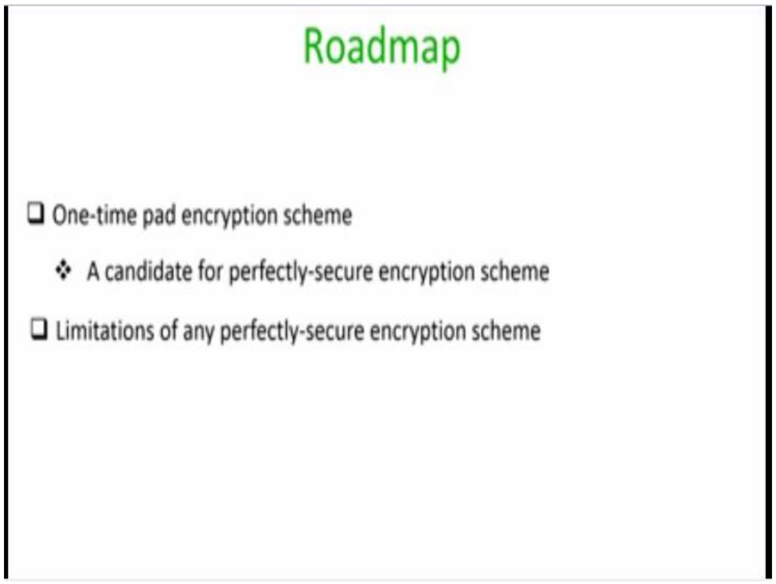
Foundations of Cryptography

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

Lecture-05
Limitations of Perfect Security

Hello everyone, welcome to lecture 5.

(Refer Slide Time: 00:30)

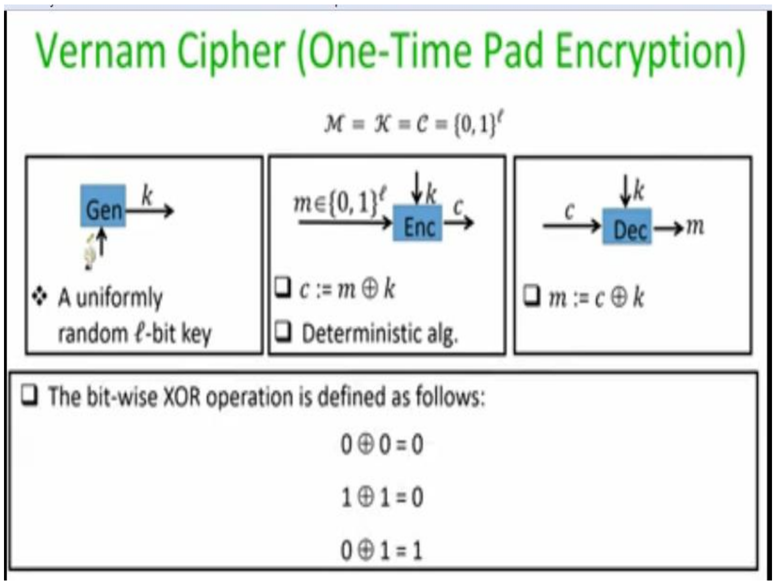


Roadmap

- ❑ One-time pad encryption scheme
  - ❖ A candidate for perfectly-secure encryption scheme
- ❑ Limitations of any perfectly-secure encryption scheme

So, the plan for this lecture is as follows We will see a candidate for perfectly secure encryption, which we call as one time pad and we will see the limitations that are imposed by any perfectly secure encryption scheme.

(Refer Slide Time: 00:44)


Vernam Cipher (One-Time Pad Encryption)

$\mathcal{M} = \mathcal{K} = \mathcal{C} = \{0,1\}^\ell$

- ❖ A uniformly random ℓ-bit key
- ❑ $c := m \oplus k$
- ❑ Deterministic alg.
- ❑ $m := c \oplus k$

❑ The bit-wise XOR operation is defined as follows:

$0 \oplus 0 = 0$

$1 \oplus 1 = 0$

$0 \oplus 1 = 1$

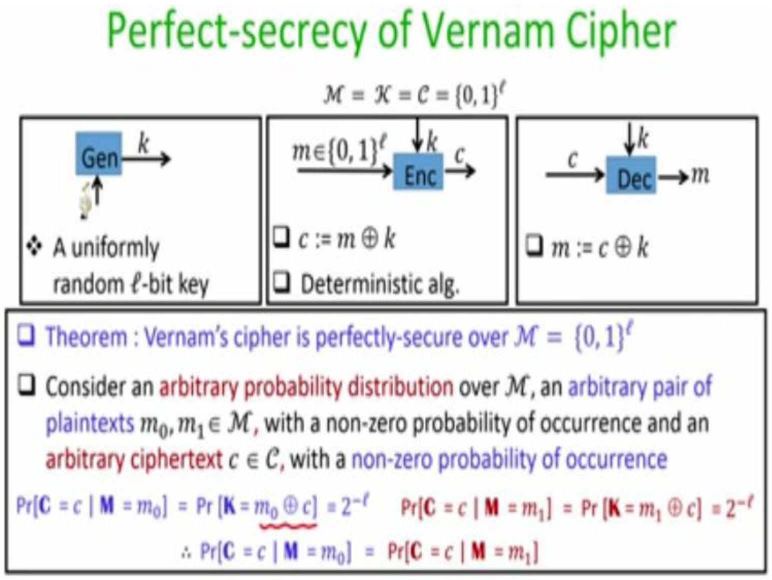
So, the one time pad encryption process, which is also called as Vernam cipher is very simple. Here the plain text space, the key space and a cipher text space are all *l* bit strings, where *l* is some system parameter which is publicly known to the sender, to the receiver and to anyone who is using this system. The key generation algorithm is going to output a uniformly random key. So since the key space is set as *l* bit strings, the key generation algorithm is going to output a uniformly random *l* bit key.

The encryption algorithm is as follows. So it takes a plain text which is going to be an *l* bit string and the key k generated by the key generation algorithm, which is also an *l* bit string. And the ciphertext is an *l* bit string, where the ciphertext is nothing but the XOR of the plain text characters and the key characters bit by bit. And as you can see that this is a deterministic algorithm there is no internal randomness as part of this encryption algorithm.

That means if you encrypt the same message m using the same key k you are going to get the same ciphertext. The decryption algorithm is just a reverse operation of the encryption algorithm in the sense it takes a ciphertext which is going to be an *l* bit string and the key k which is also going to be an *l* bit string. And to recover the message the decryption algorithm just performs the XOR of the cipher text with the key k bit by bit.

So just to recall, what exactly is an XOR operation, so the XOR operation operates with 2 bits at a time and the output is defined as follows if both the bits are same, then the output is going to be 0. Whereas if the bits are different, and output is going to be 1, that is what is the XOR operation, right.

(Refer Slide Time: 02:30)


Gen
k
\$
$m \in \{0,1\}^\ell$
k
Enc
c
c
k
Dec
m

Perfect-secrecy of Vernam Cipher

$\mathcal{M} = \mathcal{K} = \mathcal{C} = \{0,1\}^\ell$

❖ A uniformly random ℓ-bit key

❑ $c := m \oplus k$

❑ Deterministic alg.

❑ $m := c \oplus k$

❑ Theorem : Vernam's cipher is perfectly-secure over $\mathcal{M} = \{0,1\}^\ell$

❑ Consider an arbitrary probability distribution over $\mathcal{M}$, an arbitrary pair of plaintexts $m_0, m_1 \in \mathcal{M}$, with a non-zero probability of occurrence and an arbitrary ciphertext $c \in \mathcal{C}$, with a non-zero probability of occurrence

$\Pr[\mathbf{C} = c \mid \mathbf{M} = m_0] = \Pr[\mathbf{K} = m_0 \oplus c] = 2^{-\ell}$ $\Pr[\mathbf{C} = c \mid \mathbf{M} = m_1] = \Pr[\mathbf{K} = m_1 \oplus c] = 2^{-\ell}$

$\therefore \Pr[\mathbf{C} = c \mid \mathbf{M} = m_0] = \Pr[\mathbf{C} = c \mid \mathbf{M} = m_1]$

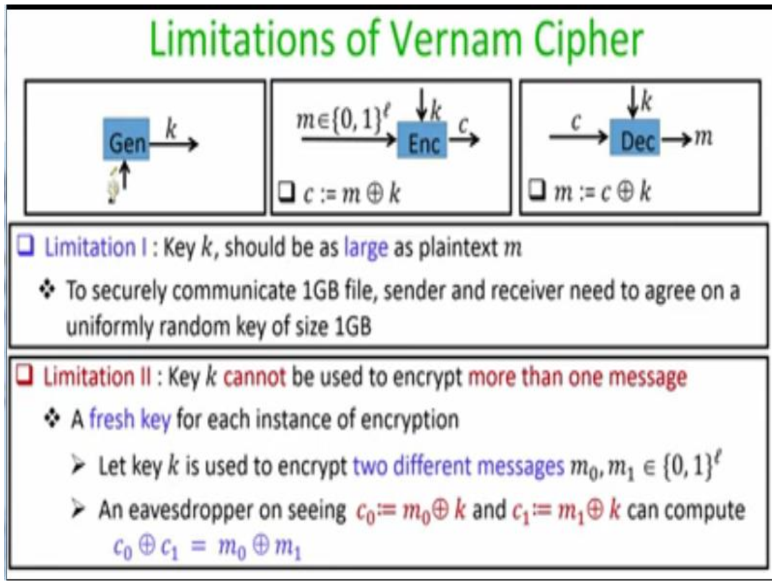
So, in some sense, you can imagine that this encryption algorithm does the masking of the message with the key, right. So the XOR operation has the effect of masking the message bit by bit with the bits of the key. Whereas the decryption operation as you can imagine, it has the reverse operation of masking namely unmasking the effect of key from the ciphertext bit by bit. So, the important thing here is that since the XOR operation is kind of reversible here, you can mask the message using the key by performing the XOR operation. And to get the effect of unmasking, you just have to XOR the key from the ciphertext to get back your message. So now let us prove that this Vernam cipher or one time pad is indeed perfectly secure. And we can prove its perfect security as per any of the 3 definitions that we saw in the last lecture.

So I am going to prove that Vernam cipher is indeed perfectly secure over the message space of *l* bit strings. And for this I am going to use the first alternate definition of perfect secrecy that we discussed in the last lecture. Namely, I am going to prove that the distribution of the ciphertext is independent of the underlying plaintext in any instance of Vernam cipher. For this we consider an arbitrary probability distribution over the plaintext space, and an arbitrary pair of messages m_0

, m_1 belonging to the plain text space as per that probability distribution which has a non zero probability of occurrence. And we also pick an arbitrary ciphertext, which has a non zero probability of occurrence. What I am going to prove is that for any m_0 and m_1 that we have picked here arbitrarily and for any cipher text c which we have picked arbitrarily here, the probability that m_0 is encrypted in c, and the probability that m_1 in encrypted is c, are same, which will prove that Vernam cipher is perfectly secure. So now let us first try to compute the conditional probability that if your plaintext is m_0, what is the probability that the Vernam cipher is going to produce the ciphertext to be c, right. So the probability that plaintext m_0 is going to produce the cipher text, c, is the same as the probability that the key which is used for encryption is this string, namely the key is XOR of m_0 and c. And what is the probability that the key which is obtained by running the key generation algorithm is indeed the XOR of m_0 and c. Well, it is $1/2^{l}$. Because, as per the syntax of our key generation algorithm, the key generation algorithm is out going to output a uniformly random key. So the probability that uniformly random key which is obtained by key generation algorithm indeed satisfies the value XOR of m_0 and c is $1/2^{l}$.

In the same way, the probability that the message m_1 is encrypted in the cipher text, c, is the same as the value of key which is used for encryption is the XOR of m_1 and c. And again, since my key generation algorithm is going to output uniform random keys, the probability that my key is XOR of m_1 and c is $1/2^{l}$. So since both these 2 probabilities are equal after seeing a ciphertext, which is going to be an l bit string, adversary cannot pinpoint whether it is an encryption of m_0, or whether it is an encryption of m_1. And hence adversary will be completely clueless. And that is why this encryption satisfies the definition of perfect secrecy. So we now have a candidate encryption scheme, which is perfectly secure.

(Refer Slide Time: 06:17)



Limitations of Vernam Cipher

Gen
k
$m \in \{0,1\}^{\ell}$
k
Enc
c
c
k
Dec
m

❑ $c := m \oplus k$

❑ $m := c \oplus k$

❑ Limitation I : Key k, should be as large as plaintext m

- ❖ To securely communicate 1GB file, sender and receiver need to agree on a uniformly random key of size 1GB

❑ Limitation II : Key k cannot be used to encrypt more than one message

- ❖ A fresh key for each instance of encryption
  - ➢ Let key k is used to encrypt two different messages $m_0, m_1 \in \{0,1\}^{\ell}$
  - ➢ An eavesdropper on seeing $c_0 := m_0 \oplus k$ and $c_1 := m_1 \oplus k$ can compute $c_0 \oplus c_1 = m_0 \oplus m_1$

So now you might be wondering that if we have a candidate encryption scheme, which is perfectly secure, why cannot we afford to use it in practice. So now let us see some of the limitations which are imposed by the one time pad scheme. The first limitation which is imposed here is that key should be as large as the plaintext because the size of the key is *l* as well as the size of the plaintext is also *l*, right.

This means that if you want to say encrypt 1 GB file, that means if the sender wants to encrypt a 1 GB file, then it has to agree upon a key which is also of size 1 GB, which seems very impractical because in practice we aim to go for encryption process, where we can use a little size key to encrypt long messages. So that is the first restriction which is imposed by Vernam cipher.

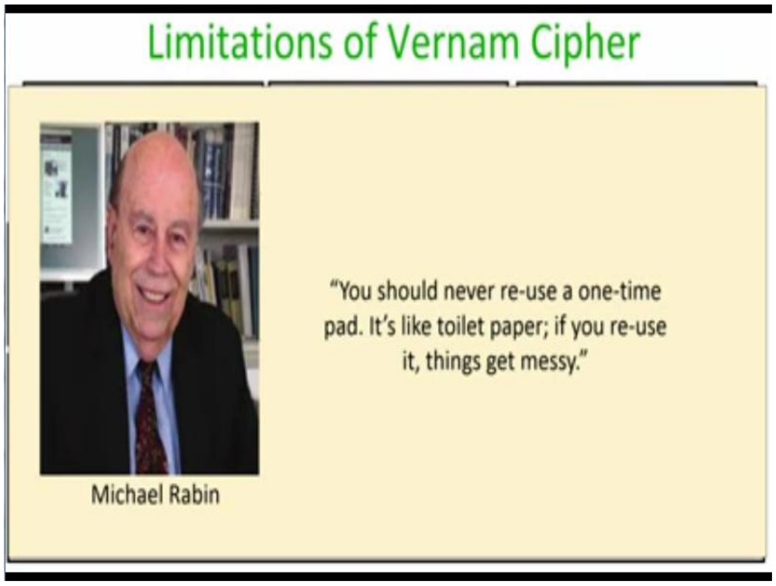
The second limitation which is imposed here is that you cannot reuse the same key for encrypting more than one message. That means each instance of the encryption needs a fresh key. I stress here that when I say here that you cannot reuse the key, I do not mean that you cannot reuse the key in the next instance, what I mean here is that in each instance, you have to again run into an independent instance or a fresh instance of key generation algorithm. It is fine if the key that you are going to obtain in the next invocation of key generation algorithm is same which you have obtained in the previous invocation. What is important here is that both this invocations of the key generation are going to output independent outputs. So from the viewpoint of the adversary or an attacker who is intercepting ciphertext, he would not be knowing that the keys which have been used whether they are same or whether they are different.

From the viewpoint of the attacker, they are independent keys. So the second restriction, which is imposed here is you cannot retain the same key and keep on encrypting multiple messages using the same key. For instance, imagine a scenario where the sender is going to use or retaining the same key k for encrypting 2 different messages m_0 and m_1 in sequence. That means it has used a key k for encrypting first a message m_0 communicated the cipher text. And suppose it is again retaining the same value of key instead of again running the key generation algorithm. And it reuses the same key k for encrypting a next message m_1, and again, the ciphertext is communicated over the channel. And suppose adversary is aware of this fact that the same key has been retained and used by the sender to encrypt two messages m_0 and m_1. So now what an attacker can do is the following.

Since it has intercepted the cipher text c_0, as well as it has intercepted the cipher text c_1 and it knows the relationship between the messages m_0 and k, m_1 and k, and c_0, c_1. Namely, it knows that c_0 is the XOR of m_0 and k and it knows that c_1 is the XOR of m_1 and k. If it performs the XOR of c_0 and c_1, the effect of k is going to vanish, right because that is what is the property of the XOR operation.

Because k will be XORed with k itself, which will give you 0. And as a result by performing the XOR of c_0 and c_1, the adversary is going to obtain the XOR of m_0 and m_1. So now you might be wondering how much information is actually revealed by learning the XOR of m_0 and m_1. And it turns out it is a significant amount of information. And that means we cannot claim the perfect secrecy for this kind of encryption process where the same k is now retained for encrypting multiple messages.

(Refer Slide Time: 09:48)



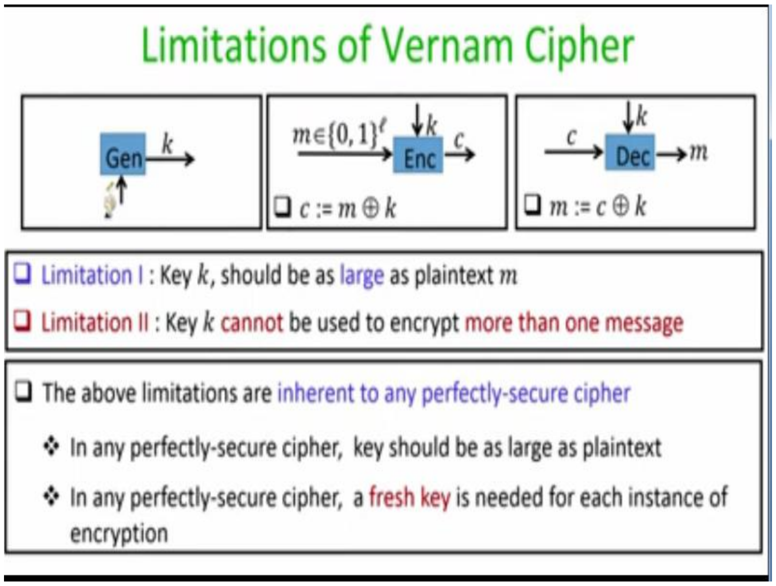
Limitations of Vernam Cipher

"You should never re-use a one-time pad. It's like toilet paper; if you re-use it, things get messy."

Michael Rabin

So here is what Turing award winner, legendary computer scientist Michael Rabin has to say about one time pad, he says that you should never reuse a one time pad, it is like a toilet paper, because if you reuse it, things can get messy, right, so you should never reuse the key. For each instance of the one time pad Vernam cipher you need to run a key generation algorithm and obtain a fresh key for encrypting your next messages.

(Refer Slide Time: 10:14)



Limitations of Vernam Cipher

$$\begin{tikzcd} \text{Gen} \arrow[r, "k"] & {} \end{tikzcd} \qquad \begin{tikzcd} {} \arrow[r, "m\in\{0,1\}^\ell"] & \text{Enc} \arrow[r, "c"] & {} \end{tikzcd} \qquad \begin{tikzcd} {} \arrow[r, "c"] & \text{Dec} \arrow[r] & m \end{tikzcd}$$

❑ $c := m \oplus k$

❑ $m := c \oplus k$

❑ Limitation I : Key k, should be as large as plaintext m

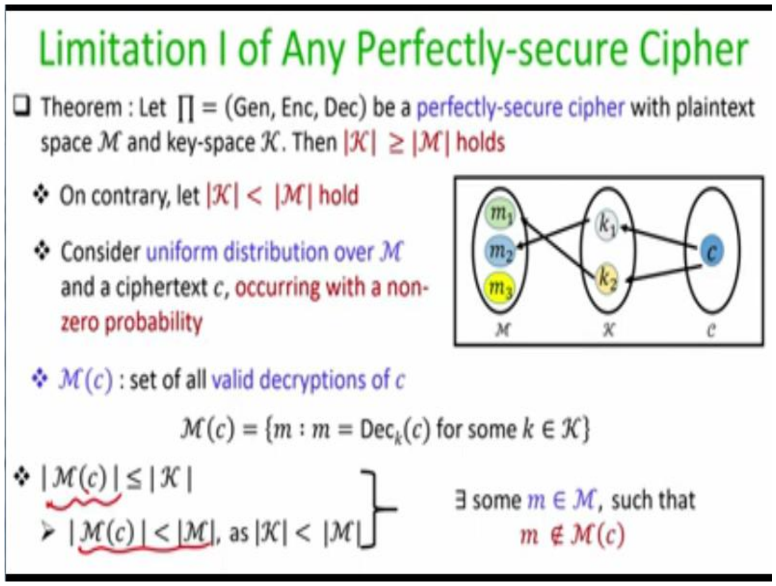
❑ Limitation II : Key k cannot be used to encrypt more than one message

❑ The above limitations are inherent to any perfectly-secure cipher

- In any perfectly-secure cipher, key should be as large as plaintext
- In any perfectly-secure cipher, a fresh key is needed for each instance of encryption

So these are the 2 restrictions imposed by Vernam cipher. Now, a natural question is whether these 2 restrictions are only with respect to one time pad or Vernam cipher, or are these 2 restrictions inherent to any perfectly secure cipher. What we are now going to prove is that these two restrictions are inherent for any perfectly secure cipher, for any perfectly secure cipher key should be as large as the plain text. And we will also prove that any perfectly secure cipher, each instance of the encryption needs a fresh key.

(Refer Slide Time: 10:49)



Limitation I of Any Perfectly-secure Cipher

- Theorem : Let $\prod$ = (Gen, Enc, Dec) be a perfectly-secure cipher with plaintext space $\mathcal{M}$ and key-space $\mathcal{K}$. Then $|\mathcal{K}| \geq |\mathcal{M}|$ holds
  - On contrary, let $|\mathcal{K}| < |\mathcal{M}|$ hold
  - Consider uniform distribution over $\mathcal{M}$ and a ciphertext c, occurring with a non-zero probability
  - $\mathcal{M}(c)$: set of all valid decryptions of c

$$\mathcal{M}(c) = \{m : m = \mathrm{Dec}_k(c) \text{ for some } k \in \mathcal{K}\}$$

- $|\mathcal{M}(c)| \leq |\mathcal{K}|$
  - $|\mathcal{M}(c)| < |\mathcal{M}|$, as $|\mathcal{K}| < |\mathcal{M}|$

$\exists$ some $m \in \mathcal{M}$, such that $m \notin \mathcal{M}(c)$

So let us first prove the first limitation here. So the theorem here, is imagine you are given a perfectly secure cipher. It may not be one time pad, it could be any perfectly secure cipher. The theorem says that if the scheme is perfectly secure, then the key space has to be as large as message space, right. So, here is the proof, the proof is going to be by contradiction. So on contrary, imagine that even though my scheme is perfectly secure my key space is less than my message space.

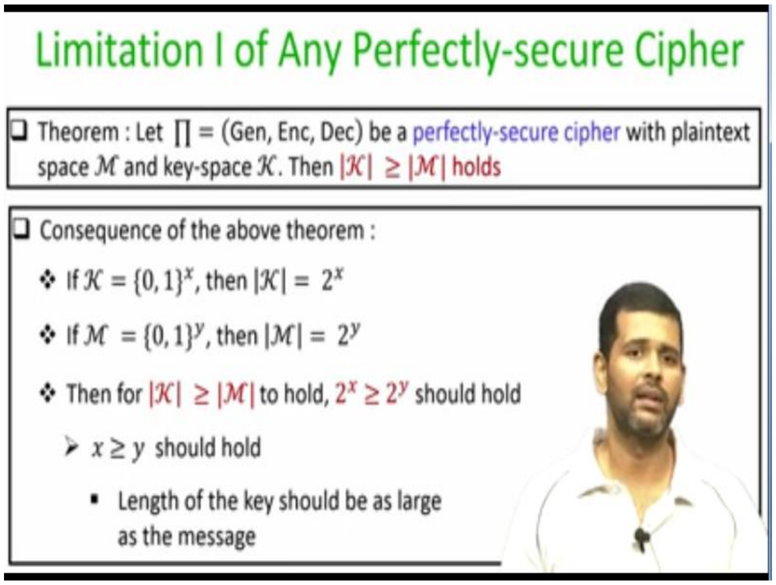
That means the number of candidate keys is less than the number of candidate plaintext. And for demonstration here is an example, I am assuming that my candidate plant text space consist of 3 plain text and my candidate key space consist of 2 possible keys. So now you consider the uniform probability distribution over the message space, that means you consider a scenario where sender could have encrypted any of these 3 possible messages with equal probability.

And you consider a ciphertext with whose probability of occurrence is non zero. So now, let me define the set M(c) which is the set of all valid decryptions of the cipher text c, namely M(c) consist of all plain text m from the plain text space, which I can obtain by decrypting the cipher text c under different candidate keys from the key space, right. So, again in this particular example, if I try to decrypt the cipher text c, I have 2 possible keys. On decrypting ciphertext by k_1, I will obtain one message and on decrypting cipher c by the second key k_2 I will obtain second message that means what I can say is that the cardinality of the set M(c) is upper bounded by the cardinality of the key space because that is the maximum number of distinct decryptions you can obtain by decrypting the ciphertext c here, and since as part of my assumption, the key space cardinality is less than the message space cardinality, what I obtained here is that the cardinality of M(c) is strictly less than the cardinality of key space which is less than the cardinality of message space.

What it means here is that there exists at least one message from the message space, plain text space, which can never lead to the ciphertext c that means on decrypting the cipher text, I will never obtain back that plain text. Again in this specific example, you can see that the decryption of c can never give you back m_3. And that is a violation of the perfect secrecy condition because the perfect secrecy condition says that if you have a uniform probability distribution over the plain text space, then for every cipher text c which occurs with non zero probability, it should be a potential encryption of all candidate messages with equal probabilities or in other sense if I see the original definitions of perfect secrecy, what we have arrived here is the fact that we have at least one message is m, where the probability of occurrence of the m is non zero, but the conditional probability that m is encrypted in the ciphertext c is 0.

That means these 2 probabilities are not same. And hence that is a violation of perfect secrecy right. So, that means we have proved that in any perfectly secure encryption scheme, the key space has to be as large as the message space.

(Refer Slide Time: 14:20)



Limitation I of Any Perfectly-secure Cipher

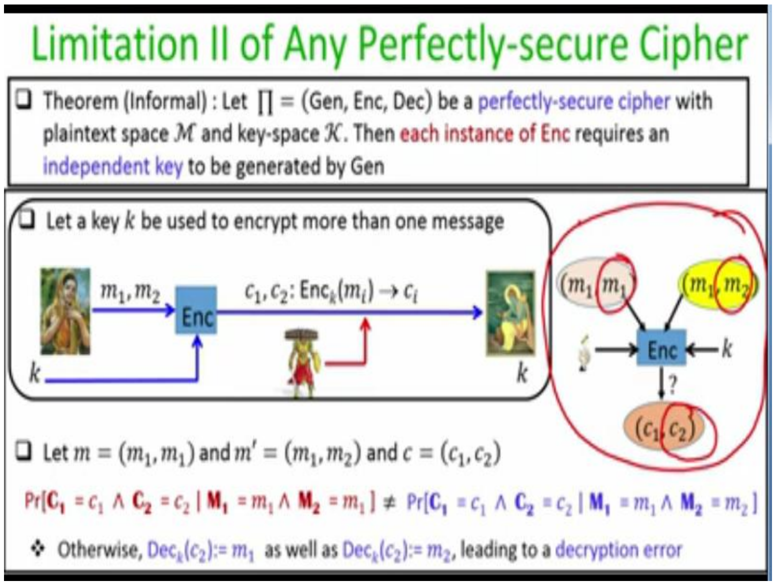
- Theorem : Let $\prod$ = (Gen, Enc, Dec) be a perfectly-secure cipher with plaintext space $\mathcal{M}$ and key-space $\mathcal{K}$. Then $|\mathcal{K}| \geq |\mathcal{M}|$ holds

- Consequence of the above theorem :
  - If $\mathcal{K} = \{0,1\}^x$, then $|\mathcal{K}| = 2^x$
  - If $\mathcal{M} = \{0,1\}^y$, then $|\mathcal{M}| = 2^y$
  - Then for $|\mathcal{K}| \geq |\mathcal{M}|$ to hold, $2^x \geq 2^y$ should hold
    - $x \geq y$ should hold
      - Length of the key should be as large as the message

Now, what is the implication of this theorem. Imagine your key space is the set of all possible strings of length x. That means, my encryption process supports encryption of x bit strings. That means the key space cardinality is nothing but 2^x. And in the same way, imagine my encryption process supports plain text space of y bit strings that means the plaintext have to be y bit strings and as a result, the cardinality of my message space is 2^y.

Now as per this theorem is my scheme is perfectly secure I need the cardinality of the key space to be at as large as the cardinality of the message space, that means 2^x should be greater than equal to 2^y. And as an implication of this, I get the fact that x should be greater than equal to y, that means the length of the key should be as large as the plaintext. That is the first limitation of any perfectly secure cipher, namely, the key should be as large the plaintext.

(Refer Slide Time: 15:24)



Limitation II of Any Perfectly-secure Cipher

- Theorem (Informal) : Let $\prod$ = (Gen, Enc, Dec) be a perfectly-secure cipher with plaintext space $\mathcal{M}$ and key-space $\mathcal{K}$. Then each instance of Enc requires an independent key to be generated by Gen

- Let a key k be used to encrypt more than one message

- Let $m = (m_1, m_1)$ and $m' = (m_1, m_2)$ and $c = (c_1, c_2)$

$$\Pr[\mathbf{C_1} = c_1 \wedge \mathbf{C_2} = c_2 \mid \mathbf{M_1} = m_1 \wedge \mathbf{M_2} = m_1] \neq \Pr[\mathbf{C_1} = c_1 \wedge \mathbf{C_2} = c_2 \mid \mathbf{M_1} = m_1 \wedge \mathbf{M_2} = m_2]$$

- Otherwise, Dec$_k(c_2)$:= m_1 as well as Dec$_k(c_2)$:= m_2, leading to a decryption error

Now, let us prove the second limitation of any perfectly secure cipher. And I would not be proving it rigorously and formally, I will just give you a very high level detail of the proof here. So the theorem states that imagine you are given a perfectly secure cipher over the plain text space m and key space k, then each instance of the encryption process of this cipher should require an independent key to be generated by the key generation algorithm. That means you cannot retain the same key for encrypting more than one message. So now let us again try to prove this you using a contradiction.

So imagine a scenario where sender retains the same key k for encrypting 2 messages m_1 and m_2 in sequence, right. So it has encrypted message m_1 producing ciphertext c_1 communicated over the channel, and then it has to use the same key k for encrypting a second message in the sequence. And again the ciphertext is communicated over the channel. And adversary has intercepted both the ciphertext and adversary is aware of the fact that the same unknown key k has been retained and used to encrypt both the messages m_1 as well as m_2.

What we are going to prove is that if something of this sort has happened, then you can never achieve perfect secrecy, right. So imagine 2 different sequences of plain text. The first sequence is when the the first message is m_1 and the second message is also m_1 and in the second sequence m', the sequences of messages are m_1 and m_2, where m_1 and m_2 are different. And imagine that adversary has intercepted a cipher text consisting of 2 sequences of cipher text where the first cipher text is c_1 and the second cipher text c_2, right.

So, as per the requirement of perfect secrecy, if the adversary has obtained a sequence of ciphertext being c_1 and c_2, and if adversary has a prior information that these 2 sequences of messages, which could have been encrypted in c_1 c_2 could be either m_1, m_1 or m_1, m_2, then the requirement of perfect secrecy is that with equal probability the ciphertext sequence c_1 , c_2 should be any encryption of m_1, m1 as well as it should be a potential encryption of m_1, m_2 right.

That means, if I consider the left hand side probability here then I have introduced here 2 random variables $\mathbf{c}_1$ and $\mathbf{c}_2$ in boldface, which denotes the value of the first cipher text sequence and the second cipher text sequence. And in the same way, I have introduced 2 random variables, boldface $\mathbf{m}_1$ and boldface $\mathbf{m}_2$, to note the candidate first plaintext and the candidate second plaintext. So my claim here is that the probability that your left hand side probability and your right hand side probability can never be same, which is a violation of perfect secrecy.

Pictorially the requirement of perfect secrecy should be that it does not matter whether the first message sequence m_1, m_1 is encrypted or whether the second sequence of message m_1, m_2 is encrypted using the key as per your encryption process with equal probability it should lead you to the cipher text sequence c_1, c_2, if at all your encryption processes perfectly secure.

But if that is the case, that means imagine a scenario where indeed the message sequence m_1, m_1 and a message sequence m_1, m_2, both leads to an encryption cipher text sequence c_1, c_2 with equal probability under the same and key k, then it means a decryption error. Because that means that if you decrypt the cipher text c_2 to using the key k, then it should be leading you back to the plain text m_1 as well as it should lead you back to the plain text m_2, which means that there is a correctness error in your encryption process.

That means there is a decryption error in your encryption process, which is a violation of the fact that your encryption process is secure, because one of the requirements of any secure cipher is the correctness requirement, which demands that your description should be unambiguous, but if we are in a scenario like this, where both the message sequence m_1 m_1 and message sequence m_1 m_2 leads to the same cipher text sequence c_1, c_2. Then that means if I just decrypt the ciphertext sequence c_2 it could lead me back to m_1 as well as it could lead me back to m_2. That means there is an ambiguity in the decryption of my original encryption process itself, which is a contradiction, this proves informally the theorem that each instance of the encryption process should run of fresh instance of the key generation algorithm right.

So that brings me to end of this lecture. Just to summarize. In this lecture we have discussed a candidate, perfectly secure encryption process, namely one time pad or Vernam cipher and we proved its perfect secrecy. We also saw the 2 restrictions imposed by one time pad. And we argued that these 2 restrictions namely the key size being as large as message and fresh key for each instance of the encryption are inherent to any perfectly secure encryption process. I hope you enjoyed this lecture. Thank you.