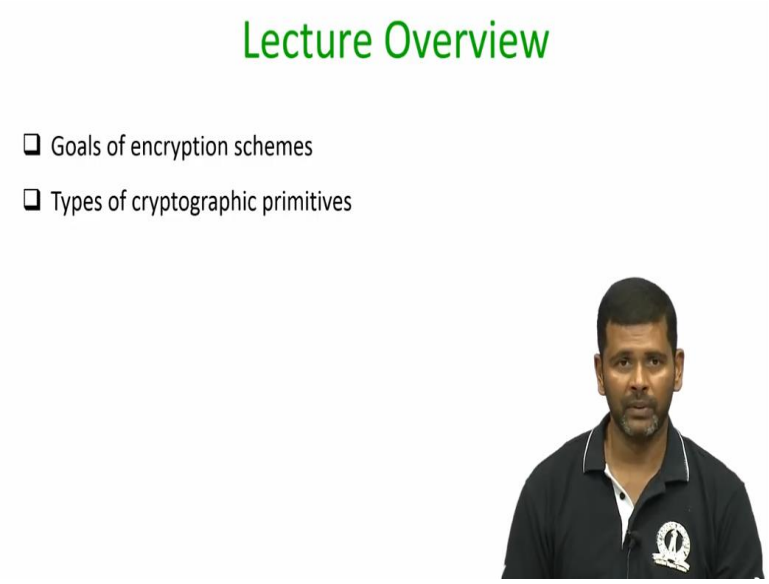
Secure Computation - Part I
Prof. Ashish Choudhury
Department of Computer Science
International Institute of Information Technology, Bangalore

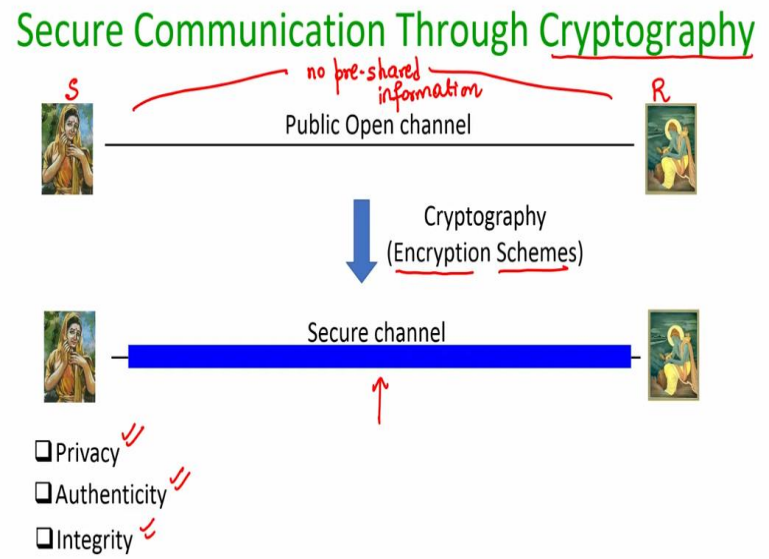
Module - 1
Lecture - 6
Recap of Basic Concepts from Cryptography

(Refer Slide Time: 00:34)



Hello everyone. Welcome to this lecture. Plan for this lecture is as follows: So, we will have a very brief discussion regarding cryptography; what are the goals of encryption schemes; types of cryptographic primitives used.

(Refer Slide Time: 00:48)



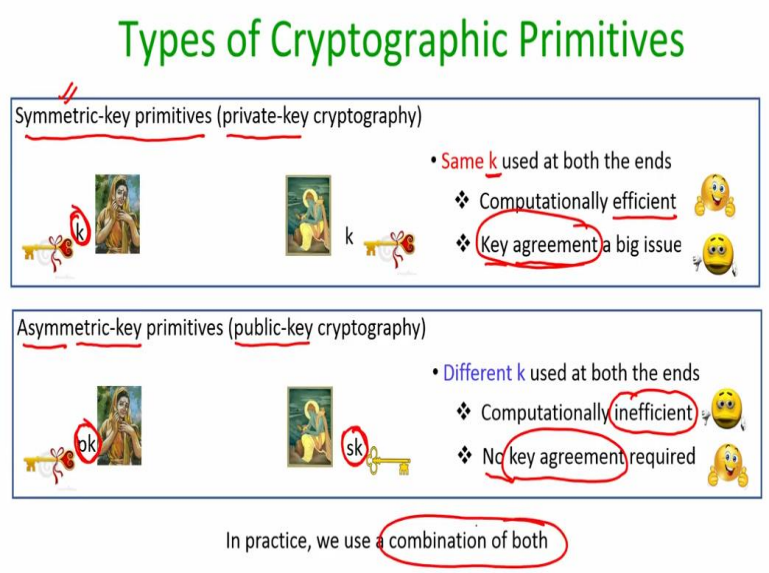
So, as I said in one of my earlier lectures, one of the main applications of cryptography is to perform secure communication. What do I mean by secure communication? So, the setting is as follows: You have a sender; you have a receiver; with no pre-shared information; and they are connected via public open channel. Now, cryptography provides you various encryption schemes, algorithms, using which sender and receiver can emulate a secure channel on top of this public open channel.

And what do I mean by emulating a secure channel? By emulating a secure channel, I mean that this channel will give you the following guarantees. It will give you the guarantee of privacy or confidentiality. Namely, whatever communication is happening over the channel, even though it is happening over the public channel, for a third party different from sender and receiver, it will be a completely meaningless communication; it cannot make out anything of that communication.

That is roughly the goal; or that is roughly the meaning of privacy. By emulating a secure channel, I also mean that the authenticity of the communication is guaranteed. That means, both S and R will have the guarantee that whatever messages or packets they are receiving over the public open channel has indeed originated from the right person or a right entity. And the third property of this secure channel emulation is that it should maintain the integrity of the communication.

That means, even though the communication is happening over the public channel, if there is a third party who has changed or who has tampered the communication, say by dropping few packets or by inserting few packets or by say reordering few packets, then it will be detected at the corresponding receiving end. So, that is what the main goal of cryptography is.

(Refer Slide Time: 03:10)



So, it turns out that we have different types of cryptographic primitives depending upon the requirements. The first category of cryptographic primitives, they are called as symmetric-key primitives often known as private-key cryptography. Why it is called symmetric-key cryptography? Because, here the encryption algorithm as well as the decryption algorithm, they operate with the same key k.

So, we imagine that there will be some key or common bit string. Do not get confused by this symbol of key; that is just for your understanding. So, you can imagine that there will be a common bit string denoted by k, that serves as a key for an algorithm which the sender is going to use. And that serves as the key even for some algorithm which the receiver is going to use.

Since same key is used at both the ends, that is why it is called as symmetric-key algorithms. And why it is called private-key cryptography? Because, the security of the entire communication depends upon the fact that this key should be known only to the sender and the receiver. It should not be known to any third party. Now, this type of cryptographic primitives has its own advantage and disadvantage.

The good part of these primitives is the following: They are computationally very efficient, very fast. That means, the encryption process, the decryption process, they will be super fast. But the downside is that they work under the assumption that a common bit string, namely a common key is kind of established or already known to the sender and the receiver. But that goes against the assumption that I made when I described a secure communication problem.

There I said that sender and receiver start with no pre-shared information. That means, we can use symmetric-key cryptography or symmetric-key encryption algorithms only if we have a mechanism to ensure that the sender and the receiver agree on some random common bit string k, known only to them, which is often called as the key agreement problem. The second class of cryptographic primitives, they are called as asymmetric-key primitives or public-key cryptography.

And unlike your symmetric-key primitives, here, the same key is not used both for encryption and decryption, or, the same key is not used by both the parties. So, different parties will be using different keys for different purposes. There will be one key which will be available with the sender, which will be used for encryption. And to perform the decryption, there will be a secret-key which is available only with the receiver.

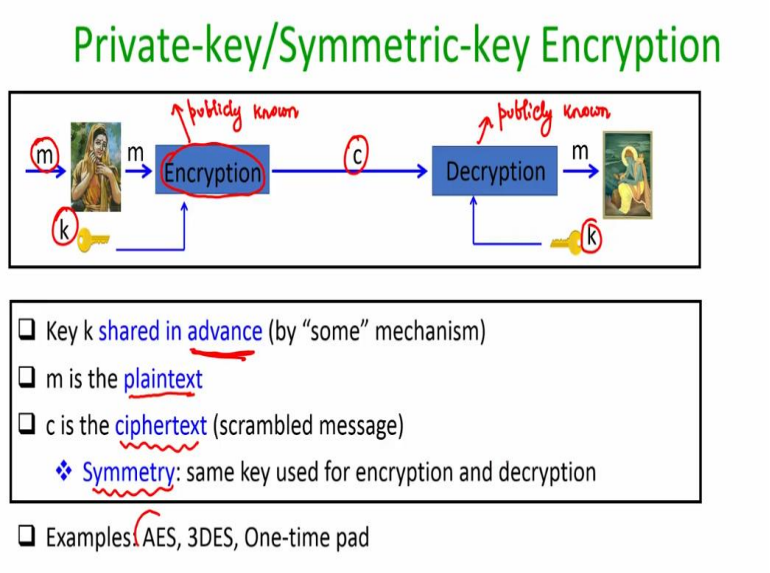
So, since different keys are used for encryption and decryption, that is why the terminology asymmetric-key. And why it is called public-key? It is not the case that both pk and sk will be available in the public domain. The key pk which is used for encrypting the messages will be available in the public domain, whereas the decryption key will be available only with the receiver.

Now, again, this category of cryptographic primitives has both advantages and disadvantages. The advantage is that, unlike your symmetric-key primitives, the key agreement is not required here. Just make the public-key available in the public domain. Anyone who wants to encrypt a message and send it to the receiver has to take that public-key, encrypt the message and communicate to the receiver.

This is unlike your symmetric-key primitives, where both for encryption as well as decryption you need the same key, and hence, the privacy of the key is very crucial there. However, the downside is that they are computationally inefficient compared to symmetric-key primitives, and they take enormous amount of time if we want to encrypt messages using public-key encryption schemes.

So, now, since we have 2 different types of primitives with different trade-offs, you might be wondering which one to use in practice. It turns out that, in practice, we use a combination of both of them.

(Refer Slide Time: 07:48)



So, now, let us go little bit deeper into the syntax of these 2 categories of encryption schemes. So, private-key encryption schemes operates as follows: So, this category of encryption schemes assumes that there is a some common random bit string of certain length denoted by k, which has been already shared in advance; that is why I highlighted it; by some mechanism. And how it will be shared in advance? We will see it later.

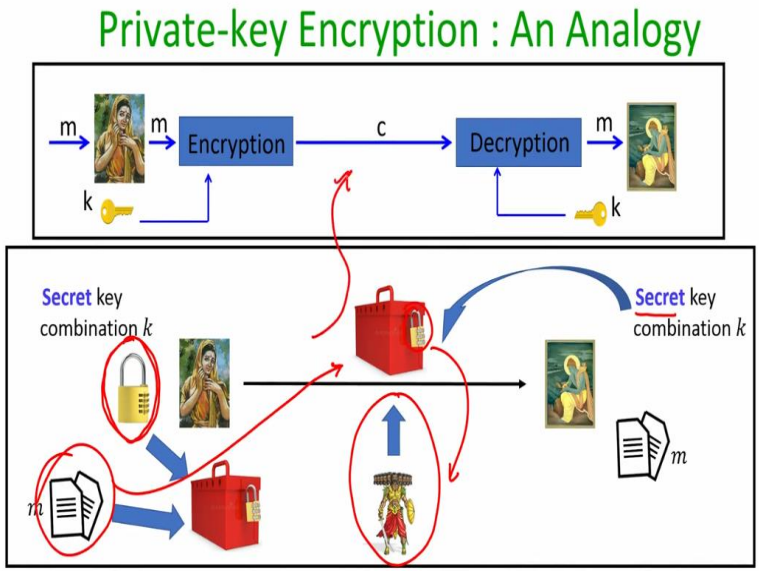
But this category of encryption schemes make the assumption that the key has been already shared or agreed upon between the sender and the receiver. Now, given this setup, if sender has some bit string or some message m, which it wants to securely communicate to the receiver, then it can do the following: So, this bit string m or the message of the sender is often called as the plaintext, because it is available in plain or in clear to the sender.

Sender will have some encryption algorithm available, whose details will be publicly known. And that will be, the details will be publicly known as part of the details of the symmetric-key encryption process. So, this encryption algorithm will take the sender's plaintext and the key, and it will produce another binary string denoted by c, which we call as a ciphertext or scrambled message.

And this scrambled message will be communicated over the public channel. Now, the receiver's goal will be to convert back this scrambled message into the sender's plaintext. And for doing that, receiver will have to use a decryption algorithm, which is again publicly known as part of the details of the encryption scheme. Now, the decryption algorithm will take the same key which sender has applied or used and take the scrambled message, and it will produce back the plaintext which sender has actually converted from plaintext to the scrambled text.

Now you can see that why it is called symmetric-key encryption scheme. Because, the same key is used for converting your plaintext into ciphertext, as well as for converting ciphertext back to the plaintext. Of course, I have not defined here the security properties, guarantees and so on. Those details are not required for this specific course. But you just need to understand how exactly a symmetric-key encryption scheme operates. So, some of the popular examples of symmetric-key encryption schemes are AES algorithm, Triple DES, One-time pad and so on.

(Refer Slide Time: 10:59)



So, now, to understand exactly how the private-key encryption works in practice, let me give you a very simple analogy. This is the way a symmetric-key encryption works. Sender and receiver will already have a common bit string. Whenever sender has a plaintext which she wants to communicate to the receiver, she applies the encryption algorithm; converts the plaintext into the scrambled text; scrambled text goes to the receiver; receiver applies the decryption algorithm and get converted back into the plaintext.

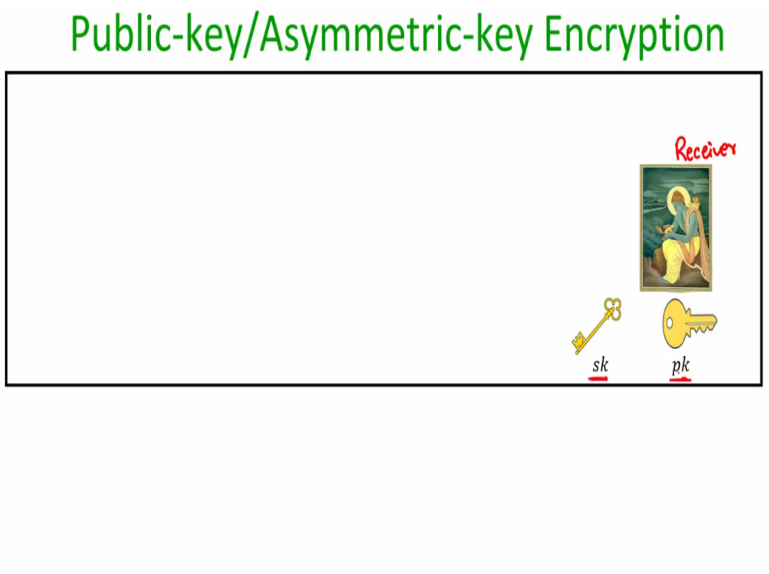
So, the way you can imagine the operation of the symmetric-key operation is as follows: You imagine that sender has a lock here, and there is a secret-key combination to open this lock. And that secret-key combination is somehow already shared between the sender and the receiver. Now, imagine sender has some document, physical document which she want to securely communicate to the receiver.

So, what she can do is, she can buy a box; she can keep the document inside the box; and she can lock the box using the key that she has with her; and now, she can courier this locked box to the receiver. That means, this locked box is now communicated over the public channel. So, you can imagine that getting this locked box delivered through a courier, is equivalent to communicating the ciphertext over the public channel.

Now, how the receiver is going to get back the document which sender has kept inside the box? So, since we are assuming that the receiver also have the same secret-key combination, it can apply that secret-key combination and open the lock. Once the lock is opened, he can open the box and find out the document. Now, if there is a third party, say the person who has actually delivered the courier; if there is a third party, and if the goal of the third party is to find out the contents of the box, then, to obtain the contents of the box, the third party has to open the box.

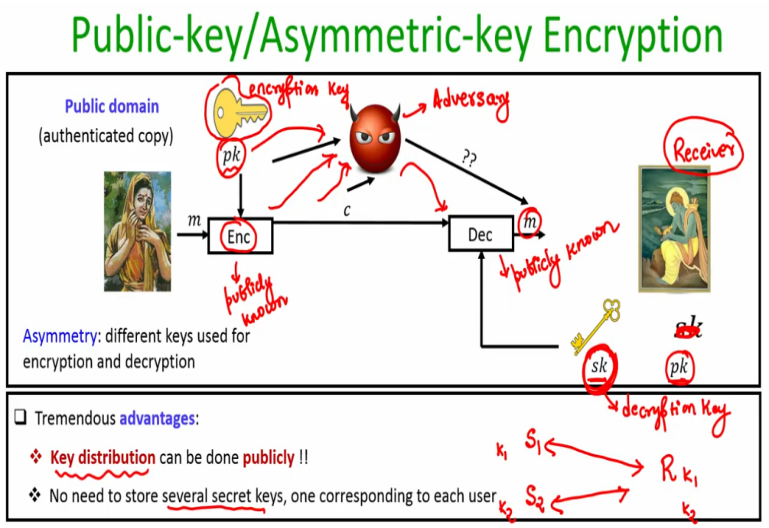
But for opening the box, he has to actually open the lock. But for opening the lock, this third party needs the secret-key combination. So, assuming that without a secret-key combination he cannot open the lock, he cannot see the contents that are kept inside the box. And that ensures the privacy of the documents. That is exactly the way your symmetric-key encryption works as discussed above.

(Refer Slide Time: 13:56)



Now, let us see the syntax of public-key encryption or asymmetric-key encryption. So, imagine this guy, Ram, is the receiver. As part of the public-key encryption scheme, Ram will have 2 keys. There will be an encryption key which will be denoted by pk and there will be a decryption key denoted by sk.

(Refer Slide Time: 14:18)



Now, this encryption key will be made available in the public domain. By public domain I mean, say for instance, it might be available in the web page of the receiver, homepage of the receiver or in some public domain. And it will be certified that, okay, this bit string pk; remember, pk is actually a bit string, but just for the sake of your understanding, I am using a key symbol here. So, it is a bit string whose purpose is to serve as an encryption key, and its value will be available in the public domain.

Everyone will know that, okay, this bit string is actually the encryption scheme of an entity called Ram. Now, if there is a person called Sita or a sender who wants to encrypt a message and send to the receiver, then in the public-key encryption scheme, this is the way she will be encrypting her message. There will be an encryption algorithm, which I call as Enc, whose details will be publicly known.

The encryption algorithm will take the sender's plaintext and the encryption key, and it will produce the scrambled text. Now, the scrambled text will be communicated over the public channel. To convert back the scrambled text or the ciphertext back to the plaintext, the receiver we will be applying a decryption algorithm, which again is publicly known; the algorithms are never kept secret.

The decryption key will take the scrambled text and the decryption key sk which is available only with Ram, and it will produce back the plaintext. Now, the security guarantee that these encryption schemes provide is the following: If there is an adversary or third party or an attacker who knows the public-key, who got access to the ciphertext, who even knows the description of the encryption algorithm, the that party cannot find out what exactly is the message.

That roughly is the loose definition of security of asymmetric-key encryption scheme. So, everything is known to the adversary except the decryption key. Only when the decryption key is available; even the details of the decryption algorithm is also publicly known, everything is known to the attacker. The only thing that is not known to the attacker is the value of this bit string sk.

So, now you can see that why it is called asymmetric-key encryption. Because, different keys are used for encryption and decryption. One key, namely the public key pk is serving the role of encryption key, and another key is serving the role of the decryption key. Now, this type of encryption schemes has tremendous advantages. The main advantage is that you no longer require to do a prior key agreement.

Namely, if the person, receiver or Ram wants to get messages communicated to him securely, he need not know the identity of the sender in advance. What he has to do is, he has to generate

a public-key or the encryption key and a decryption key and just make the public-key available in the public domain, that is all. Whoever is the sender and wants to communicate a message privately to the receiver, just have to look for the receiver's public-key from the public domain, and that is all.

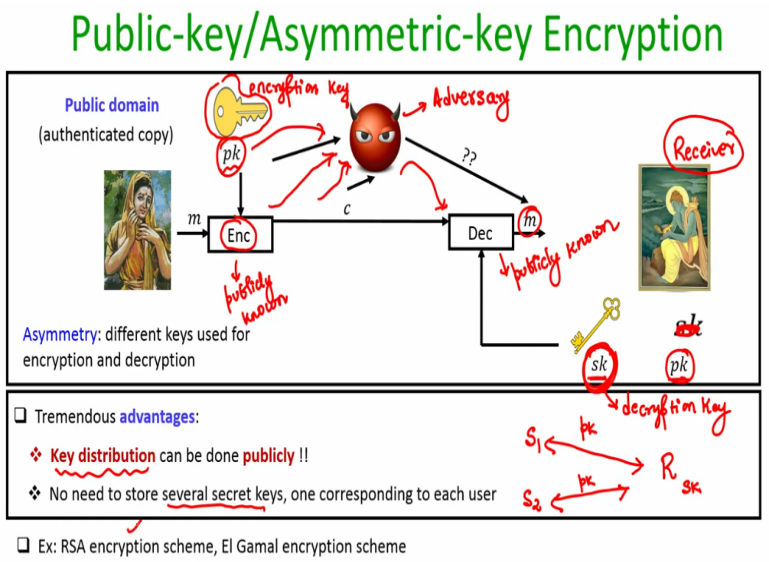
Receiver does not have to worry about who is going to be the sender, about whether she should provide him a secret-key and so that only he can do the secure communication to her; those things are gone now. And another advantage is that it is not needed to store several secret-keys for each possible sender. So, in the case of symmetric-key encryption scheme, there has to be, or specific key k shared between the sender and the receiver.

So, if I consider the viewpoint of the receiver R, and if there are 2 potential senders S_1 and SS_2 who would like to securely communicate to R, then in the symmetric-key world, we need different keys, different in the sense independent keys.

One key, R will be using while doing secure communication with S_1. And there should be another key, independent key k_2 which should be used to perform secure communication with S_2. The same key k_1 cannot be used for both performing secure communication with S_1 as well as S_2 in the symmetric-key world. Because, in that case, if S_1 is using the key k_1 and S_2 is also using the key k_1, then S_2 can simply go and eavesdrop the communication between S_1 and R, and find out what is happening between them.

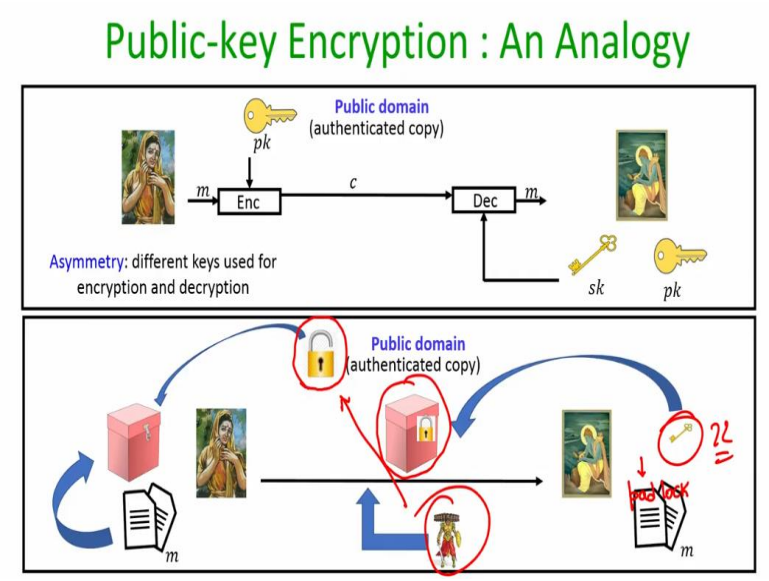
So, that is why, in the symmetric-key world, we need an independent key for every pair of sender and receiver. But if you consider the public-key world, that need not be the case. It does not matter how many senders are there.

(Refer Slide Time: 20:28)



It could be hundreds of senders. Receiver just need to maintain 1 public-key and 1 secret-key or the decryption key. All of them are going to use the same public-key and encrypt messages and communicate to the receiver. So, even if S_2 goes and eavesdrops the communication happening between S_1 and R, since S_2 does not have the decryption key sk, she cannot decrypt the contents and get back to plaintext. So, the examples of this class of encryption schemes are your popular RSA encryption scheme, El Gamal encryption scheme and so on.

(Refer Slide Time: 21:03)



Similar to the case of symmetric-key encryption, let me give you a real life analogy of how you can understand the operation of public-key encryption. Imagine this is your receiver, and receiver has generated 2 things. It has a padlock. And by padlock I mean, for locking it, you just need to press it. If you just press it, it gets locked, but to open it, you need the key.

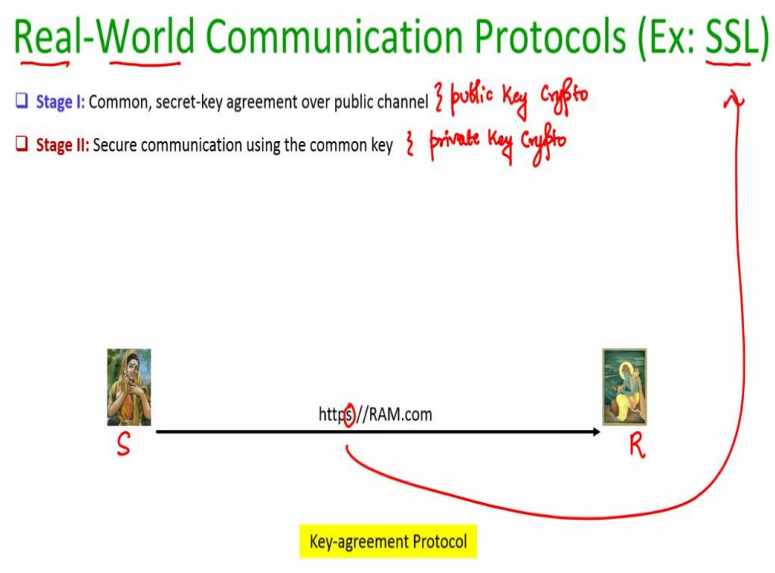
That means, for locking you do not require the key, but for opening you required the key. Now, what this receiver can do is the following: It can make this opened; it can make available the padlock in an open state in the public domain. Say, it keeps it in the post box; it keeps it in the post office and indicates in the post office that, okay, whoever wants to communicate any message privately to me can use this padlock; and the padlock is in the open state, remember that.

Now, imagine there is a sender who has some document. Now, sender wants to use this padlock and communicate the message securely to the receiver. So, what the sender can do is, she can go to the post office along with some box. Inside the box, she can keep the documents; take the padlock and press it, and the box get locked. And now, she can send the locked box using the postal service.

That is equivalent to ciphertext getting communicated over the public channel. And now, the receiver, since he possess the key for opening the padlock, he can apply the key and get back the document. So, that is an analogy of your public-key encryption. Again, if there is a third party, say the staff of the post office, who knows the details of this padlock; who is actually seeing the locked box going.

But since the post office staff does not know the value of this opening key or padlock key, they cannot open the padlock. And hence they cannot go and see the contents or the documents kept inside the box. That is a very loose analogy of the way public-key encryption works. So, now, you might be wondering what kind of encryption schemes we use in real-world communication protocols.

(Refer Slide Time: 23:35)



Say for example, the SSL protocol. So, most in real-world communication, whenever we want to do secure communication, we basically use an SSL protocol. There are various versions of SSL protocols. And this SSL protocol has 2 stages; and for various stages, we use different kinds of cryptographic primitives. So, in stage number 1, a common secret-key will be agreed between the sender and the receiver over the public channel.

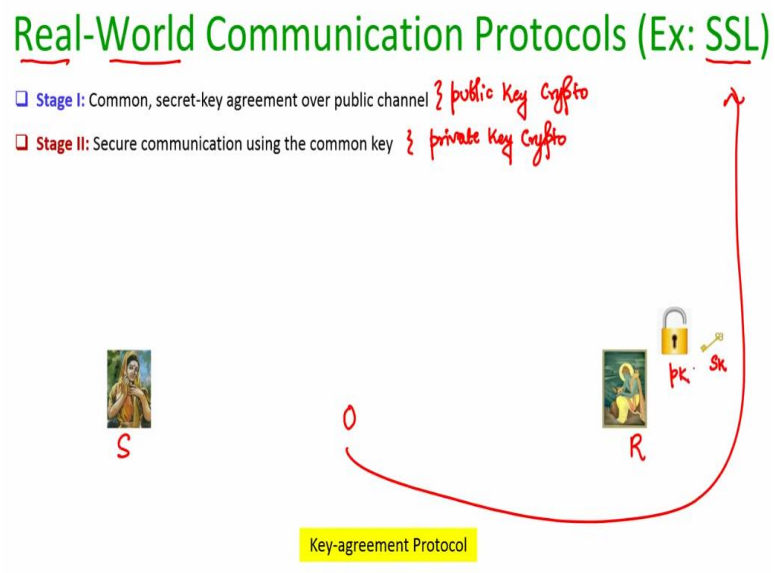
And this is done using public-key cryptography. Now, once the secret-key is agreed upon between the sender and the receiver, the actual secure communication starts in stage 2 of the SSL protocol using private-key crypto. So, that is why, you can see that we use best of the both. You might be wondering that why we do not do both stage 1 and stage 2 using public-key crypto. Because, key agreement would not be required.

We do not do that because, as I said earlier, the overhead associated with public-key crypto is tremendous. So, that is why we do not want to do the encryption of the entire session communication happening between the sender and the receiver using the public-key crypto. It is only for a small portion of the communication, namely, just for doing the key agreement, we use the public-key cryptography component.

Once the key is established between the sender and the receiver, we actually resort to symmetric-key cryptography for performing the actual secure communication. So, let us see. Let us try to understand how it happens. Imagine this is my sender; this is my receiver. And sender, in her browser, types https://ram.com. So, as soon as you type https, your SSL protocol

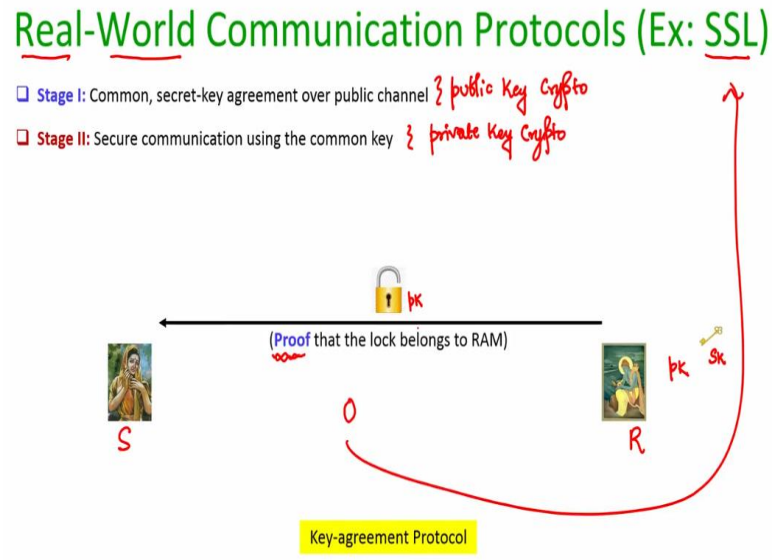
starts getting executed. Now, as soon as Ram receives this communication request from sender, namely an https request, what Ram does is the following:

(Refer Slide Time: 26:00)



It generates a public-key secret-key pair.

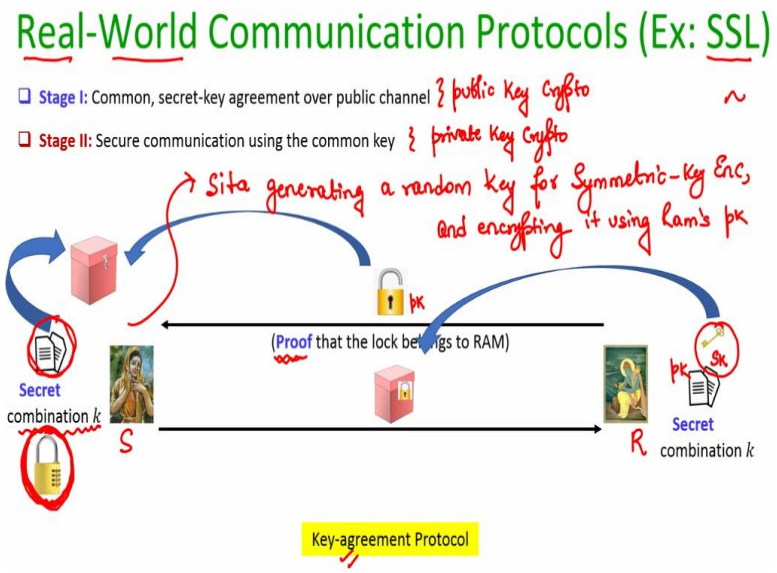
(Refer Slide Time: 26:08)



And it sends the public-key, namely the opened padlock to Sita, along with a proof that this padlock or this public-key pk indeed belongs to Ram; it does not belong to a random person. Now, what exactly is the proof used? Let us not go into the detail, but believe me, that there is a proof associated with this lock, some kind of certificate which certifies that this bit string or this padlock indeed belongs to the person Ram.

If the certificate verification fails, then basically, the sender will actually get a warning message in the browser that, okay, we cannot verify the authenticity of this public-key pk, you can proceed at your own risk. But if the certificate is verified, then pk will be communicated to the sender.

(Refer Slide Time: 27:11)



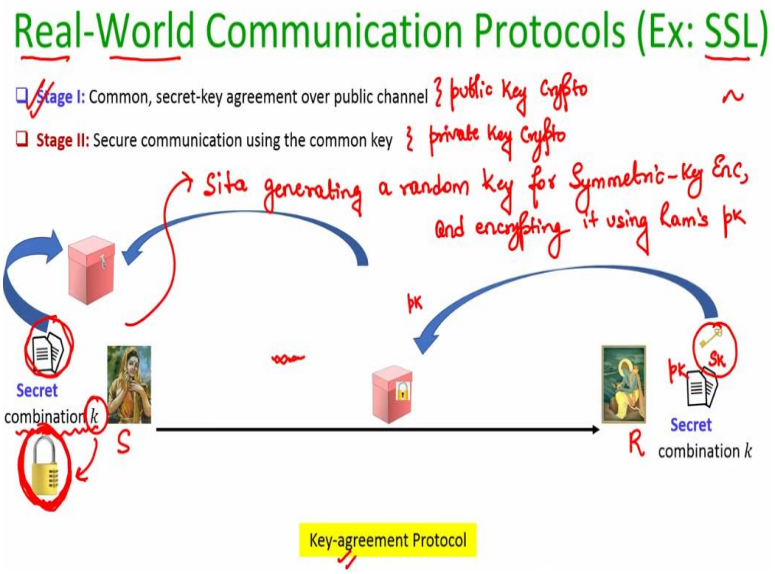
So, now, what sender has to do is the following: What she does is the following: She generates a secret combination for a lock. So, she generates a lock; she has a lock with herself and there is a key combination for opening it. That is her document. So, right now, she do not have any specific plaintext to communicate to the receiver, because right now, we are doing the key agreement.

So, what Sita is doing here is, she has now a padlock; not padlock; she has actually a lock, which requires a key, both for opening as well as locking. This is different from your padlock. For padlock, you need the key for opening, but for locking you do not require the key. But the lock which Sita has, it requires a key, both for locking as well as opening. So, she has the lock and the key corresponding to this lock.

What she does is the following: She keeps the key that is required for operating this lock inside a box, and lock it using the padlock of Ram, namely the public-key; and communicate it to the Ram. So, basically, you can imagine this entire process as if Sita is generating a random key for symmetric-key encryption and encrypting it using Ram's public-key. That means, the key for symmetric-key encryption is serving as the plaintext which Sita would like to encrypt using Ram's public-key.

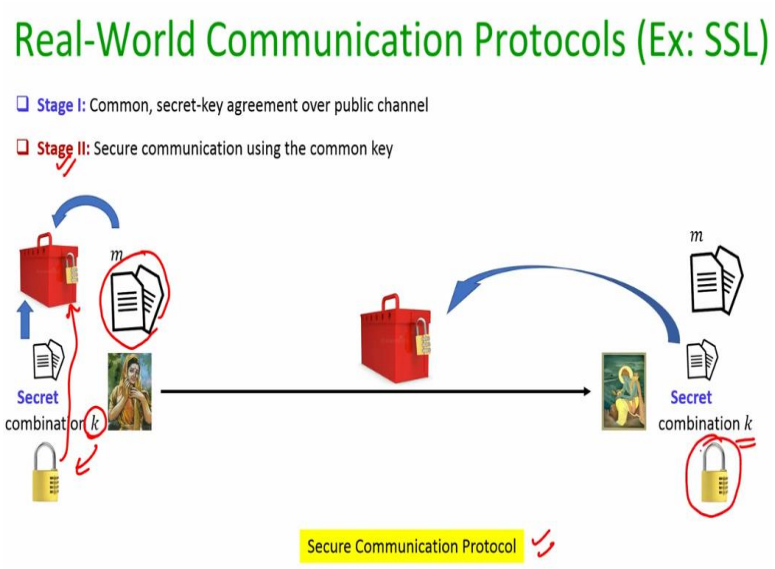
So, Sita right now do not have any specific plaintext. The only plaintext that Sita has is a randomly generated key for a random lock that only she has right now. And that key she is encrypting by treating it as a plaintext using Ram's public-key. Now, once Ram receives this ciphertext or the encrypted key, since Ram has the decryption key for opening the padlock, he can open it. And now, Ram will also have the same secret-key combination which is required to operate this key. And this will ensure that key agreement has happened now. You can see that.

(Refer Slide Time: 30:22)



So, in stage 1, the only secure communication which happens between the sender and the receiver is basically to agree upon this secret-key combination, which is required to operate this lock, nothing else. And that finishes your stage 1. Of course, there are other details also involved as part of your SSL protocol, but I am just giving you a very simplifying abstraction.

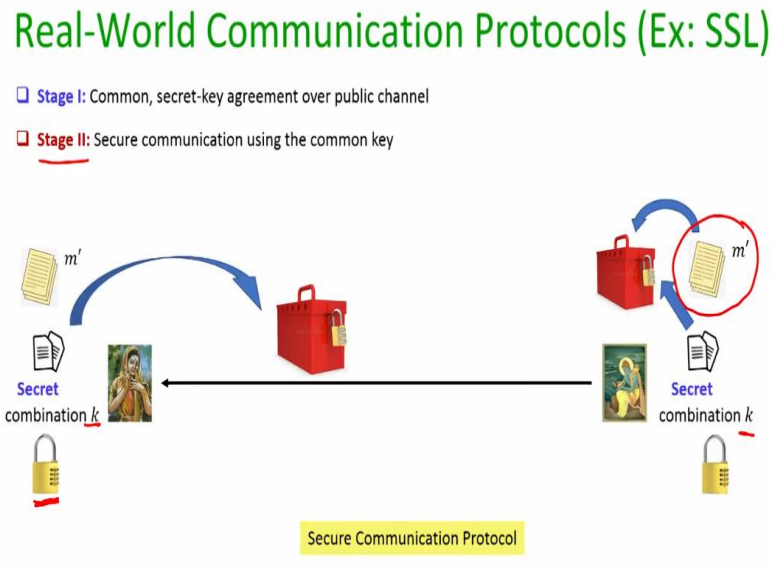
(Refer Slide Time: 30:47)



Now, in stage 2, the actual secure communication starts. And now you can see that how magically Sita and Ram have agreed upon this secret-key combination by actually performing the communication over the public channel itself. And now, since we have ensured that there is a secret-key combination established between the sender and the receiver or Sita and Ram, they can go to stage 2.

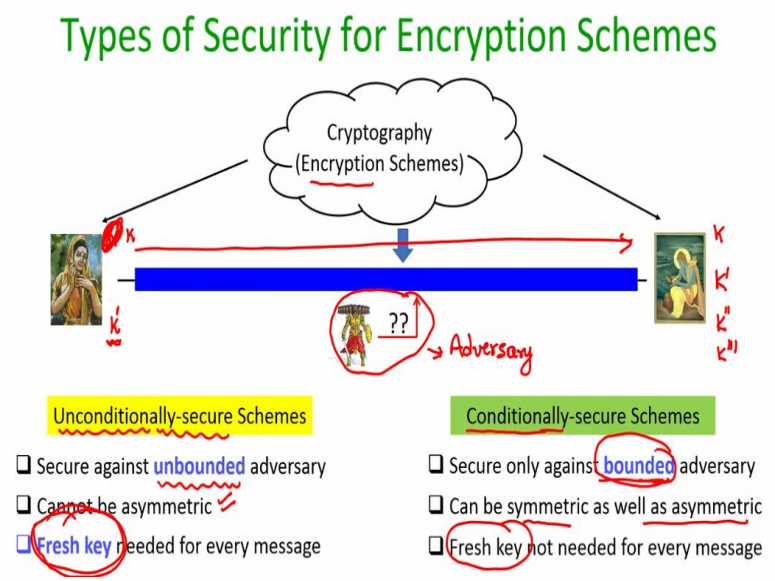
And whatever plaintext Sita now wants to securely communicate to the receiver, she can start encrypting it using the secret-key combination. Namely, she has to keep it in a document; keep the documents inside the box; lock it using this key and communicate it to the receiver. Since receiver also have the same secret-key combination, he can apply it and he can open the box to get back the document. And now, he will also have the same lock;

(Refer Slide Time: 31:48)



Which he can now use to communicate back any message m' if he has. So, now you can see that it is now the same lock which is used for all the communication during the stage 2. Now, different locks are not used. It is the same lock which is used every time Sita wants to communicate to Ram or Ram wants to communicate to Sita, because both of them have now the same secret-key combination magically established. So, that is how the real-world protocols like SSL operates.

(Refer Slide Time: 32:22)



Now, there are 2 types of encryption schemes when it comes to the level of security that they provide. So, loosely speaking, what is the level of security? Remember, the goal of encryption schemes is to emulate a secure authenticated channel on top of a public channel. Now, this entire communication should be secure against a third party who is often called as the adversary, who wants to cause harm to the sender or the receiver.

Now, this adversary need not be a person, basically it could be a virus or it could be a kind of algorithm which is trying to cause harm to the secure communication happening between the sender and the receiver. Now, this adversary could also have some computing resources requirements, because, finally, it also has to run some algorithm to find out what is the actual communication happening between sender and receiver; it has to analyse the ciphertext and so on.

So, depending upon the computing power of the adversary, we have 2 categories of encryption schemes. The first category of encryption schemes, they are called as unconditionally secure encryption schemes. They are called unconditionally secure because they give us security

guarantees, namely the privacy, authenticity and integrity guarantees, even if this adversary, this third person power is unbounded.

That means, even if the third party or the adversary deploys all the computing resources of the universe, still it cannot cause any harm to the secure communication happening between the sender and the receiver. Whereas, the second category of encryption schemes which are called as conditionally secure schemes, it gives us the security guarantees, namely the privacy guarantees, authenticity guarantees and integrity guarantees, only if the attackers or the adversary's computing resources are bounded.

So, of course, you might be saying that, why I will be going for conditionally secure schemes? Because, if I have the option of deploying unconditionally secure schemes, I will deploy them, because that provides me more security. It gives me security guarantees even against the more powerful adversary. But there are some trade-offs associated. So, these unconditionally secure schemes, it always have symmetric-key variant.

That means, by nature, they have to be symmetric-key encryption schemes. That means, the same key will be used both for encryption and decryption; you cannot have a public-key variant. And that is why the key agreement is a huge challenge while deploying this unconditionally secure schemes. And worse, what you require here is the following: You need that, for each instance of encryption, a fresh key needs to be used.

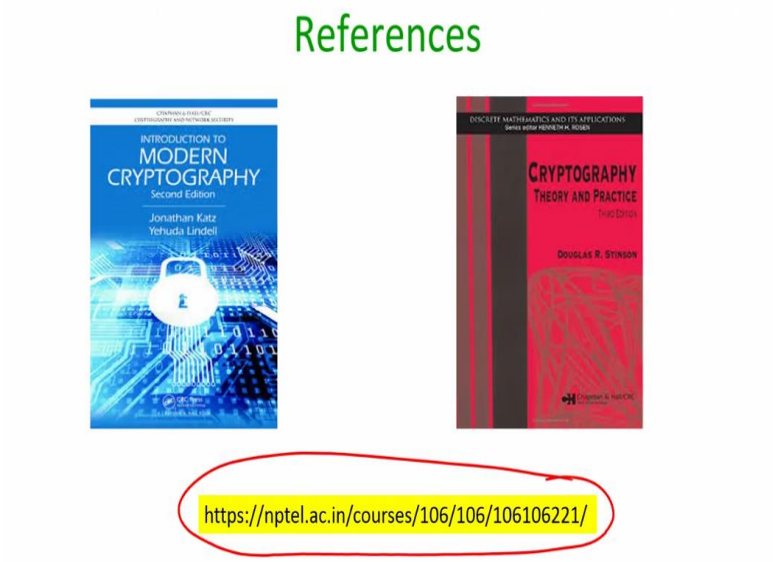
That means, every time sender wants to communicate a message to the receiver, she should have a fresh key. As soon as she performs the encryption and communicate the message to the receiver, she should throw off this key. And again, next time she wants to do a secure communication, she has to generate a fresh key. By fresh key, I do not mean a key which is content wise different from the previous key.

By fresh key, I mean a key which is generated independently of the previous key. It could be the case that the fresh key contents is same as the previous key. That is fine. But it has to be generated independently by some algorithm. That means, the probability that k' is same as k, could be non-zero. But from the viewpoint of the third party or the attacker, k and k' should be different.

It could be that they are not different, but they are independent. And we need that all the keys which are used by sender to communicate to the receiver, they should be magically communicated beforehand to the receiver, because, then only the receiver can do the decryption. So, that is why, even though you get a very strong security guarantees from unconditionally secure schemes, you need a kind of a very stringent requirement, namely, fresh keys for every message.

This is not the case for conditionally secure schemes. Because, for conditionally secure schemes, you can use a single key for encrypting the entire session messages. And conditionally secure schemes are available both in the symmetric as well as asymmetric-key variant. So, all your public-key encryption schemes like RSA, El Gamal, they are secure only when your adversary is computationally bounded. As soon as your adversary becomes computationally unbounded, their security will be broken.

(Refer Slide Time: 37:36)



So, that roughly concludes the discussion regarding cryptography that we need fit for this course. If you are interested to know more about the specific details, security properties of encryption, decryption and other cryptographic primitives, you are referred to this NPTEL course. And these are some wonderful textbooks to learn more about cryptography.