

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

Lecture-02
Symmetric Key Encryption

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

(Refer Slide Time: 00:36)

Roadmap

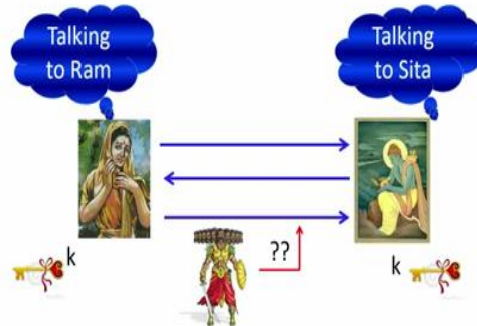
- ❑ Fundamental problems addressed by cryptography
 - ❖ Key agreement problem
 - ❖ Secure Communication problem
- ❑ Types of cryptographic primitives
 - ❖ Symmetric-key primitives, Asymmetric-key primitives
- ❑ Symmetric-key encryption scheme
 - ❖ Syntax
 - ❖ Informal security definition
- ❑ Kerckhoffs' Principle

We will discuss the fundamental problems addressed by cryptography namely the key agreement problem and a secure communication problem. Then we will discuss the various types of cryptographic primitives namely the symmetric key primitives and asymmetric key primitives. And then we will start our discussion on symmetric key primitives where we will see the syntax of symmetric encryption mechanism and informal security definition and then we will finish the lecture with Kerckhoffs' principle. Right!

(Refer Slide Time: 01:05)

Core Problems Addressed by Cryptography

❑ Problem I: Key agreement



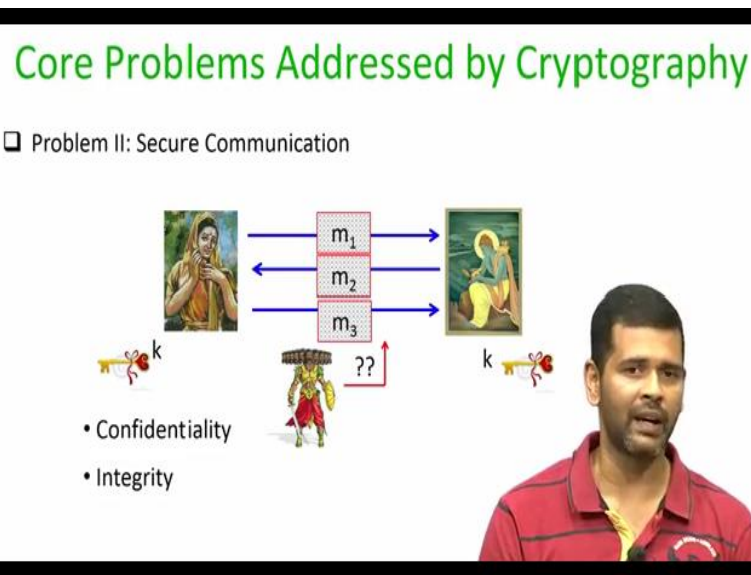
So, if you remember in the last lecture we discussed that the central problem addressed by cryptography is that of secure communication. And it turns out that secure communication is actually solved by solving 2 core problems and the first core problem is that of key agreement. So, in the key agreement problem, the scenario is the following. We have 2 entities a sender and the receiver who do not know each other and who do not have any kind of pre shared information.

And they are meeting for the first time and the requirement for key agreement protocol is that as for the protocol the sender and the receiver should talk to each other over a public channel and it should be guaranteed to the sender that indeed it is talking to the Ram receiver and vice versa. And at the end of the protocol both sender and receiver should output a common key k . The security requirement from the key agreement protocol is that even though the sender and the receiver are talking publicly and if there is a third party who is actually eavesdropping the communication or noting down the entire communication which is happening between the sender and the receiver, then the third party should not be able to figure out what exactly is the key which sender and receiver are going to output at the end of the protocol. So, it is like saying the following that if I consider a class and the key agreement protocol should require that if I want to agree upon a key with a particular student, then using the key agreement protocol, I should shout something in the class.

And in response that particular student should re shout back something to me and using whatever I communicated to him and whatever he communicated back to me, both of us should be able to compute a common key and no one else in the class who have actually seen the actual messages that I have communicated to that particular student and that student has communicated to me should not be able to figure out what exactly is the key that myself and that particular student are going to output.

That is the first core problem addressed by cryptography. And this will be the theme of the second half of the course, where we will see how using public key cryptography which we can solve this key agreement problem.

(Refer Slide Time: 03:19)



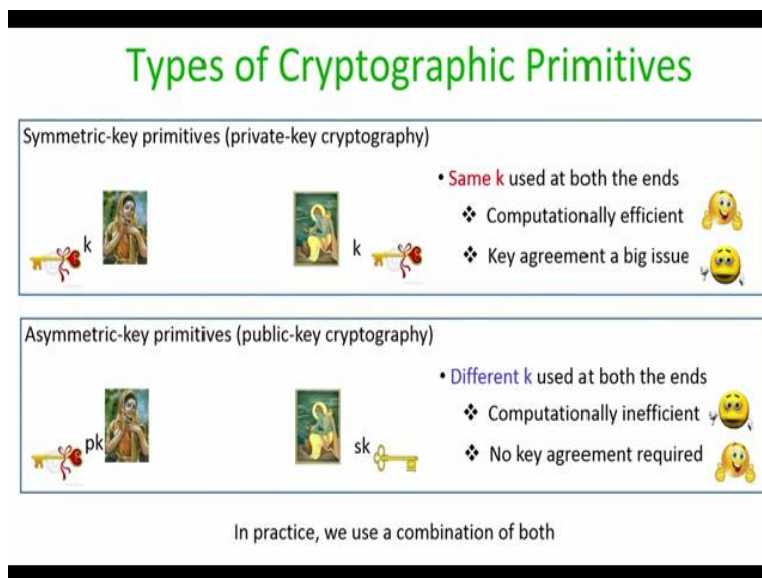
The second core problem which is addressed by cryptography is that of secure communication. And here the setting is as follows. We imagine that sender and receiver now have a secret key k which has been agreed upon by some magical mean in this context, actually, by running a key exchange protocol. So we imagine that sender and receiver have executed a secure key exchange protocol, and as a result, they have a common key already available with them.

Now using the common key, the goal of the sender and the receiver is to talk over a public channel such that the confidentiality of the communication and integrity of the communication is ensured. By confidentiality I mean any third party who do not have the knowledge of the key

should not be able to figure out what exactly are the contents, which sender and receiver are exchanging to each other.

And by integrity, I mean that if there is a third party who actually intercept some of the communication and tries to change the contents of the messages, then it should be detectable at both the ends. So that is the second problem addressed by cryptography and this will be the theme of the first half of the course, namely symmetric cryptography, where we will assume that somehow the sender and the receiver have already agreed upon the key and our goal will be to solve the problems of confidentiality and integrity.

(Refer Slide Time: 04:39)



So, roughly in cryptography, we use 2 kinds of primitives. The first kind of primitives are the symmetric key primitives, which are also called as private key primitives. And here the same key is going to be used by both the sender as well as the receiver. That is why the name symmetric key primitives or private key cryptography. Symmetric because it is symmetric in nature. The same key is getting used at both the senders end and as well as the receivers end.

And the name private key cryptography because the key k which is common to both the sender and the receiver is going to be private, it would not be available in the public domain. Now, there are both pros and cons of these kinds of primitive. The good side of these primitives is that they are computationally very efficient in practice. And the downside of these primitives is that the

key agreement is going to be a big issue. That this all these primitives works under the assumption that the common key has been already agreed upon by some magical mechanism, and which is a big assumption. So ensuring that both sender and receiver have the same key is the challenging task. That is a downside of these primitives.

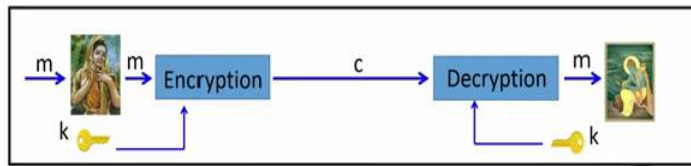
The second kind of primitives are called as asymmetrical primitives or public key cryptography primitives where the primitive is operated with 2 keys. One of the keys which is the public key will be available in the public domain whereas the other key namely, the secret key will be available only to one of the entities or in the context of encryption scheme, it will be available only with the receiver. That means this has asymmetry nature. That is why the name is asymmetric key primitive. And the reason it is called public key primitive is that one of the keys is available in the public domain.

Again, it has these kinds of primitives have both plus as well as the downside. The good side about these primitives is that you do not need any kind of key agreement. That means if I take a public encryption scheme for instance, then there is no requirement to do any kind of key agreement. Whoever wants to encrypt a message for me, it can take my public key which will be available in the public domain. So I do not need to explicitly agree upon a key with that particular sender.

The downside of these kind of primitives is that they are computationally inefficient that means the amount of computation which are involved in this kind of primitive is of several orders of magnitude compared to the computation required in the symmetric key primitives. In practice, we use a combination of both. So when we will be winding up our course and we will be discussing the real world protocols such as SSL, we will see that how we combine both these 2 primitives to get the best of the both worlds.

(Refer Slide Time: 07:19)

Private-key/Symmetric-key Encryption



- ☐ Key k shared in advance (by "some" mechanism)
- ☐ m is the plaintext
- ☐ c is the ciphertext (scrambled message)
- ❖ Symmetry: same key used for encryption and decryption



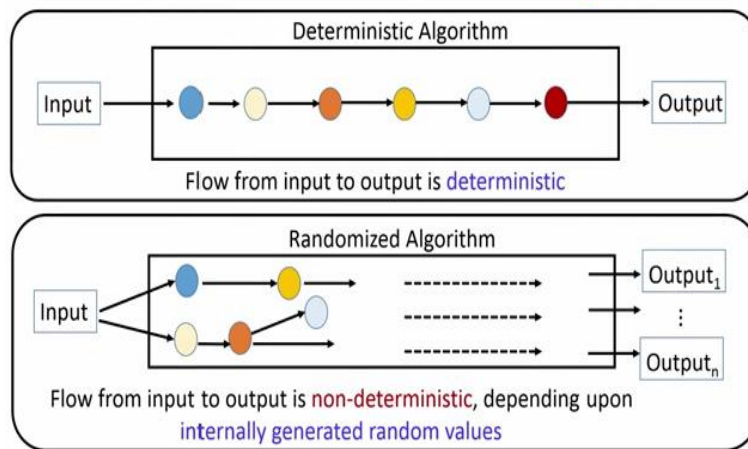
So now let us start our discussion on private key or symmetric encryption schemes. So, on a very high level, the goal is to following. The setting is as follows, we have a sender and the receiver and by some magical mechanism, we assume that a random key has been shared up agreed upon between the sender and the receiver. So I denote the key as k which is actually going to be a bit string. Now sender is interested to send some message m , which is again going to be a bit string, and in the cryptography jargon, we call this string m to be the plaintext.

Now using the key, sender is going to use an encryption algorithm, which takes the message and the key as input. And it produces another bit string denoted as C , which is the ciphertext or the scramble text, which is going to be communicated over a public channel to the receiver. At the receiving end, receiver is going to take the ciphertext which is a bit string, and the key which is the same key with which the sender has operated the encryption process and it is going to operate a decryption algorithm.

Now so the decryption algorithm is going to take the scramble text and the key and it is going to output the plain text which sender has actually encrypted using the encryption algorithm. So the reason this encryption mechanism is called symmetric encryption mechanism is that we have a symmetry that the same key is getting used both for the encryption as well as the decryption.

(Refer Slide Time: 08:51)

Deterministic vs Randomized Algorithms



So before going into the formal description of a symmetric key encryption process, let us first try to understand the difference between deterministic and a randomized algorithm. In a deterministic algorithm, the output is a deterministic function of the input. That means, if we consider the straight transition inside the algorithm, then the flow from the input to the output is always a deterministic function.

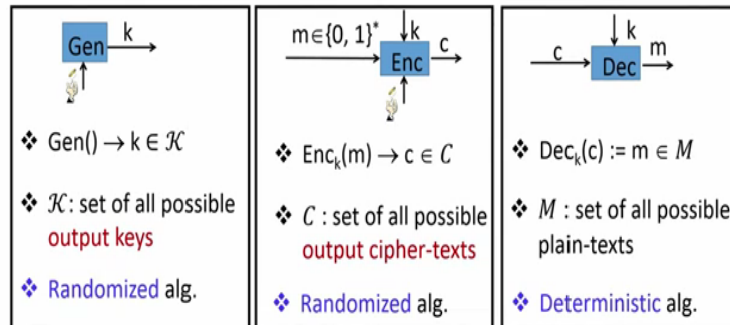
What I mean by this is, if I run a deterministic algorithm on an input x 100 times, I am going to get the same output y 100 times. I would not get different output that means the output Y is a deterministic function of the input x . Whereas in a randomized algorithm, the flow from the input to output is non deterministic. And the flow is going to be decided by the value of random bits strings which are going to be generated inside the algorithm, as part of the algorithm.

That means, in a randomized algorithm, it is not guaranteed that if I call that algorithm with the same input again and again, it is not guaranteed that I am going to get the same output. Output may depend upon the sequence of random bits, which I am going to generate inside the algorithm. And depending upon the value of those random bits, what path I followed inside the algorithm. So that is a very high-level difference between a deterministic algorithm and a randomized algorithm.

(Refer Slide Time: 10:17)

Syntax of Symmetric-key Encryption

□ A symmetric-key encryption scheme (cipher) is a collection of three algorithms (Gen, Enc, Dec)



□ Gen, Enc are preferably randomized, while Dec is always deterministic

Now, let us see the syntax of a symmetric encryption scheme. So, any symmetric key encryption scheme, which is also called as a cipher is going to consist of 3 algorithms. The first algorithm is the key generation algorithm, which is denoted by Gen. This algorithm is not going to take any external input. What this algorithm is going to do is as part of the algorithm inside there will be some random bit strings which are going to be generated, which I denote by this particular notation.

When I write this notation that someone is tossing the coin, I mean that inside the algorithm, random bit strings are going to be generated and based on the value of that bit string, that output is going to be determined. The output of this key generation algorithm is a key denoted by the symbol k . And since it is a randomized algorithm, every time I run the key generation algorithm, I am bound to get a different output, I would not get the same output.

The syntax that we use to denote the key generation algorithm is the following. We say that the key generation algorithm Gen, it does not take any input. So that is why the brackets are left is empty because it does not take any external input. And it is going to output a key which I denote by k . And this k belongs to a larger set, this calligraphic \mathcal{K} , which is the set of all possible keys which is key generation algorithm can output.

And the important thing here is that since this algorithm is a randomized algorithm, that is why I do not say that the output of Gen assigned a value key, instead I say that the output of Gen is going to take one of the possible value k from the set of all possible keys. I am not using the assignment operator because it is not a deterministic function. It is a randomized function it is a randomized algorithm.

So here this calligraphic K (the capital K) is the set of all possible keys which the key generation algorithm could output. For instance, if we know that the key generation algorithm is going to output a 256 bit key, then I know that this calligraphic K is the set of all possible 256 bit strings, namely, the key space K is 2 to the power to 256. And as I said earlier, this key generation algorithm is going to be a randomized algorithm.

The second algorithm in any symmetric key encryption scheme is the encryption algorithm denoted as Enc. And it is going to take 2 external inputs namely the plain text which the sender wants to encrypt, which is a bit string which I denoted by the symbol B , and the key k , which the key generation algorithm would have output. And apart from these 2 inputs, it has an internal input namely the internal random coins, which are generated or the random bits which are generated as part of this encryption process.

And as a result, this encryption algorithm is a randomized algorithm. So, based on the message, the key space and the random bits which are generated inside the encryption algorithm, the encryption algorithm is going to output a ciphertext denoted as c . So, as I said that since the encryption algorithm could generate internal random bits to decide the outcome c , it is a randomized algorithm.

And the syntax that we use to denote the encryption algorithm is the following: we say that the encryption of the plain text m (within the brackets I am writing the message m as the input) is the external input with this encryption function and the key k is put as a subscript. I will say that the message m is encrypted under the key k . And since this algorithm is a randomized algorithm, I am not using the assignment operator to denote the output of this encryption algorithm.

Instead, I am saying that the output of this encryption algorithm is going to be one of the possible ciphertext from the set of all possible ciphertext which your encryption algorithm could produce. Since this encryption process is a randomized algorithm, what it means is that even if I call this encryption process with the same value of m and the same value of k multiple times, I am going to get different ciphertext because every time I call this encryption process, the set of random bits strings which are generated as part of the algorithm will be different. And since c is going to be a function of the message, key and the internal random bits, value of c will be different for different invocations of c . So that is the syntax of encryption algorithm.

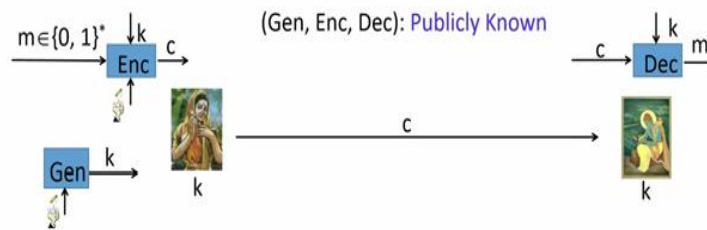
Now, the decryption algorithm takes the ciphertext c as the external input and the key k which has been generated by the key generation algorithm, in return it produces the plaintext m . The syntax I follow to denote the decryption algorithm is the following: we say that decryption of the input c so here c is the external input, which is fed to the decryption function along with the key k . I will say that the decryption of the ciphertext c under the key k is going to produce the message m .

The message m belongs to the largest set of possible plaintext namely M , which is the plain text space. Notice that my decryption algorithm here is a deterministic function. There are no random bits strings which are generated inside the decryption algorithm to decide the outcome m . As a result of that, I am not using the arrow notation to denote the outcome of decryption. Instead, I am using the assignment operator to denote the outcome of the decryption function. What I mean by that is if I call the decryption algorithm multiple times with the same value of k and the same value of c , I am bound to get the same m again and again, I would not get different m for the different invocations of Dec_k on the same value of c and k . In that sense, it is a deterministic function. As a result, we use the assignment operator to denote the output of the Dec function.

So that is the syntax of your key generation algorithm, encryption algorithm and decryption algorithm. Notice that we required the key generation algorithm and encryption algorithm to be randomized whereas the Dec algorithm should be deterministic, and there is a reason for that, which we will be able to appreciate when we discuss the various attack models in this lecture.

(Refer Slide Time: 17:09)

How to (Typically) Use a Symmetric Cipher



- ❑ Execute Gen and share the output key k at the beginning of the session
- ❑ Encrypt the plaintext using k and send the ciphertext c over the public channel
- ❑ Decrypt the received ciphertext c using the key k

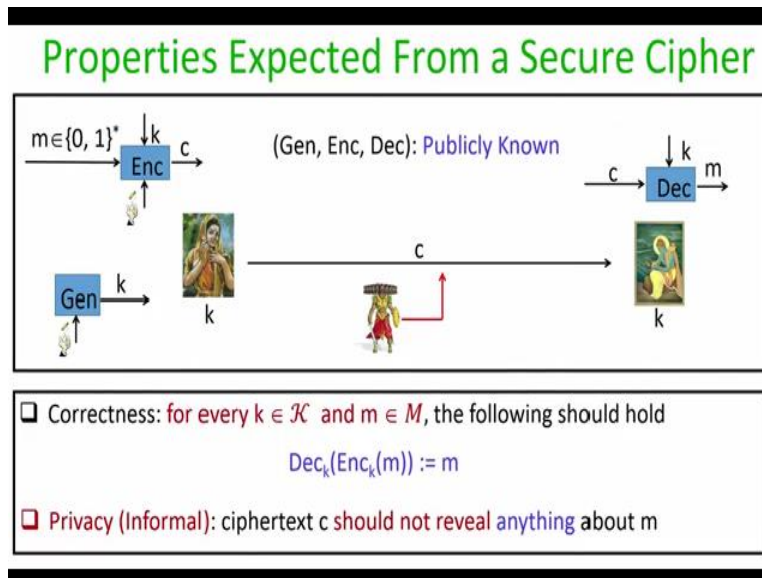
Now how to typically use a symmetric cipher. Imagine we have a symmetric key encryption scheme. Namely, we have a collection of key generation algorithm, encryption algorithm and decryption algorithm. And we assume that the steps of this key generation, encryption and decryption algorithms are publicly known. That means, the steps are known even to the sender and the receiver or to any third party in this world.

Now, to use this symmetric cipher, what sender is going to do is, it is going to run the key generation algorithm, which is going to output one of the candidate keys namely k from the key space. This key will be agreed upon with the receiver by some magical mechanism at the beginning of the session. So, we imagine that at the beginning of the session sender runs this key generation algorithm. In that session there are several messages, which sender would like to encrypt and communicate to the receiver using this k . The first step of the session will be the key generation algorithm and the agreement of that key with the receiver by some magical mechanism.

Now once the key is agreed upon, every time sender wants to encrypt a plain text m using this key what is going to happen is the following, it is going to run the encryption algorithm to produce the cipher text. The cipher text will be communicated over the public channel, through which sender is communicated to the receiver. As and when the receiver receives the ciphertext what it is going to do is receiver will be knowing what encryption process has been used by the

sender. So it will know the corresponding decryption process and not only that, it will also know the key using which the sender has operated the encryption process. Using the same key, it will operate the decryption process and will recover the plain text. That is how typically we are going to use a symmetric key encryption process.

(Refer Slide Time: 18:57)



Now what are the properties that we need from any secure encryption scheme or a secure symmetric key encryption scheme. Roughly we need 2 properties. The first property is the correctness property, which says that, for any key k which the key generation algorithm has output and for any plain text m which has been encrypted by Enc , the following condition should hold. If I encrypt a plain text m under the key k to obtain a ciphertext c and if I decrypt that cipher text c using the decryption process under the same key k , I should get back the original m . That is the correctness property.

To give you the analog if you have a physical lock, and if I have 2 copies of the key, what this correctness property demand is that if I actually lock that key using the key and send that lock in the lock condition to you. Then, if you have also the same key and if you try to open that lock in the lock position using the key that you also own, you should be able to open the lock from the lock condition to the open condition. That is roughly the analogy you can imagine from this correctness requirement.

The second important property which we need or expect from a secure symmetric encryption process is that of privacy. And now we are going to see that what are the challenges we face when we try to formalize the privacy requirement. Intuitively, when we say that a symmetric key encryption process is secure, or it achieves a privacy property, I mean, that by seeing the ciphertext c the adversary or the bad guy who has observed the ciphertext c should not be able to compute anything about the message m .

That means imagine there is a third party who is the bad guy who has intercepted the ciphertext who knows the details of the key generation process, the encryption process and the decryption process. The only thing that bad guy does not know is the value of key with which sender has operated the encryption process. The privacy property demands informally, that even after this knowledge of ciphertext, should not give any information about the message m .

(Refer Slide Time: 21:11)

Challenges in Formalizing the Privacy Definition

- ❑ **Definition 1:** An encryption process is secure if the ciphertext does not reveal the underlying key
 - ❖ Consider an Enc algorithm which always outputs plaintext as the ciphertext
- ❑ **Definition 2:** An encryption process is secure if the ciphertext does not reveal the underlying plaintext
 - ❖ Consider an Enc algorithm where the first 1% of the ciphertext is the same as the first 1% of the plaintext

However, it turns out that if we want to formalize the privacy definition, there are several challenges that we face. So what I am going to do next is I will propose a series of definitions to formalize the privacy condition. And then we will see a potential loophole in each of these potential definitions, which is going to show you that how difficult it is indeed to come up with a right definition of privacy.

My first candidate definition to formalize the privacy definition is the following. I will say that an encryption process is secure if the ciphertext does not reveal the underlying key. The intuition behind this definition is that if the key is revealed to the adversary, then it can find out any subsequent message which has been encrypted under that key. The minimum requirement which I require from any private encryption scheme is that the ciphertext should not reveal the underlying key. Well it turns out that the requirement is definitely meaningful from any secure encryption process. But this definition is completely useless.

For example, consider an encryption algorithm, which always outputs a cipher text, which is same as the plain text, that means the cipher text does not depend upon the key at all. And the value of the cipher text is the same as the plain text. If you see this definition, and this candidate encryption process, definitely the encryption process is not revealing anything about the key. As per this definition, you can label this encryption algorithm as a secure encryption algorithm. But this kind of encryption algorithm is completely useless to use in practice because it is completely revealing your plain text.

So now let us rectify this definition. And let me come up with the second candidate definition to formalize the privacy definition. My definition 2 is: I will say an encryption process is secure if the ciphertext does not reveal anything about the underlying plaintext, because that is what is the basic intuition of the privacy. The problem with this kind of definition is what do you mean by ciphertext does not reveal the underlying plaintext? How much it should reveal, how much it should not reveal?

For example, you might have an encryption process, where 99% of the plain text is not revealed by the ciphertext. But unfortunately, the ciphertext might be revealing 1% of the plain text and that 1% of the plain text which is getting leaked by the cipher text might be the critical information which you want to hide. Again, this definition and this candidate encryption process might look like satisfying the definition 2 but actually such kind of encryption algorithm I cannot use in practice. The problem with this definition is I am not exactly specifying what it means by revealing and not revealing and how much to reveal and how much not to reveal.

(Refer Slide Time: 24:05)

Challenges in Formalizing the Privacy Definition

□ Definition 3: An encryption process is secure if the ciphertext does not reveal any character of the underlying plaintext

- ❖ Consider an Enc algorithm where the ciphertext reveals whether the underlying plaintext is less than or greater than a certain value

□ Definition 4: An encryption process is secure if the ciphertext does not reveal any meaningful information about the underlying plaintext

- ❖ The notion of meaningful information varies from application to application

So to fix this definition, let me propose the third potential definition to formalize the privacy definition. The definition 3 says that an encryption process will be considered secure if the ciphertext does not reveal any character of the underlying plaintext. So this will take care of the potential bug which was there in our definition 2 or the candidate encryption scheme that we proposed to violate definition 2.

Because now, even if 1% is revealed, it is not going to satisfy definition 3. But again, there is a potential loophole in this definition. Consider an encryption algorithm where ciphertext does not indeed reveal any of the underlying characters of the plain text. But a cipher text might reveal the range of the plain text namely it might reveal whether the plain text is less than a particular threshold or whether it is greater than a particular threshold.

Again, if I view the definition, this candidate encryption scheme indeed satisfies the property because no character of the underlying plaintext is revealed. But what is getting revealed is whether the plaintext is less than a certain value or greater than a certain value. And that might itself be a security breach. So I cannot afford to use such kind of encryption algorithm to encrypt sensitive data. Because if the ciphertext is going to reveal the range of the sensitive data, I might be in trouble.

So let us try to fix this potential problem in definition 4 and the definition 4 says that an encryption process will be considered as secure if the ciphertext does not reveal any meaningful information about underlying plaintext. That means it should hide not only the underlying plain text, it should also hide whether the message is less than a certain value greater than a certain value and so on.

Well, intuitively this definition is good. But the problem here is what exactly constitutes a meaningful information varies from application to application. For certain application, the underlying characters of the plain text might be the sensitive information, for another application whether the message is less than a certain value or greater than certain value that might be the meaningful information, and so on. You cannot exhaustively list on what constitutes a meaningful information for a particular application. That is a downside of this definition. So, that is as a result, I cannot take it as a meaningful definition of privacy.

(Refer Slide Time: 26:25)

Challenges in Formalizing the Privacy Definition

- ❑ **Definition 5:** An encryption process is secure if the ciphertext does not help to compute any function of the underlying plaintext
 - ❖ Precisely what we expect from a secure cipher. But
- ❑ There are certain **loopholes** in the above definition
 - ❖ **How to formalize** whether a given ciphertext helps to compute a given function of the underlying plaintext ?
 - ❖ What is the **underlying adversary/attack model** ?
 - Is the adversary **passive or malicious** ?
 - Does the adversary have access to any kind of **"additional information"** ?

So to fix that problem, let us come up with the next possible definition of the privacy. And this definition says that an encryption process will be called as secure if the ciphertext does not help the attacker to compute any function of the underlying plaintext. That means imagine an attacker who has seen the ciphertext who knows the encryption process. And the attacker is interested to compute some function, say F of the underlying message which sender has actually encrypted in the ciphertext. We will say the encryption process is secure, if using the knowledge of the cipher

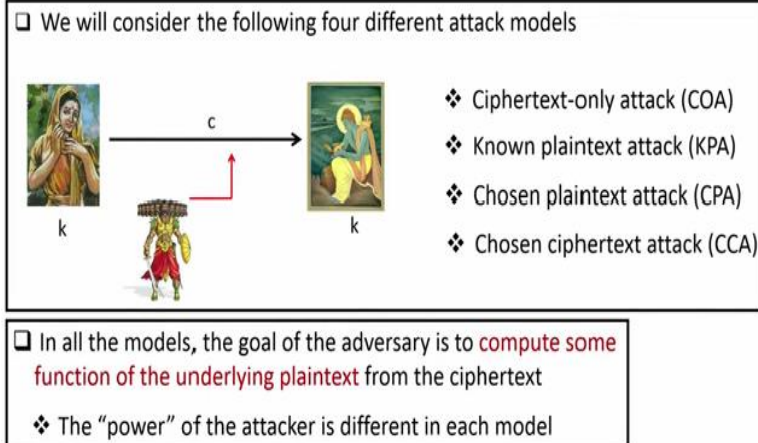
text, adversary is not able to compute the function of the underlying message. Well, this is precisely what we expect from a secure cipher.

But still there are certain loopholes in this definition. Namely, there are 2 loopholes. The first loophole is that how do you formalize whether a particular cipher text has helped the adversary to come to a function of the underlying plain text or not. How do you judge that, how do you mathematically formalize that? That is the first loophole in this definition. And the second loophole in this definition is what exactly are the capabilities of the adversary you are considering, that is not specified in this definition. That means whether you are considering an eavesdropper adversary who is simply observing the ciphertext and trying to compute a function, or whether you are considering an adversary or malicious adversary, who could change the contents of the ciphertext and see the behavior of the or the other response of the receiver and then trying to compute the functions of the underlying messages.

Also, this definition does not specify is the adversary also provided with any kind of additional information that means, what are the various capabilities of the adversary which you are considering. So even though intuitively definition 5 is the right definition, the 2 potential shortcomings of this definition is the exact formalization of whether the ciphertext is helping the adversary to compute any function of the underlying plaintext and what precisely is the attack model or the adversarial model you are considering.

(Refer Slide Time: 28:33)

Various Attack Models



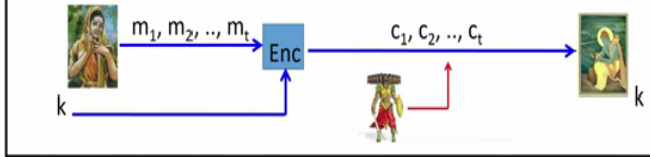
That brings us to the discussion of various attack models which we consider in cryptography. So, typically in cryptography we consider the following 4 attack models. The first attack model is the ciphertext only attack or the COA attack model. The next attack model is the known plaintext attack model or KPA model. The next attack model is the chosen plaintext attack or CPA model. And a final attack model is the chosen ciphertext attack model or CCA attack.

In all these attack models, the scenario is the same. We have a sender and the receiver who have agreed upon some common key, by running the key generation algorithm and agreeing the key with other entity by some magical mechanism such that the adversary or the bad guy or the attacker is not aware of the key and the adversary has intercepted some ciphertext. The goal of the adversary is to compute some function of the underlying plain text which has been encrypted in this c . What differs in this attack models is the power of the attacker that means what kind of additional information apart from the ciphertext is available to the attacker. So what we are going to do next is, we will go through each of these attack models and see what are the capabilities of the attacker.

(Refer Slide Time: 29:45)

Ciphertext-only Attack (COA)

- ❑ Simplest possible attack



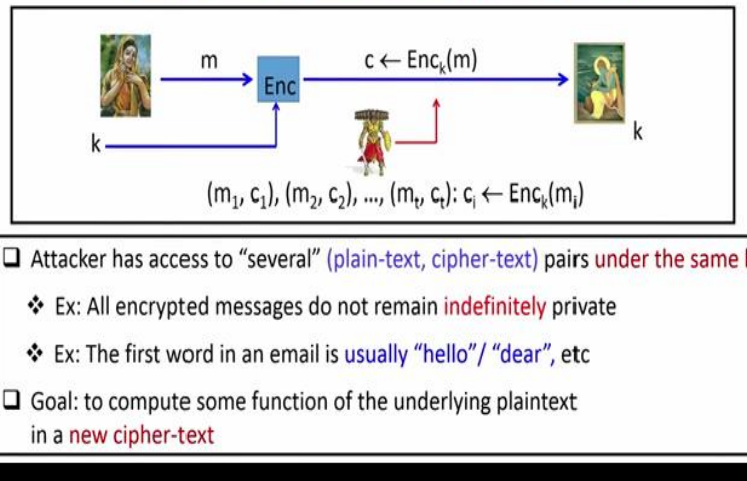
- ❑ Attacker has access to a ciphertext (ciphertexts)
 - ❖ **Passive** in nature
- ❑ Goal: adversary wants to compute some function of the underlying plaintext(s)

Let us start with the ciphertext only attack model or the COA attack model, which is the simplest possible attack scenario. And here the scenario is the following. We have the sender and the receiver who have agreed upon a common key not known to the attacker and sender has encrypted several messages using the same k as per the encryption process and the details of the encryption process is known to the attacker.

The attacker has eavesdropped and intercepted the ciphertext. The attacker does not have to do anything fancy here; it just has to listen over the channel what cipher text contents have been communicated. So that is why this attack is passive in nature. And the goal of the attacker here is to compute some functions of the underlying plaintext. Apart from the cipher text knowledge, no additional information is available to the attacker in this attack model.

(Refer Slide Time: 30:33)

Known Plaintext Attack (KPA)



The next attack model is the KPA attack model which is slightly stronger attack model than the COA attack model because here the adversary is available with some additional information compared to the previous model. The additional information which is available to the adversary is the following. The adversary is available with a collection of (messages, ciphertext) pairs here namely, message (m_1, c_1) message (m_2, c_2) and message (m_t, c_t) .

That means it got a database of several (message, ciphertext) pairs where each of the ciphertext in the pair is an encryption of the corresponding message in the pair under the same unknown key. That is important here, all the messages in the ciphertext are under the same unknown key which is not known to the attacker. Now, you might be wondering, what exactly, how exactly it is possible for an attacker to come up with such pairs of (message, ciphertext) in real life.

So there could be several scenarios through which it is possible for an attacker to get access to such (message, ciphertext) pair. For instance all encrypted messages do not remain private indefinitely or if you consider the message to be email exchange, then the first message in any email is usually the message hello dear hi, etc. In that sense, the encryption of certain messages under the unknown key is going to be revealed to the attacker.

In this attack model, we assume that attacker has already got a collection of several (message, ciphertext) pairs. And now a fresh message has been encrypted by the sender under the same key

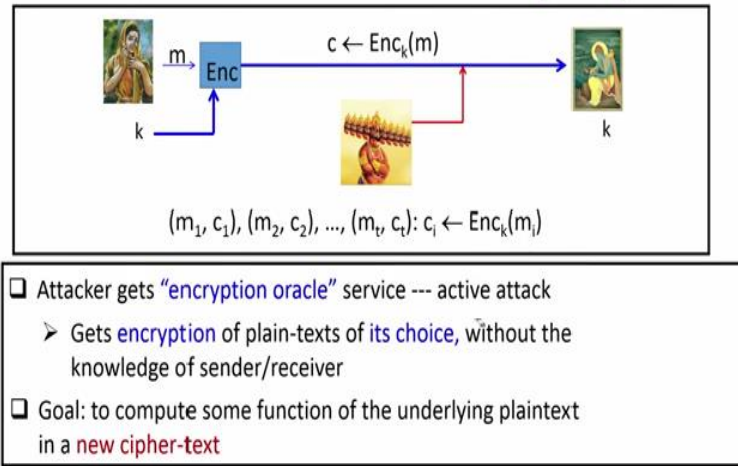
k and communicated over the public channel. And the goal of the attacker is to compute some function of this message m using the help of the ciphertext. I stress that the message m the fresh message, which is now getting encrypted by the sender might already belong to the collection of (message, ciphertext) pairs which adversary possess.

This shows the importance of your encryption process to be randomized. So recall when we discuss the syntax of symmetric encryption process, I stress that the encryption process should be randomized. That means even if I encrypt the same message under the same key, I should obtain a different ciphertext. If you do not use a randomized encryption process, but instead use a deterministic encryption process and if the same message is getting encrypted multiple times, then just by observing whether the cipher text gets repeated over the channel, adversary could come to know whether the same message is getting encrypted or not.

In this specific case, if the fresh message m , which sender is actually communicating to the receiver is already present in the list and if my encryption processes is a deterministic encryption process. Then this C is going to be already present in the list of (message, ciphertext) pair and adversary could easily find out the exact contents of the fresh message. That is why I need my encryption process to be a randomized encryption process. So that is the KPA attack model. And as you can see, this is a more powerful attack model. Because not only the adversary is available with the encryption of the message m , he is also available with the encryption of several earlier messages with the encryption of several prior messages under the key k .

(Refer Slide Time: 33:55)

Chosen Plain-text Attack (CPA)



The next attack model is a more powerful attack model, and this is called the chosen plaintext attack model or the CPA attack model. The scenario here is the following. We have a common key k agreed upon between the sender and the receiver by some magical mechanism. And now we assume here that the adversary gets an encryption oracle service to the encryption box. What I mean by that is the adversary can force or it can influence one of these 2 parties.

Say for example, the sender to encrypt any message of adversary's choice under the key k . And importantly, the sender who is actually providing the encryption oracle service would not be aware that actually it is encrypting messages of adversary's choice and providing those ciphertext to the adversary. So that is why we say, or we model this kind of service as an encryption oracle because adversary won't be knowing the value of the key. But still it will be able to get or see the encryption of messages of any message of its choice under that unknown key.

It can get the encryption oracle service for any number of messages as long as it is polynomially bounded or it is practical in nature. That means, it should not be possible for the adversary to get the encryption oracle service for an infinite amount of time. He should be able to get encryption oracle service only for a limited amount of time or for a practical amount of time. So, we assume that it gets, or it interacts with the encryption oracle service and submit messages of its choice i.e. the plain text of his choice and c is the corresponding ciphertext and it builds upon a

dictionary of (message, ciphertext) pairs. In this particular case, the message (m_1, c_1) , (m_2, c_2) and the message (m_t, c_t) where all these ciphertexts are the encryptions of the corresponding message under the unknown key k .

So, the kind of information that is available to the attacker here is same as in the KPA and in KPA model also adversary was provided with several (message, ciphertext) pairs under the unknown k . Here in the CPA model also adversary is available with several (message, ciphertext) pairs under the unknown key k . The difference here is the messages in the adversary's database in the CPA model is now under the control of the attacker.

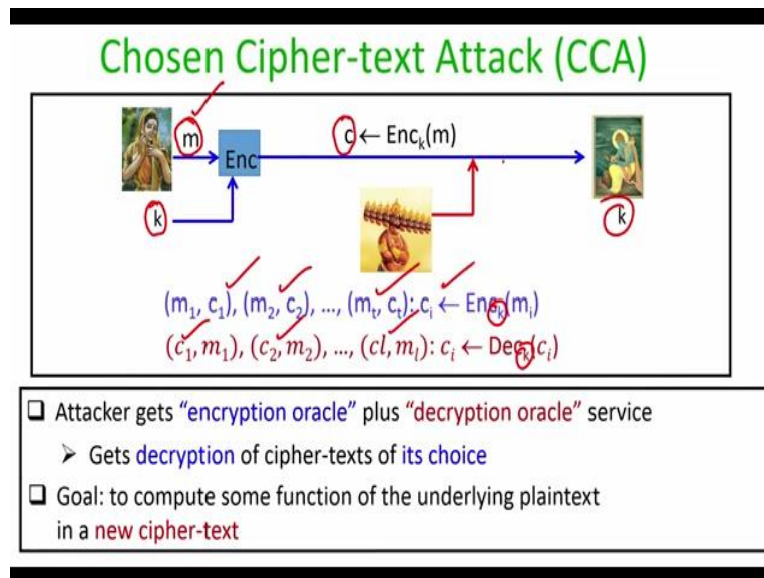
That means if adversary, depending upon the context of the underlying application is already aware that these are the potential messages which sender might encrypt in the future. The CPA model allows the adversary to see encryption of those messages in advance by getting an encryption oracle service. Now, you might be wondering that how in reality it is possible for an adversary to influence the sender that "please encrypt this messages for me" without actually sender knowing that she or he is providing the encryption oracle service to the attacker.

When we will be discussing the CPA attack model in some of our subsequent lectures, we will see how in reality it is possible to launch CPA attack. For now, assume that adversary is already available with this (message, ciphertext) pairs. The goal of the adversary is the following: a fresh message has been encrypted by the sender under the same unknown key k and has been communicated over the secure open channel to the receiver and adversary has intercepted this ciphertext.

The goal of the attacker is to compute some function of the message which has been encrypted in this fresh ciphertext. I stress here that the message which is freshly encrypted might already belong to the database of (message, ciphertext) pairs, which adversary possess. I also stress that the CPA attack model is active in nature that means to get this encryption oracle service, the adversary not only has to listen over the public channel between the sender and the receiver.

It also need to change the contents of the ciphertext which are getting communicated between the sender and the receiver and see the response of the receiver or the sender to get the encryption oracle service that means these kinds of this particular type of attack is no longer an eavesdropping attack is now an active attack. So that is the CPA attack model for you.

(Refer Slide Time: 38:14)



Now let us go to the next attack model, which is the strongest possible attack model, which we call as the CCA attack model. And here adversary can now get 2 kinds of oracle services, it has got access to the encryption oracle service that means it can ask for encryption of any message of its choice by influencing the sender or the receiver plus it can get the decryption oracle services as well from any of the 2 parties say for example, from the receiver.

What I mean by the decryption oracle service is that it can submit any ciphertext of his choice because the adversary might be already aware that what could be the candidate ciphertext that your encryption process might produce. So it can come up with any candidate ciphertext and submitted to the Dec_k , it can influence the receiver to decrypt the ciphertext for the adversary. In response, the receiver is going to decrypt those ciphertext and send back the corresponding plain text back to this adversary.

In that sense, we say that the adversary gets access to the decryption oracle. There is no restriction on the order in which the adversary could get the encryption and the decryption oracle

service it could see, or it could ask for the encryption oracle service for certain number of steps and then it could ask for the decryption oracle service for certain number of steps. There is absolutely no restriction on what messages he could get encrypted under the encryption oracle. And there is absolutely no restriction on what ciphertext it could get decrypted from the decryption oracle. We make no such restriction.

Now adversary has 2 kinds of database, the first database is what it gets from the encryption oracle service. This is nothing but the encryption of several messages of adversary's choice under the key k and a second database is the decryption of several candidates ciphertext of adversary's choice. And the important thing here is that all encryptions and decryptions are done under the same unknown key k , that is important.

Adversary is now more powerful here and the goal of the adversary is to following: imagine sender has a fresh message which has been encrypted and adversary eavesdrops the cipher text c , the goal of the adversary is to compute some function of the underlying plain text m with the help of the database that adversary has already prepared, I stress that the fresh message the sender is sending might already belong to the list database that adversary produce. The goal of the adversary is to compute that message m which has been freshly encrypted using the help of the database that is available to the attacker.

So, these are the 4 attack models which we consider in the literature. Now when I say that a particular encryption scheme is secure in a particular attack model. For example, if I say that an encryption scheme is secure in the CCA model, by that I mean that even if adversary is launching a CCA attack, that means adversary got access to the encryption oracle service, it got access to the decryption oracle service, still it should be difficult for the adversary to compute any function of the underlying plain text which has been freshly encrypted and communicated over the channel.

In the same way, if I say an encryption process is secure and the COA model or the ciphertext only attack model, I mean that just by looking into the ciphertext adversary should not be able to

compute any function of the underlying plain text. So, these are the 4 attack models which we will discuss in this course rigorously.


(Refer Slide Time: 41:43)

Keys and Kerckhoffs' Principle

- ❑ To maintain security, key should be definitely a secret
- ❑ What about Enc and Dec algorithm ?
 - More security by keeping them private too ?

❑ Kerckhoffs' Principle:

"A cryptosystem should be secure even if everything about the system, except the key, is a public knowledge"



19th century Dutch cryptographer

Now, the final thing which I want to discuss for this lecture is the Kerckhoffs' principle. So definitely to maintain the security, the key with which the encryption and the decryption process are operated should be a secret. Because if the keys leaked to the adversary, adversary can definitely find out what has been encrypted or decrypted.

Now, you might be wondering what about the encryption and decryption algorithm? Will I get more security by keeping the encryption and decryption algorithm private? Well, in cryptography we strictly say that we should not follow this, and this is coming from a well known principle which we call Kerckhoffs' principle.

Kerckhoffs' is a well known 19th century Dutch cryptographer. He laid down several guidelines, which should be followed by any good practical, secure cipher. One of the guidelines states that that a crypto system should be secure if everything about the system is public knowledge except the key. That means what this principle clearly states that the secrecy of the system should only rely on the secrecy of the key and not on the secrecy of the encryption and decryption process.

(Refer Slide Time: 42:59)

Importance of Kerckhoffs' Principle

- ☐ Maintaining the privacy of a key is relatively easier
 - Key size \approx 100 bits, Program size : 1000 times larger
 - Algorithm can be leaked, reverse engineered
- ☐ Easy to **replace** a key if the key is exposed
- ☐ Infeasible to assign a **secret pair of algorithm** for every pair of parties
- ☐ Published designs undergo **public scrutiny** and so likely to be more secure

Dangerous to Use a Proprietary Encryption Scheme



Now what is the importance of Kerckhoffs' principle? What I mean by that, what are the arguments in the favor of Kerckhoffs' principle. The first argument in the favor of Kerckhoffs' principle is the following: maintaining the privacy of a key is a relatively easier task compared to maintaining the privacy of a pair of algorithms. Because keys are roughly of size 100 bits, 200 bits, whereas programs are 1000 times larger than the size of the key. So that is why maintaining the privacy of a key is relatively an easier task than maintaining the privacy of a large algorithm.

Also the algorithm can **believe** or reverse engineer that even if you assume that encryption and decryption process which are operated by the sender and receiver are private, just by launching a KPA attack where adversary gets access to several messages and ciphertext pair. Adversary could reverse engineer the steps of the encryption and decryption process.

Another argument in the favor of Kerckhoffs' principle is that if your keys leaked, it is very easy to replace the key because your keys are of very short magnitude, whereas if a program is leaked, then it is also encryption processes leaked and it is very, very difficult to come up with a substitute. Because as you will see in this course, coming up with new encryption algorithm is really, really a very challenging task. Also, it is infeasible to imagine a scenario where a user selects a pair of secret encryption and decryption process for every party with which it wants to do the secure communication.

For example, if I want to do secure communication with 100 parties, I cannot come up with 100 secret algorithms or 100 secret encryption algorithms for each of these users, whereas it is feasible to imagine that I am going to use the same encryption and decryption process with each of the 100 users. But I am going to use 100 different keys for operating each of these instances of encryption and decryption algorithm.

Most importantly, if your encryption and decryption algorithms are publicly available, they go through public scrutiny. And if something has been proved to be secure, even after they are existing for last 30-40 years. Then we have a strong confidence that indeed, those published algorithms are more secure compared to an algorithm about which you do not know any details.

The summary here is it is extremely dangerous to use any kind of proprietary encryption process. So if someone says that, “hey, I am not going to leave the description of my encryption process, but I give you the guarantee that it gives you a very good amount of security”. You should not believe such encryption process, because you do not know what kind of loopholes might be present in such an algorithm. So it is always recommended to go or use algorithms which has available in public domain and has been scrutinized publicly.

So that brings us end of this lecture. To summarize, we discussed the various types of cryptographic primitives that we use in cryptography. We also discussed the syntax of symmetric encryption process. We focused on the importance of the key generation and encryption algorithms to be randomized, because this comes as an implication of Kerckhoffs’ principle, which says that the secrecy of the whole system should rely on the fact that only the key is hidden not the algorithms. We also discuss the various kinds of attack models, namely the COA attack model, the KPA attack model, the CPA attack, model and CCA attack model. I hope you enjoyed the lecture! Thank you.