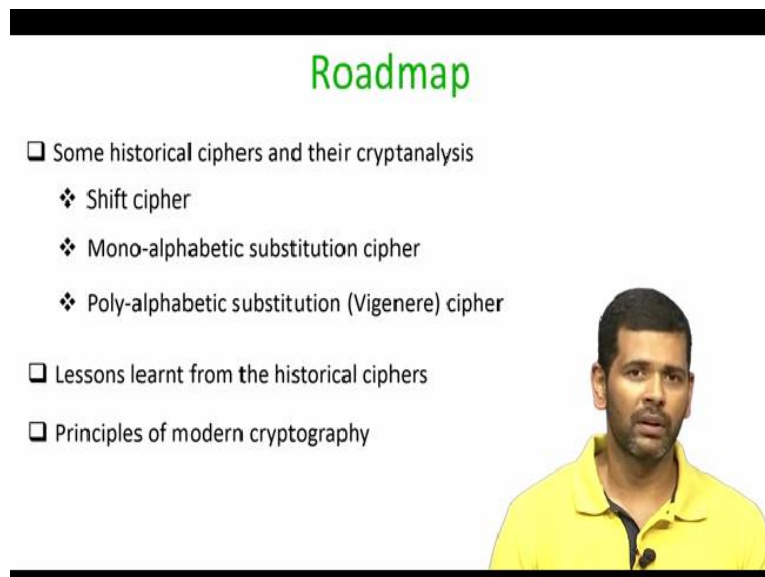


Foundations of Cryptography
Prof. Dr. Ashish Choudhury
(Former) Infosys Foundation Career Development Chair Professor
International Institute of Information Technology-Bengaluru

Lecture-03
Historical Ciphers And Their Cryptanalysis

Hello everyone, welcome to the third lecture. The plan for this lecture is as follows.

(Refer Slide Time: 00:34)



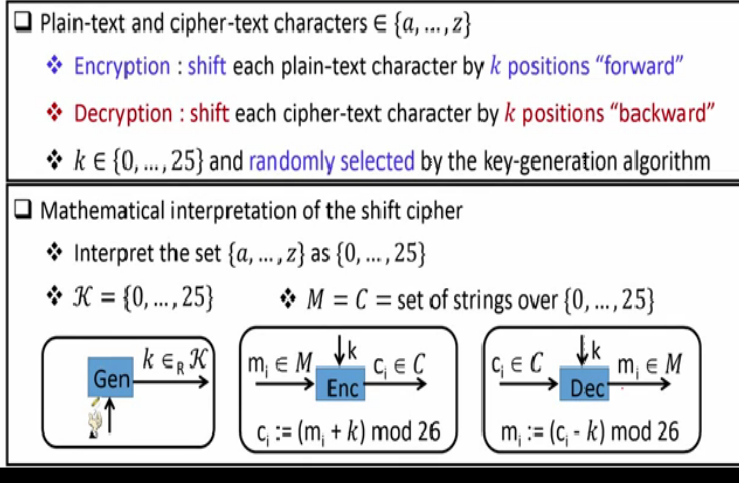
The slide titled "Roadmap" in green text lists the topics for the lecture. It includes a checkbox for "Some historical ciphers and their cryptanalysis" which is expanded to show three sub-points: "Shift cipher", "Mono-alphabetic substitution cipher", and "Poly-alphabetic substitution (Vigenere) cipher". Below this is a checkbox for "Lessons learnt from the historical ciphers" and another for "Principles of modern cryptography". A small video inset of Prof. Dr. Ashish Choudhury is visible in the bottom right corner of the slide.

- ❑ Some historical ciphers and their cryptanalysis
 - ❖ Shift cipher
 - ❖ Mono-alphabetic substitution cipher
 - ❖ Poly-alphabetic substitution (Vigenere) cipher
- ❑ Lessons learnt from the historical ciphers
- ❑ Principles of modern cryptography

In this lecture we will discuss some of the historical ciphers such as shift cipher, mono alphabetic substitution cipher and poly alphabetic substitution cipher and we will see their cryptanalysis, namely, we will see how badly these ciphers were broken. We will see the lessons that we learn from these historical ciphers. And finally, we will discuss the principles of modern cryptography.

(Refer Slide Time: 00:56)

Shift Cipher



So, let us discuss the shift cipher which is a very simple symmetric encryption process and in the shift cipher the plain text and the cipher text characters belong to the alphabet of some natural language. So for instance we assume that the plaintext and ciphertext characters belong to the English alphabet set namely, the set of letters a to z. And the encryption process here is to shift each instance of the plain text character by k positions forward.

And the decryption process is to shift back each ciphertext character by k positions in the backward direction. And the shift here namely the value of k is going to be one of the values in the set 0 to 25, which is randomly selected by the key generation algorithm. So, the mathematical interpretation of the shift cipher is as follows. Since the plain text characters are belonging to the set a to z, we interpret the character ‘a’ as the symbol 0 character ‘b’ as a symbol 1 and in the same way we interpret the character ‘z’ as the symbol 25. That is a mapping we assume is available both to the sender and the receiver or to any third party who is going to use an instance of shift cipher. Now the key space here is the set 0 to 25. Because the actual value of shift namely k is going to be one of the value in randomly selected in this set.

And the plaintext space and the ciphertext space is going to be the set of strings over the set 0 to 25. Because my plaintext is going to be any English text where the characters of the plain text will be mapped to the alphabet in the set 0 to 25. And in the same way, the cipher text which is

going to be some text over the set 0 to 25, will be interpreted as some string over the set 0 to 25. So syntactically, the key generation algorithm for the shift cipher is as follows.

The key generation algorithm is going to output a uniformly random value namely ' k ' which is going to be the value of shift which sender and receiver are going to use and this shift is going to be a random value from the key space. So, the notation here belonging to subscript R denotes that the output of the key generation algorithm, namely ' k ' is randomly selected over the key space.

And randomness here is uniform randomness. So the notation \in_R denotes that the value is uniformly selected over the set ' K '. So that is the interpretation of the notation belonging to subscript r . The syntax for the encryption algorithm of the shift cipher is as follows. So imagine we are given an English text which we are interpreting as a string over the set 0 to 25. That means each character of the plain text is a symbol in the set 0 to 25.

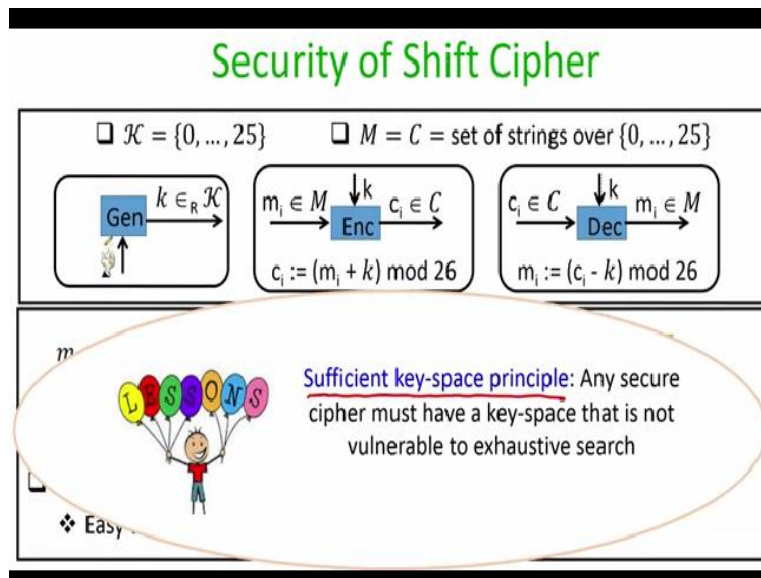
So the encryption of the i^{th} plain text character happens as follows, we take i^{th} plain text character m_i and we have the value of shift which is going to be any of the value in the range 0 to 25. And encryption algorithm produces and i^{th} cipher text character, where the i^{th} cipher text character is the summation of i^{th} plain text character with the shift modulo 26. Now this operation modulo 26 ensures that, we do the wraparound if you are actually going beyond value 25.

What I mean by this is, if, for example, your m_i the i^{th} plaintext character is say 'a', and if my value of ' k ' is say 25, right or say my i^{th} plain text character, say 'b', and if my value of ' k ' is 25, then if 'b' gets mapped to 1, and my k is 25. And if I add 25 to 1, actually, I will be crossing the value, the range of values that is allowed for the ciphertext characters, namely 0 to 25. That means I have to do a wrap around.

And that effect of wrap around is happening by doing the modulo operation. So that is the encryption process of the shift cipher and analogously we have the corresponding decryption operation, where we take i^{th} cipher text character, the value of shift and to recover back i^{th} plain

text character, we simply go back 'k' positions back from the i^{th} cipher text character. And if required, we do the wraparound by doing the modulo operation, modulo 26 operation.

(Refer Slide Time: 05:25)



So now let us try to analyze the security of shift cipher and we will do the analysis in the simplest possible attack model namely the ciphertext only attack model, where the scenario is the following, we assume that we have a sender and a receiver and sender has encrypted an English text consisting of 'l' characters where each of the characters belongs to the set 0 to 25. And there is a random value of shift which has been already agreed upon between the sender and the receiver.

The adversary has intercepted the cipher text which consists of 'l' cipher text characters and it knows that the i^{th} ciphertext character is related to the unknown i^{th} plaintext character by this relationship. So here m_i as well as the k is unknown to the attacker. And the goal of the attacker is to recover back the message. So the simplest possible attack which adversary can launch here is what we call us the brute force attack.

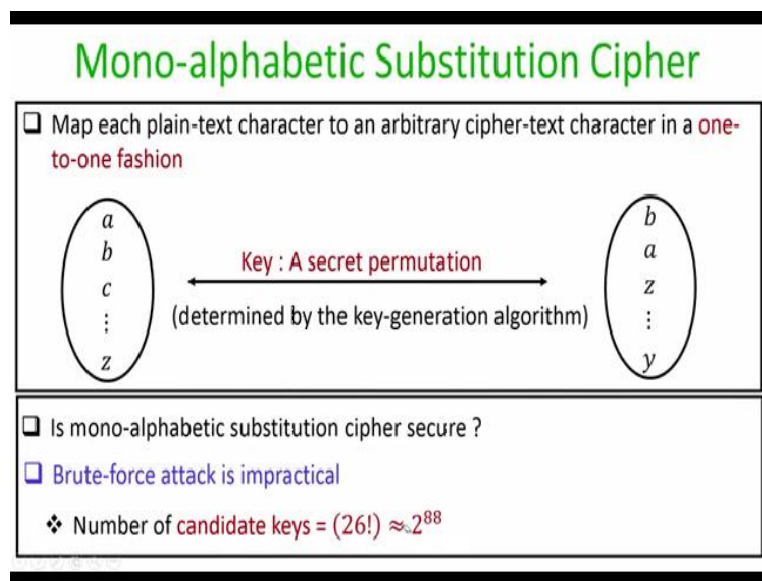
And the idea behind a brute force attack is the following. What adversary has to do is, it knows that the value of unknown k is one of the possible values in the range 0 to 25. So what it can do is it can take the ciphertext c and it can think in his mind all possible candidate values of k and it

can try to decrypt the cipher text as per the candidate k itself, because it knows the corresponding decryption process as well.

And as soon as it hits upon the candidate key for which the decryption gives back a meaningful text, by meaningful I mean a meaningful English text, it knows that it has hit upon the right key and hence the recovered message is indeed the right message with high probability. And this attack is very easy to launch because here adversary has to do the computation namely the decryption of the cipher text only with 26 candidate keys, which is very easy to perform.

Consequently, we can say that shift cipher can be very easily broken by just doing a brute force attack. So the important lesson that we learn from this attack is what we call us sufficient key space principle which states that any secure cipher should have an enormously large key space, namely, the set of candidate keys should be enormously large, so that it becomes impractical for that adversary to do a brute force kind of attack. That is the bare minimum condition necessary, condition that we require for our cipher to be secure.

(Refer Slide Time: 07:49)



So now let us see how we can fix the problem that is associated with the shift cipher to prevent a brute force attack. And this leads to another interesting cipher which we call as mono alphabetic substitution cipher. And the idea here is instead of shifting each plaintext character by the same

unknown amount k , we shift each plaintext character by a different amount in a one to one mapping. What I mean here is the following.

We know that the plaintext characters could be any of the characters in the range a to z, or 0 to 25. What we do is we map this plain text character in a one to one mapping to the set of ciphertext character a to z. That means that the key is going to be a secret permutation of the set 0 to 25, which is going to be known only to the sender and the receiver.

For example, if we take this specific permutation which maps the plain text character a to b, that means in the plain text wherever the plain text character is a, all those instances of a are going to be replaced by b in the same way wherever in the plain text, we have the character c all those instances of c are going to be replaced by z and so on. Since the receiver also knows the same secret permutation on the receiving the ciphertext it has to just perform the reverse operation wherever it is seeing the ciphertext character z, it has to replace it with c and so on.

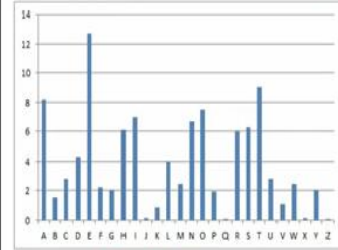
So the decryption operation can be performed by the receiver. Now you can see the candidate key space here is extremely large, namely, the set of possible keys is actually 26 factorial because each key is nothing but a permutation of the set a to z and there could be 26 factorial such candidate keys. So if the adversary tries to do a brute force attack, which we saw in the context of shift cipher, the amount of brute force or the computation that adversary has to perform is of order of magnitude 2 to the power 88 (2^{88}), which is an enormous amount of computation. That means we can definitely say that brute force attack is not possible to launch over shift substitution cipher.

(Refer Slide Time: 09:52)

Cryptanalysis of Mono-alphabetic Substitution

Frequency analysis : applicable when the plaintext space is a natural language

Idea: exploit the redundancy present in the underlying natural language



Average English letter frequency

Bigram	Percentage	Bigram	Percentage
TH	3.15	HE	2.51
AN	1.72	IN	1.69
ER	1.54	RE	1.48
ES	1.45	ON	1.45
EA	1.31	TI	1.28
AT	1.24	ST	1.21
EN	1.20	ND	1.18

Average English bigram frequency

THE, ING, AND, HER, ERE, ENT, THA, NTH,
WAS, ETH, FOR

Popular English trigram in decreasing order

Most frequently occurring character/bigram/trigram in the ciphertext, corresponds to most frequently occurring character/bigram/trigram in the plaintext

But that does not necessarily mean that that is the only attack possible on the mono alphabetic substitution cipher. In fact, it turns out that there is a very interesting attack, which we call as the frequency analysis attack. And this attack can be launched on any cipher, where the underlying plain text space is a natural language which is actually the case in our example, because we are considering a setting where the plain text are actually English text. So, the idea here is to exploit the redundancy which is available in any natural language for example, English. What I mean by redundancy is the following. If we consider large, long English text and we know that on an average the character E occurs more frequently compared to any other character.

In the same way we know on an average in long English text, the character T occurs more frequently compared to the character U and so on. Not only we have the average English letter frequency available for long English text, we can also say that we are available with the average frequency of English bigrams in long English text, for example we know that in long English text on an average the bigram 'th' occurs more frequently compared to the bigram 'nd' and so on.

And in the same way, we have the frequency of most popular English trigrams and so on. And this is a public data available to any user in the universe. So, the idea behind the frequency analysis is that even though each character of the plain text got mapped to a different ciphertext

character, the relationship between most frequently occurring ciphertext character and a plain text character is preserved.

That means if the attacker sees an enormously long ciphertext corresponding to the encryption of enormously long English plain text, and what the adversary can do is it can perform or it can prepare a chart similar to the chart of average English letter frequency where it will tell him which cipher text character is occurring most frequently compared to other ciphertext character and so on.

So for instance, imagine that after preparing this chart adversary finds out that the ciphertext character R occurs the maximum amount of time. So, what adversary can think in his mind to decipher text character R corresponds to the plain text character E and so on. In the same way, the relationship between the most frequently occurring cipher text bigram and the most frequently occurring plain text bigram is preserved and so on.

So, by doing the comparison, adversary can easily find out the corresponding secret permutation between the plain text character space and a cipher text character space which the sender and the receiver are operating the instance of mono alphabetic substitution cipher. So, as you can see, even without doing the brute force attack, adversary could easily recover the secret permutation.

And hence the mono alphabetic substitution cipher can be very easily broken in the cipher text only attack model right.

(Refer Slide Time: 12:54)

Poly-alphabetic Substitution (Vigenere) Cipher

❑ Idea : invoke **multiple instances** of shift cipher

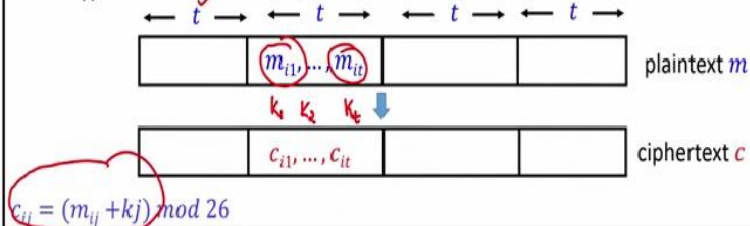
❖ In each instance, a plain-text character is mapped to a different ciphertext character

❑ $M = C = \{0, \dots, 25\}^*$

❑ $\mathcal{K} = \{0, \dots, 25\}^t$, t randomly chosen by Gen

❑ Key-generation algorithm : output a **uniformly random key** $k = (k_1, \dots, k_t)$

❑ Encryption:



So now let us see another interesting historical cipher which we call as poly-alphabetic substitution cipher, and which is also called as Vigenere cipher after the name of its inventor. And the idea here is to fix the potential problem that was there in context of mono alphabetic substitution cipher. Namely, the idea here is to invoke multiple instances of shift cipher in such a way that in each instance, a plain text character is mapped to different ciphertext character.

That means it is not going to happen that wherever the plaintext character a is it always gets mapped to a specific ciphertext character that is not going to happen, that will mean the relationship between the most frequently occurring plaintext character and the most frequently occurring ciphertext character is going to be disturbed. And the way we achieve this is as follows. So again here the message and the plain text space and the ciphertext space is the set of all possible it is a set of strings over the set 0 to 25. And the key space is now going to be a string of length t , where t is also randomly selected by the key generation algorithm and value of t could be anything with the minimum value of t being 1. And the value of key is going to be any string of length t over the set 0 to 25. That means your key generation algorithm is going to output a key consisting of t characters, where t is also randomly selected by the key generation algorithm. And each character of the key is going to be one of the values in the set 0 to 25 right, that is a key generation algorithm. And the encryption algorithm happens as follows. So imagine the sender has a plain text, what sender is going to do is the following it is going to divide its

plain text into blocks of t , t , t , it might be the case that client is not actually a multiple of t . That might be possible that the last block may not be a multiple of t .

But for simplicity assume that the last block is also a multiple of t . Once the message is divided into blocks of size t , t and t . The idea here is to encrypt each of the block using the key by running an instance of shift cipher. Namely we produce i^{th} ciphertext block as follows. So imagine that the characters in the i^{th} block are m_{i1} , m_{i2} and m_{it} . So there are t such characters. And we have the key characters k_1 , k_2 and k_t .

So, the idea here is we produce the i^{th} ciphertext, with the characters of the i^{th} cipher text blocks are c_{i1} , c_{i2} , and c_{it} , where the j^{th} character of the i^{th} ciphertext block is obtained by shifting the i^{th} j^{th} plaintext character by the shift k_j modulo 26.

(Refer Slide Time: 15:44)

Vigenere Cipher : An Example

❑ Key $k = \text{CIPHER} = (2, 8, 15, 7, 4, 17)$

❑ Plaintext $m = \text{thiscriptosystemisnotsecure}$

19 7 8 18 2 17	24 15 19 14 18 24	18 19 4 12 8 18	13 14 19 18 4 2	20 17 4
$\leftarrow 6 \rightarrow$	$\leftarrow 6 \rightarrow$	$\leftarrow 6 \rightarrow$	$\leftarrow 6 \rightarrow$	$\leftarrow 3 \rightarrow$
$+ \text{mod } 26$	$+ \text{mod } 26$	$+ \text{mod } 26$	$+ \text{mod } 26$	$+ \text{mod } 26$
2 8 15 7 4 17	2 8 15 7 4 17	2 8 15 7 4 17	2 8 15 7 4 17	2 8 15
21 15 23 25 6	8 0 23 8 21 22 15	20 1 19 19 12 9	15 22 8 25 8 19	22 25 19

❑ Ciphertext $c = \text{vp} \underline{x} \underline{z} \underline{g} \underline{i} \underline{a} \underline{x} \underline{i} \underline{y} \underline{w} \underline{p} \underline{u} \underline{b} \underline{t} \underline{t} \underline{m} \underline{j} \underline{p} \underline{w} \underline{i} \underline{z} \underline{i} \underline{t} \underline{w} \underline{z} \underline{t}$

So let me demonstrate what exactly I am trying to say. Imagine that the value of t is 6 here okay, and key which is randomly generated by the key generation algorithm suppose it is the character “cipher” namely we map c to 2, to i to 8 and so on. That means this is the random key which is generated by the key generation algorithm. And it is known both to the sender as well as to the receiver right.

Now assume the plaintext which sender wants to encrypt is the English message, “thiscryptosystemisnotsecure”. So what the first step to encrypt display in text will be to map each of the characters in the plain text to the set 0 to 26, namely to the characters 0 to 26. So for example, t gets mapped to 19, h gets mapped to 7 and so on. Once we convert the English plain text into letters in the range 0 to 26, next we divide the message into blocks of 6 6 6 because our keys of size 6 characters, so as you can see, in this particular example, the last block is not a multiple of 6, so it consists of only 3 symbols, that is fine. Now what we do is we encrypt the first block as follows. The key for our example is string 2 8 15 7 4 and 17, what we do is we add this letters are the symbols or numbers to the block 19 7 8 18 2 and 17 and each of the addition is performed modulo 26.

Namely, the character 19 is shifted by 2. And since we are not crossing the range 0 to 26, we obtain 21. In the same way, the character 7 gets shifted by 8 to obtain the character 15. And in the same way the last character of the first block which is 17 gets shifted by the amount 17 to obtain the character 6. So now you can see since we are actually crossing the range 0 to 25, we do the mod 26 operation to do the wraparound. And as a result of doing the wraparound we get the ciphertext character as 6. In the same way we take the second block, and again we add the characters of the key one by one and do the modulo 26 operation and consequently, once we obtained the cipher text characters where the alphabets belong to the set 0 to 26. Each of those alphabet characters get mapped back to the English character set.

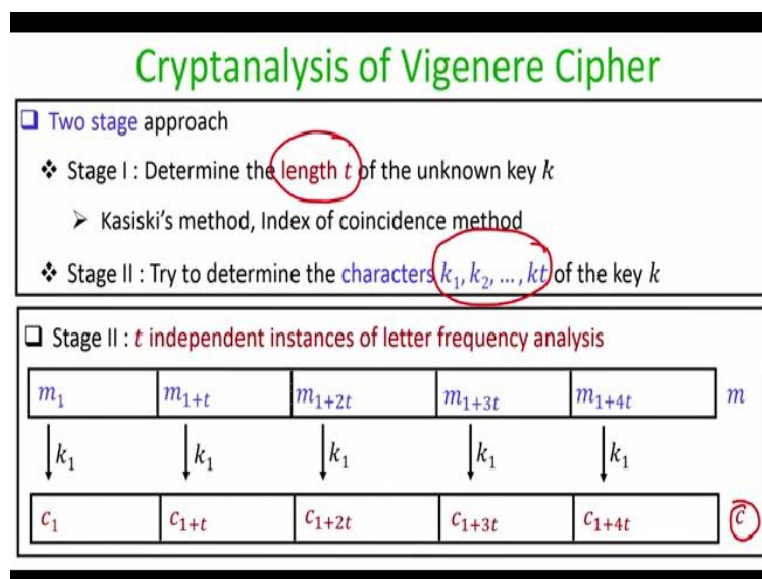
And as a result, the string “vpxzgiaxivwpubttmjpwizitwzt” will be the encryption, this string will be the encryption of the plain text “thiscryptosystemisnotsecure”. At the receiving end the receiver is also going to perform the same operation, it will divide the ciphertext into blocks of 6 6 6. And in this case, the last block will consist of 3 characters, interpret them as symbols in the range 0 to 25 and then subtract each of the key character modulo 26 to obtain back the plaintext.

That is the way we are going to perform the encryption and decryption operation in the Vigenere cipher. And now the idea here is the following. If you see the encryption, and if you see the plain text and the ciphertext, what I have highlighted here is all the instances of the symbol ‘s’ right. You see each instance of the symbol ‘s’ gets mapped to different cipher text character. For

example, the first instance of the plain text character s gets mapped to z right. The second instance of the letter s gets mapped to v and so on. And the reason this is happening is because each time the value s get shifted by different amount, which depends upon the position of the plaintext character s that means where exactly it is operating, and by how much amount did get shifted, the amount of shift by which s is getting shifted will not be same across all the instances of the letter s .

Because that depends upon the position of the letter s as well as the value of the character of the key which is going to be used for shifting the instance of s . Right, as a result, we are actually somehow disturbing the pattern which is available between the plain text and the cipher text character, that means we can no longer say that the most frequently occurring ciphertext character corresponds to the most frequently occurring plaintext character and so on.

(Refer Slide Time: 20:06)



So, now you can see one may feel that Vigenere cipher is really difficult to break, but it turns out that we can actually perform the cryptanalysis of Vigenere cipher that means we can break the Vigenere cipher in the ciphertext only attack model itself. And attack happens by doing a performing a two stage approach. In stage 1, the goal will be to determine the length of t namely the size of the period of the key right.

So, remember, t could be any value starting from 1 onwards and there are several well known methods, and an interesting method to determine the value of t . So, for example, you can use Kasiski's method or another interesting method called index of coincidence method and so on. Once we know the value of t , in stage 2 what we are going to do is we are going to recover back the unknown characters of the key.

And once we know the unknown characters of the key, we can easily perform the decryption operation and recover back the plain text. So, what I will do is I will assume that we know how to determine the value of t and I leave it as an assignment for you to go through the references and see these methods and find out how exactly the value of unknown t can be computed. What I am going to discuss is the following is that once the value of t is known, how to find out the unknown characters of the key right.

And the idea here will be to run t independent instances of frequency analysis. What I mean by that is imagine the adversary got the ciphertext because we are assuming we are in the ciphertext only attack model, and its goal is to find out the unknown plain text. The adversary knows the following that, if we consider the first ciphertext character, and then the $(t + 1)^{\text{th}}$ ciphertext character and then $(2t + 1)^{\text{th}}$ ciphertext character and so on. Then this constitutes a stream of cipher text characters. And this stream of cipher text characters is related to the unknown stream of plain text characters consisting of the first plain text character, $(t + 1)^{\text{th}}$ plain text character, $(2t + 1)^{\text{th}}$ plain text character and so on by the relationship is that the stream of ciphertext character is actually a shifted version of the corresponding stream of the plain text characters, where the amount of shift is k_1 , which exactly adversary wants to compute.

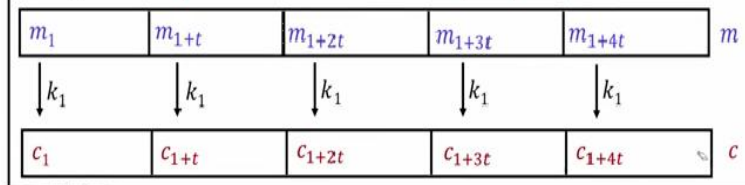
(Refer Slide Time: 22:33)

Cryptanalysis of Vigenere Cipher

❑ Two stage approach

- ❖ Stage I : Determine the **length t** of the unknown key k
 - Kasiski's method, Index of coincidence method
- ❖ Stage II : Try to determine the **characters k_1, k_2, \dots, k_t** of the key k

❑ Stage II : t independent instances of letter frequency analysis



In the same way the adversary can think of second stream of ciphers consisting of the second cipher text character and $(2 + t)^{\text{th}}$ cipher text character and so on. And it knows that all the second stream of cipher text character is related to another stream of plain text character, namely the stream of plain text characters consisting of m_2 and $m_{(2+t)}$ and so on right. By the relationship that all of them are shifted version, where the amount of shift is some unknown value of k_2 and so on.

That means from the viewpoint of the attacker what attacker can do is it can just separate out and form t independent streams of ciphertext character, and it has to now just perform a frequency analysis on each of these independent streams. And if we assume that the plaintext is sufficiently long, corresponding ciphertext is also sufficiently long, then by performing the frequency analysis on this t dependent streams, the adversary could easily recover back the unknown characters of the key.

And hence it could recover the key completely. So that is an interesting method to do the crypt analysis of Vigenere cipher right.

(Refer Slide Time: 23:45)

Lessons Learnt from Historical Ciphers

❑ Can be broken by launching a ciphertext-only attack

➤ Imagine how badly they can be broken in the stronger attack models

❑ Sufficient key-space principle

➤ Key space should be sufficiently large to make brute-force attack infeasible

➤ Only a necessary condition for a secure cipher

❑ Designing secure ciphers is a tough task

➤ Vigenere cipher, German Enigma, etc though considered secure, but eventually broken





So what are the lessons we learned from the historical ciphers. The lessons we learned are the following. All of them can be easily broken in cipher text only attack model. So imagine how badly they could be broken if we consider this ciphertext or this encryption process to be operated in a more stronger attack model say the KPA attack model and CPA attack model there the adversary will be privileged with some more additional information.

So for example, if you consider this CPA attack model, then the adversary not only have access to the ciphertext, it will also have access to the encryption oracle and hence it could more comfortably break the underlying encryption process. Another lesson that we learned from the historical ciphers is the sufficient key space principle, which says that the bare minimum necessary condition that any secure cipher should have is to ensure that its key space is sufficiently large to make the brute force attack infeasible.

And overall, the most important lesson that we learned from the historical ciphers is that designing secure cipher is indeed a tough and challenging task. Ciphers were proposed over a period of time and people thought they are secure, but eventually all of them were badly broken.

(Refer Slide Time: 24:59)

Classical vs Modern Cryptography

<ul style="list-style-type: none"><input type="checkbox"/> Classical cryptography was an art<ul style="list-style-type: none">➤ No scientific foundations --- end result : disaster<input type="checkbox"/> Modern cryptography:<ul style="list-style-type: none">➤ Strong scientific foundations and principles	
<ul style="list-style-type: none"><input type="checkbox"/> Principle 1: Formal security definitions<input type="checkbox"/> Principle 2: Precisely stating any (unproven) assumption used in the construction<input type="checkbox"/> Principle 3: Rigorous proof of security	

That brings us to the difference between classical and modern cryptography. If you consider classical cryptography, it was not a science it was mostly an art because there were no scientific foundations, no proper definitions, no mathematical security proof, and as a result the end result was very disastrous. Whereas modern cryptography is a science and it is based on strong scientific foundations and principles.

Namely the 3 principles of modern cryptography are : principle 1 is formal security definitions where we properly and mathematically define what exactly we mean by security. Principle 2 states that once we have the definition and proceed to construct a primitive which satisfies a given definition, then we should precisely state any assumption or namely unproven assumption, which we are using in our construction. Looking ahead we will see in this course that all the crypto primitives that we are going to design their security will be based on some unproven hardness assumptions. So principle 2 clearly states that we should precisely state in our construction, what exactly is the underlying assumption used in our construction and principle 3 states that once we have the construction we have to formally prove that indeed the construction that we have given satisfies the definition that we have given in stage 1. So, that brings me to the end of this lecture. To conclude, we discussed some of the historical ciphers and how badly they were broken.

And we also discuss some of the important lessons that we learned from the historical ciphers, and we discussed the principles of modern cryptography. I hope you enjoyed this lecture. Thank you.