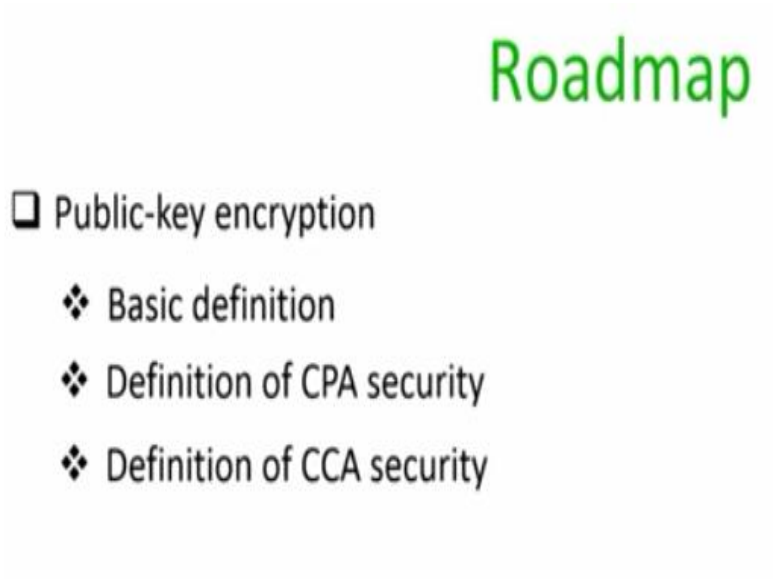
Foundations of Cryptography

Prof. Ashish Choudhury
Department of Computer Science
International Institute of Information Technology - Bangalore

Lecture 43
Public Key Encryption

(Refer Slide Time: 00:32)

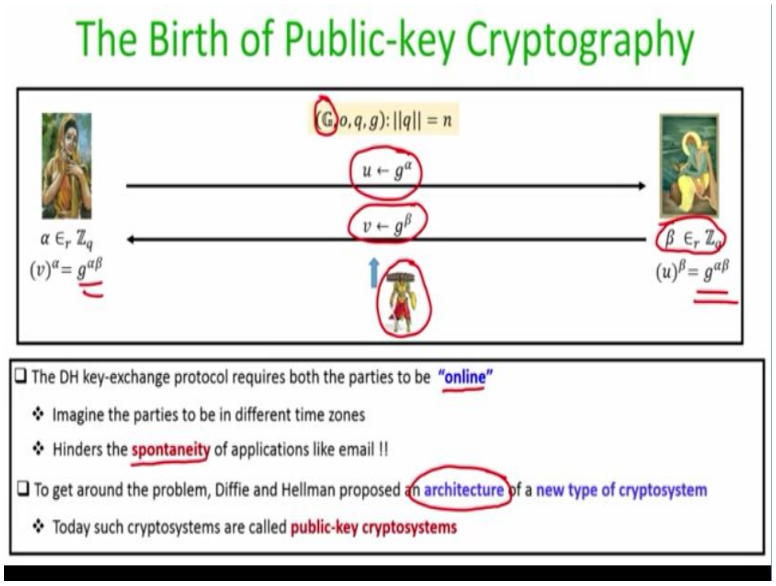


Roadmap

- ❑ Public-key encryption
  - ❖ Basic definition
  - ❖ Definition of CPA security
  - ❖ Definition of CCA security

Hello everyone. Welcome to this lecture. Just to recap in the last lecture, we had seen some cryptographic applications of discrete log assumptions and till now, we have seen the definition of cyclic groups and various assumptions in the cyclic groups and so on. So the plan for this lecture is as follows: We will start our discussion on public key encryption, where we will introduce the syntax of public key encryption schemes and we will see the definition of CPA security and CCA security in context of public key encryption scheme.

(Refer Slide Time: 01:00)


The Birth of Public-key Cryptography

$(\mathbb{G}, o, q, g): ||q|| = n$
$u \leftarrow g^{\alpha}$
$v \leftarrow g^{\beta}$
$\alpha \in_r \mathbb{Z}_q$
$(v)^{\alpha} = g^{\alpha\beta}$
$\beta \in_r \mathbb{Z}_q$
$(u)^{\beta} = g^{\alpha\beta}$

- The DH key-exchange protocol requires both the parties to be **"online"**
  - Imagine the parties to be in different time zones
  - Hinders the **spontaneity** of applications like email !!
- To get around the problem, Diffie and Hellman proposed an **architecture** of a **new type of cryptosystem**
  - Today such cryptosystems are called **public-key cryptosystems**

So let us see how exactly the public-key cryptography came into existence. So let me recall the Diffie–Hellman key exchange protocol and for simplicity, I am assuming that the underlying cyclic group g is a multiplicative group. So the Diffie–Hellman key exchange protocol is a single-round protocol where we have a sender and a receiver and sender picks her contribution towards the final key, namely random index α in the range 0 to q – 1.

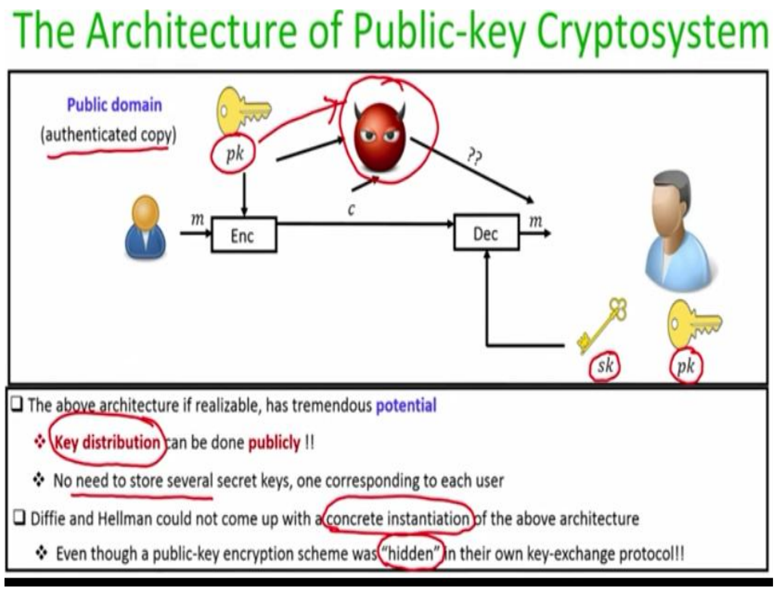
And sends her contribution g^{α} and parallelly the receiver picks his contribution namely a secret β in the range 0 to q – 1 and sends his contribution, namely g^{β} and that the final key is $g^{\alpha\beta}$, which is agreed upon between the sender and the receiver and if there is an eavesdropper, who has eavesdropped the communication between the sender and a receiver, then, if the CDH assumption holds, then we have proved that this key exchange protocols gives you a weak privacy, namely adversary does not learn anything about the resultant key, but it is a weaker notion of secrecy, because if you want a stronger notion of secrecy, namely the indistinguishability based definition of the key exchange protocol, then we need to have the DDH problem to be hard in the underlying group.

So disadvantage of this key exchange protocol, this Diffie–Hellman key exchange protocol is that it requires both the parties to be available online, in the sense that imagine the sender and a receiver to be present in different time zones, say for instance when sender is sending her contribution or her message for a Diffie–Hellman key exchange protocol, say it is late night at her end and say at that time, it is early morning at a receiving end.

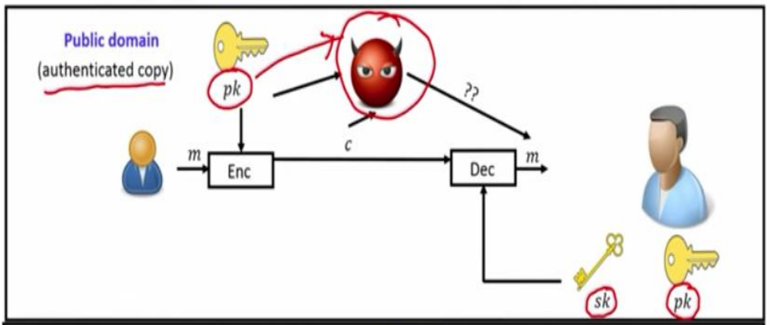
So until and unless sender in this Diffie–Hellman key exchange protocol receives the receiver's contribution, namely g^{β}, the key is not agreed upon. So in that sense, this Diffie–Hellman key exchange protocol, it requires both the entities to be available online and as a result, it hinders the spontaneity of applications like email. What I mean by that is, imagine, if sender in this case Sita wants to run the Diffie–Hellman key exchange protocol, agree upon a key and then use that key to encrypt an email, which she wants to send to Ram, then until and unless Ram comes online, the key exchange would not happen and until and unless key exchange does not happen, Sita cannot encrypt her email and communicate it to Ram. So in that sense, this Diffie–Hellman key exchange protocol hinders the spontaneity of email-like applications.

So to get around this problem, Diffie and Hellman proposed an architecture for a new kind of encryption process or new kind of cryptosystem. Today, such crptosystems are popularly known as public-key cryptosystem.

(Refer Slide Time: 03:54)



The Architecture of Public-key Cryptosystem



- ❑ The above architecture if realizable, has tremendous potential
  - ❖ Key distribution can be done publicly !!
  - ❖ No need to store several secret keys, one corresponding to each user
- ❑ Diffie and Hellman could not come up with a concrete instantiation of the above architecture
  - ❖ Even though a public-key encryption scheme was "hidden" in their own key-exchange protocol!!

So let us see the exact architecture of public-key cryptosystem, which Diffie and Hellman proposed. So in the so-called public-key cryptosystem, which Diffie and Hellman proposed or the architecture of the public-key cryptosystem, which Diffie and Hellman proposed, we have two entities or say in this case, we have a receiver, for example, and receiver will do a key generation and instead of generating a single key, which was the case for the symmetric key cryptographic primitives, the receiver is going to generate two keys.

One of the keys will be denoted as sk or secret key, which will be available only with the receiver and other key will be denoted as pk, which will be available in the public domain. So once the receiver generates the keys sk and pk, it is going to make the public key available in the public domain and in the public domain, an authenticated copy of this key will be available.

What do I mean by the authenticated copy is that, wherever this public key pk is available, it is guaranteed that this public key pk indeed belongs or indeed is generated by this designated receiver. How exactly that is ensured, we will see that later on, but for the moment, assume that we have mechanism using which the receiver can produce (a public key, a secret key) pair and it can put its public key in an authenticated fashion in some public domain.

Now if there is any sender in this world, who wants to encrypt some message as per this public key cryptosystem to the so-called receiver, then what the sender has to do is, it has to look for the public key of the designated receiver, take the plain text which sender wants to send and encrypt and communicate in an encrypted fashion and it will use the encryption algorithm of the public key cryptosystem.

So now this encryption algorithm will be different from the encryption algorithms of a symmetric key encryption process. This will be an encryption scheme of a public key cryptosystem. So using the public key of the designated receiver and encryption algorithm, sender takes the plain text, feeds a plain text and computes the cipher text, which will be now communicated to the designated receiver.

And once receiver receives the cipher text, it will use that corresponding decryption algorithm of the public key cryptosystem, but now to decrypt the cipher text; it will be using the secret key, which is available only with the receiver. So this is unlike the symmetric key cryptosystem, where the same key was used for encryption as well as for decryption. So here, different keys are used for encryption and decryption.

For encryption, pk is used, which is available in the public domain, but to decrypt a cipher text, the secret key sk is going to be used, which will be available only with the designated receiver and a security property that we require from this whole architecture is that, if there is an eavesdropper, for the moment, we are assuming an eavesdropper, later on we can go for a security against a more powerful malicious adversary.

But for simplicity, assume now we are dealing with an eavesdropper. So the security property that we require here is that, if there is a computationally bounded eavesdropper, who knows the description of the public key, because anyhow the public key is going to be available in the public domain and even if the adversary knows the description of the encryption algorithm and decryption algorithm and it has intercepted the cipher text c, it should not learn anything about the underlying plain text, which is encrypted in the cipher text c. So that is a security property that we require informally from this so-called public key cryptosystem. So what Diffie and Hellman proposed is, they just proposed an architecture of this kind of cryptosystem and what they thought is that if at all this kind of architecture is realizable, that means if we can come up with a candidate encryption process, candidate decryption process, and a candidate key generation process, which satisfies the above requirement, then such an instantiation will have tremendous potential. In the sense, that key distribution problem is no longer going to be a challenging task. The key distribution can now then be done publicly, in the sense that if I am a receiver, I do not care who is going to the sender, with whom I want to do a secure communication.

I can just generate a pair of secret key and public key and I can safely just put my public key in some public domain, say in my home page or in some public directory and whosoever wants to encrypt some message to me, he or she can look for my public key available in the public domain, and using the encryption algorithm, he or she can encrypt a message and send it to me.

So I do not have to run any sophisticated key agreement protocol to do the key agreement with any designated sender, with whom I want to do secure communication. That means, the issue of both the sender and the receiver to be available online, which was the case for Diffie-Hellman key exchange protocol is no longer happening here. The receiver need not be available online. Once it generates the key, it can make the public key available and it can go offline.

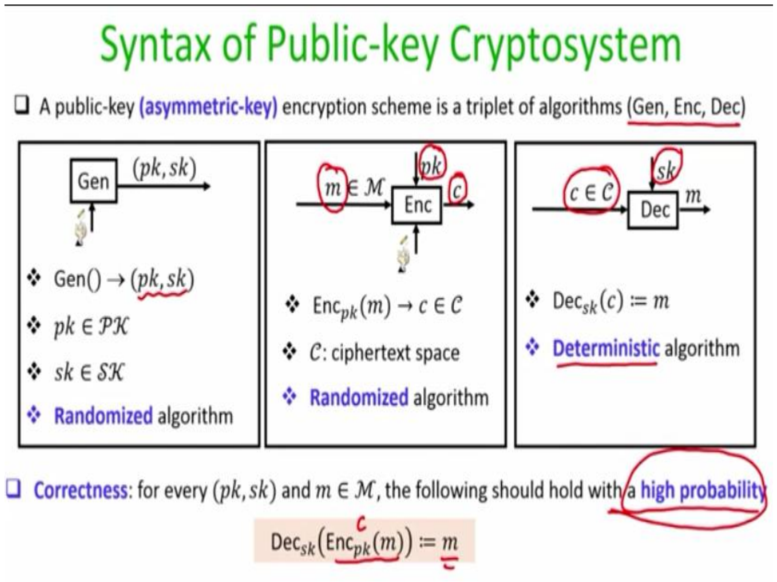
Whosoever wants to encrypt a message for the receiver, it can pick the public key and compute to cipher text and send it to the receiver. Whenever the receiver comes online, he or she can decrypt a cipher text using the secret key available with the receiver. So that is the huge advantage, if we can realize this kind of architecture. And the second advantage is that we do not need to store several secret keys, one corresponding to each user.

So again, if I go to the symmetric key world, if I am a receiver and if I want to do secure communication with 100 users, then for each of the users, I have to store a designated or dedicated secret key. The same secret key I cannot use to do secure communication with every possible sender. Whereas in the case of public key cryptosystem, I do not need to store separate keys for each of the user with whom I want to do secure communication.

I just need to generate one pair of keys, (a single public key and a single secret key), public key I can comfortably put in the public domain and that suffice for the entire communication. So even though Diffie and Hellman proposed this architecture of wonderful public key cryptosystem, they could not come up with a concrete instantiation of the above architecture and they failed to realize that an instantiation of public key cryptosystem was actually hidden in their own key exchange protocol.

We will see later on that how exactly by just making some minor modifications to the Diffie-Hellman key exchange protocol, we can get an instantiation of the public key cryptosystem. But Diffie and Hellman failed to spot the modification. They failed to realize that indeed a public key cryptosystem is hidden in the key exchange protocol and that is why Diffie and Hellman are not credited with the first instantiation of the public key cryptosystem. They just came up with the architecture.

(Refer Slide Time: 10:58)



Syntax of Public-key Cryptosystem

- A public-key **(asymmetric-key)** encryption scheme is a triplet of algorithms (Gen, Enc, Dec)

- $\text{Gen}() \rightarrow (pk, sk)$
- $pk \in \mathcal{PK}$
- $sk \in \mathcal{SK}$
- **Randomized** algorithm

- $\text{Enc}_{pk}(m) \rightarrow c \in \mathcal{C}$
- $\mathcal{C}$: ciphertext space
- **Randomized** algorithm

- $\text{Dec}_{sk}(c) := m$
- **Deterministic** algorithm

- **Correctness:** for every (pk, sk) and $m \in \mathcal{M}$, the following should hold with a **high probability**

$$\text{Dec}_{sk}(\underbrace{\text{Enc}_{pk}(m)}_{}) := m$$

So now let us go to the formal details and syntax of the public key cryptosystem. So a public key cryptosystem which is also known as asymmetric key encryption scheme, is a collection of three algorithms, a key generation algorithm, an encryption algorithm, and a decryption algorithm. The key generation algorithm does not take any external input, but it is a randomized algorithm and it is going to produce two keys, a key pk, which is going to be available in the public domain and a key sk, which will be available only with the receiver.

And this algorithm has to be a randomized algorithm, because if the algorithm is deterministic and the steps of the algorithm are publicly known, then anyone will know what exactly is pk and sk, which is going to be output by the algorithm. The encryption algorithm is a randomized algorithm, which takes an external input plain text, from the plain text space and a public key for encryption and it will produce a cipher text c from the cipher text space.

And for any reasonable security guarantee, we need this algorithm to be a randomized algorithm. On the other hand, the decryption algorithm has two external inputs. So it takes a cipher text, which you want to decrypt and a secret key sk and by running the decryption algorithm, we obtain or recover back the plain text. So now you can see why this cryptosystem is called asymmetric key, because we are using different keys.

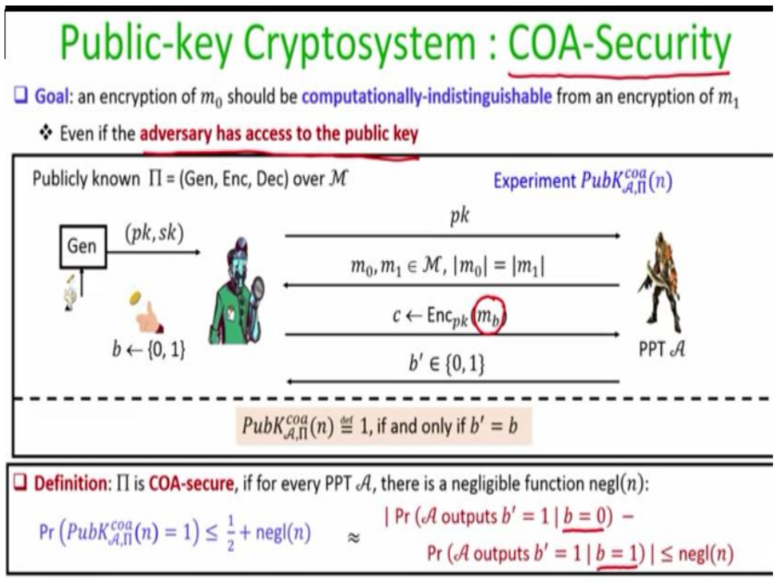
The key pk is used for the encryption process, whereas the key sk is used for the decryption process and for decryption to be unambiguous or error free, we require the decryption algorithm to be a deterministic process. So we now need two properties from any public key cryptosystem. The correctness property states that for every key pair, which we obtain by running the key generation algorithm and for every plain text from the plain text space, the following should hold with high probability.

Namely, if I encrypt some message m using the public key pk and if the resultant cipher text is decrypted using the corresponding secret key sk, then I should obtain back the original plain text with high probability. So now you see that, there might be a small error probability that the correctness guarantee fails in the context of public key cryptosystem and this is slightly different from the correctness requirement that we had for the symmetric key cryptosystem, where the same key was used for encryption and decryption.

And there we required the correctness to be error free. That means, there should be absolutely no error in recovering the plain text, if we encrypt and decrypt under the same key, but when we come to the public key cryptosystem, depending upon the exact steps of the key generation algorithm, there might be a case where the public key and secret key are not consistent with each other. We will see what exactly consistency mean.

In that case, if the public key and secret key are not consistent with each other, there might be an error in the decryption process. That is why, we allow for a very small error probability to be associated with the correctness guarantee in the context of public key cryptosystem, because that stems from the fact that your key generation algorithm might have some small error in generating the parameters correctly.

(Refer Slide Time: 14:27)


Gen
(pk, sk)
$b \leftarrow \{0,1\}$
pk
$m_0, m_1 \in \mathcal{M}$, $|m_0| = |m_1|$
$c \leftarrow \text{Enc}_{pk}(m_b)$
$b' \in \{0,1\}$
PPT $\mathcal{A}$

Public-key Cryptosystem : COA-Security

❑ **Goal**: an encryption of m_0 should be **computationally-indistinguishable** from an encryption of m_1

❖ Even if the **adversary has access to the public key**

Publicly known Π = (Gen, Enc, Dec) over $\mathcal{M}$

Experiment $PubK^{coa}_{\mathcal{A},\Pi}(n)$

$PubK^{coa}_{\mathcal{A},\Pi}(n) \overset{\text{def}}{=} 1$, if and only if $b' = b$

❑ **Definition**: Π is **COA-secure**, if for every PPT $\mathcal{A}$, there is a negligible function negl(n):

$$\Pr\left(PubK^{coa}_{\mathcal{A},\Pi}(n) = 1\right) \leq \frac{1}{2} + \text{negl}(n) \quad \approx \quad |\Pr(\mathcal{A} \text{ outputs } b' = 1 \mid \underline{b = 0}) - \Pr(\mathcal{A} \text{ outputs } b' = 1 \mid \underline{b = 1})| \leq \text{negl}(n)$$

So now let us see how we define the security in the context of public key cryptosystem and we will start with the very basic security guarantee, namely that of cipher text only attack security, where the goal is that my encryption process should ensure that an encryption of message m_0 should be computationally indistinguishable from an encryption of the message m_1, which was the case when we defined the COA security in the context of symmetric key encryption process.

But now let us see how exactly the COA security is modeled in the case of public key cryptosystem. So remember that since we are in the public key world, even the adversary will have access to the public key, because the public key is going to be available in the public domain. This is unlike the symmetric key world, where we have just a single key used by both the sender and a receiver and that key is not given to the adversary.

So the COA indistinguishability experiment $PubK^{coa}_{\mathcal{A},\Pi}\ (n)$ in the public key world is as follows. We have a polytime adversary and a challenger. The challenger runs the key generation algorithm, obtains a key pair, and the public key is given to the adversary and this models the fact that in reality whenever the receiver does the key setup, the public key will be available in the public domain, including the potential adversary.

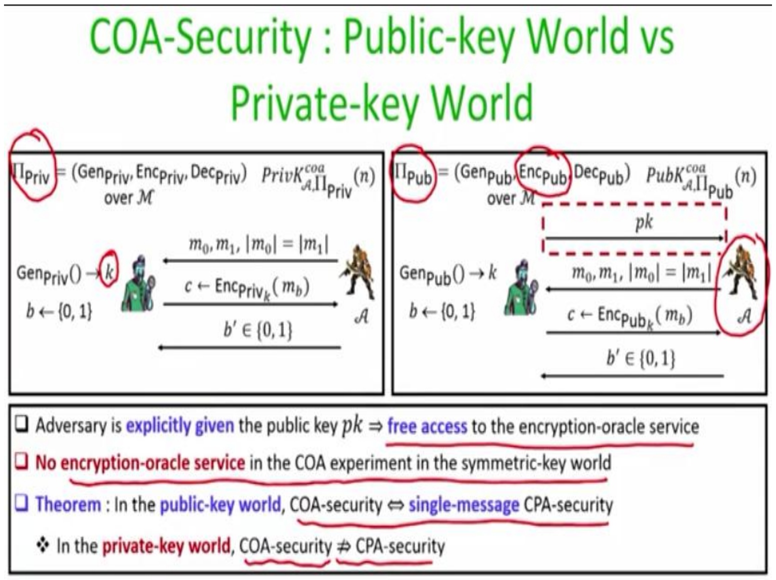
Now the adversary generates a pair of challenge plain text m_0, m_1 with the only restriction, it could be any pair of plain text with the only restriction being that their length should be same and challenger generates a challenge by picking one of these two messages with equal probability for encryption, and once it decides what message to encrypt m_b, it encrypts a message and the resultant cipher text is given to the adversary.

And a challenge for the adversary is to identify whether is seeing c, whether the cipher text c, which he is seeing is an encryption of m_0 or whether it is an encryption of m_1. So it outputs the index b**'** and the definition here is, we say that the COA experiment output 1 namely adversary has won the experiment, if and only if b**'** is equal to b.

Now the security definition is we say a public encryption scheme Π is COA secure, if for every polytime adversary participating in this experiment, there should be a negligible function, such that the probability of adversary winning the experiment is upper bounded by half plus negligible or equivalently the distinguishing advantage of the adversary is upper bounded by some negligible function.

That means, it does not matter whether the challenge cipher text c is an encryption of m_0 or whether the cipher text c is an encryption of m_1 with almost the same probability the response from the adversary should be the same.

(Refer Slide Time: 17:23)



COA-Security : Public-key World vs Private-key World

Π_{Priv} = (Gen$_{Priv}$, Enc$_{Priv}$, Dec$_{Priv}$) over $\mathcal{M}$
$PrivK^{coa}_{\mathcal{A},\Pi_{Priv}}(n)$
Gen$_{Priv}$() → k
b ← {0, 1}
$m_0, m_1, |m_0| = |m_1|$
c ← Enc$_{Priv_k}$(m_b)
b′ ∈ {0, 1}
$\mathcal{A}$
Π_{Pub} = (Gen$_{Pub}$, Enc$_{Pub}$, Dec$_{Pub}$) over $\mathcal{M}$
$PubK^{coa}_{\mathcal{A},\Pi_{Pub}}(n)$
pk
Gen$_{Pub}$() → k
b ← {0, 1}
$m_0, m_1, |m_0| = |m_1|$
c ← Enc$_{Pub_k}$(m_b)
b′ ∈ {0, 1}
$\mathcal{A}$

- ❑ Adversary is explicitly given the public key pk ⇒ free access to the encryption-oracle service
- ❑ No encryption-oracle service in the COA experiment in the symmetric-key world
- ❑ Theorem : In the public-key world, COA-security ⇔ single-message CPA-security
  - ❖ In the private-key world, COA-security ⇏ CPA-security

So now, we have the definition of COA security in the public key world, so what I will do now is, I will compare the COA security indistinguishability game, that we had seen in the context of symmetric key cryptosystem and we will compare how exactly it differs from the COA security game in the context of public key cryptosystem. So this is the COA security indistinguishability game against a symmetric encryption process, which I denote as Π_{Priv}.

So the prefix Priv denotes basically that it is a symmetric key encryption process or a private key encryption process, where the same key is going to be used for both encryption and decryption. On your right hand side, I have a public key encryption system, which I denote as Π_{Pub} and the corresponding indistinguishability game is here. So if you see the difference between the two experiments is that, in the COA game that we have for the public key crypto system, the public key pk is explicitly given to the adversary to model the fact that for any public key crypto system, the adversary will be knowing the public key, because it is available in the public domain and this is unlike the symmetric key encryption process, because in the symmetric key encryption process, the same key k is going to be used both for encryption and decryption and that is why the key k cannot be given to the adversary.

Because if the key k is given to the adversary, there is no way we can prevent the adversary from identifying whether c is an encryption of m_0 or whether c is an encryption of m_1. So that is a syntactic difference in the COA game between the two worlds, but if you see that in the public key world since the adversary is explicitly given the public key, that means it gets free access to the encryption oracle service.

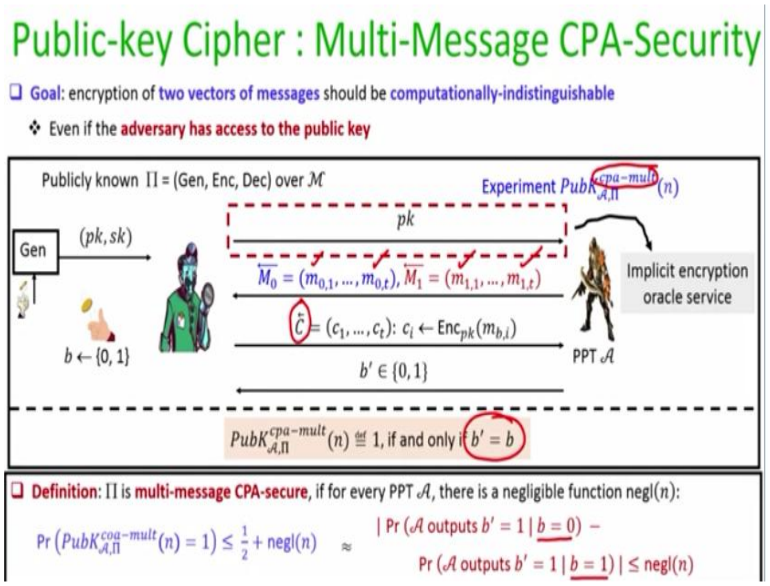
That means, if the adversary wants to encrypt some message m under the encryption process Enc_{Pub}, then it can do it itself. It does not have to ask for an encryption oracle service, because the encryption algorithm Enc_{Pub} is going to be operated with the public key pk and since the description of the encryption algorithm is publicly known and a public key is also available to the adversary, it can encrypt whatsoever message it wants to encrypt on its own without explicitly asking the experiment or the challenger.

Whereas in the symmetric key world, no encryption oracle service was provided in the COA experiment in the symmetric key world, right because as soon as we provide encryption oracle service, we go to the CPA world. That means, what we can conclude is in the public key world, COA security and single message CPA security are equivalent to each other, because the single message COA experiment in the public key world is exactly the same as the single message CPA experiment that we can imagine in the context of public key world.

Because since the public key is available to the adversary, it can encrypt whatsoever message it wants on its own and hence it has full access to encryption oracle service and this is unlike the symmetric key world, where we know that the notion of COA security and the notion of CPA security are not equivalent to each other. In fact, the pseudorandom generator based encryption process or stream cipher and the one time pad, they are COA secured.

But we know that as soon as we provide encryption oracle service, they are not CPA secure, right. So in the symmetric key world, these two notions of security are different, but surprisingly, the syntax of public key encryption process ensures that COA is, if at all we have an encryption process, which is COA secure, it automatically implies it is also single message CPA secure.

(Refer Slide Time: 21:20)



Public-key Cipher : Multi-Message CPA-Security

- **Goal**: encryption of two vectors of messages should be computationally-indistinguishable
  - Even if the adversary has access to the public key

Publicly known Π = (Gen, Enc, Dec) over $\mathcal{M}$

Experiment $PubK_{\mathcal{A},\Pi}^{cpa-mult}(n)$

Gen
(pk, sk)
$b \leftarrow \{0, 1\}$
pk
$\overleftarrow{M_0} = (m_{0,1}, \ldots, m_{0,t}), \overleftarrow{M_1} = (m_{1,1}, \ldots, m_{1,t})$
$\overleftarrow{C} = (c_1, \ldots, c_t)$: $c_i \leftarrow \text{Enc}_{pk}(m_{b,i})$
$b' \in \{0,1\}$
PPT $\mathcal{A}$
Implicit encryption oracle service

$PubK_{\mathcal{A},\Pi}^{cpa-mult}(n) \stackrel{\text{def}}{=} 1$, if and only if $b' = b$

- **Definition**: Π is multi-message CPA-secure, if for every PPT $\mathcal{A}$, there is a negligible function negl(n):

$\Pr\left(PubK_{\mathcal{A},\Pi}^{coa-mult}(n) = 1\right) \leq \frac{1}{2} + \text{negl}(n) \approx$ $|\Pr(\mathcal{A} \text{ outputs } b' = 1 \mid \underline{b = 0}) - \Pr(\mathcal{A} \text{ outputs } b' = 1 \mid \underline{b = 1})| \leq \text{negl}(n)$

So now let us see how the multi message CPA security will look like, in the public key world. So here the goal is that my encryption process should ensure that encryption of two vectors of messages should be computationally indistinguishable even if the adversary has access to the public key, which implicitly means that he has access to the encryption oracle service and this can be modeled by the experiment $PubK_{\mathcal{A},\Pi}^{cpa-mult}(n)$.

Where the rules of the game are as follows: the challenger runs the key generation algorithm obtains the pair of public key and secret key. The public key is given to the adversary and now the adversary submits a pair of challenge vectors, consisting of same number of messages, say t number of messages, where the restriction is that component wise the messages in the 0th vector and the 1st vector should be of same size.

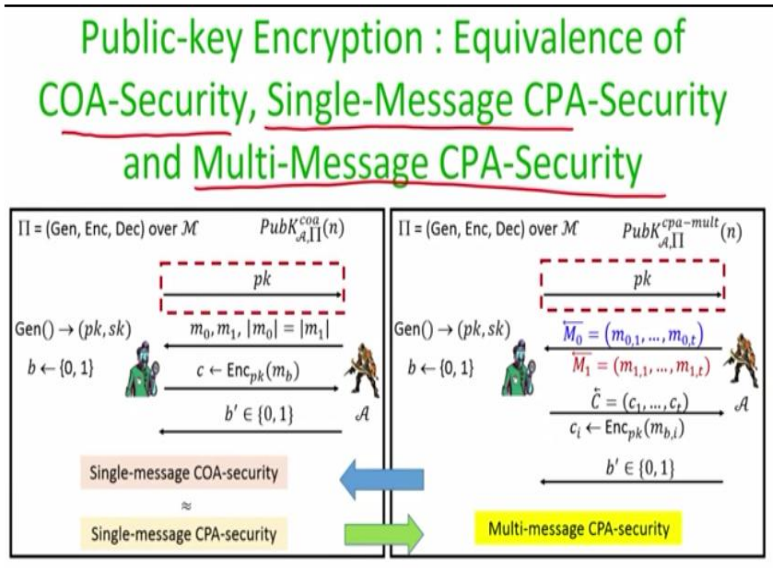
That means, the first message in the 0th vector and first message in the first vector should be of same size and in the same way, the tth message in the 0th vector and the tth message in the first vector should be of same size and so on. Other than that, there is absolutely no restriction, what could be the messages in these challenge vectors. Now the challenger generates the challenge cipher text by randomly choosing one of these two vectors with equal probability for encryption.

And once it has decided the index b, it encrypts all the messages in that particular vector and a resultant challenge cipher text vector is given as a challenge to the adversary whose goal is to now identify whether it is seeing an encryption of the messages in the 0th vector or whether it is seeing an encryption of the messages in the first vector. So it outputs a bit b' and again you see that since we are in the public key world and we are giving the adversary the public key.

That means, the adversary has implicit encryption oracle service. It does not have to explicitly ask for encryption of plain text of its choice, because it can do it itself. Now the rules of the experiment is as follows. We say that the output of the experiment is 1 or the adversary has won the experiment, if and only if b' equal to b. That means, adversary has correctly identified whether it has seen an encryption of 0th vector or the first vector.

And a security definition here is that we say that an encryption process is multi message CPA secure, for every poly time adversary participating in this experiment, there exist some negligible function, such that the probability adversary wins the experiment is upper bounded by half plus that negligible function in the security parameter, where of course the probability is taken over the random choice of the experiment, namely the random choice of the key generation algorithm. The random choice of b and random coins if at all, if there are any with respect to the adversary A. Equivalently, we say that an encryption process is multi message CPA secure, if the distinguishing advantage of the distinguisher or the adversary is upper bounded by some negligible function in the security parameter. That means, it does not matter whether the 0th vector is encrypted or the first vector is encrypted with almost the same probability the adversary A's output should be the same.

(Refer Slide Time: 24:37)



Public-key Encryption : Equivalence of COA-Security, Single-Message CPA-Security and Multi-Message CPA-Security

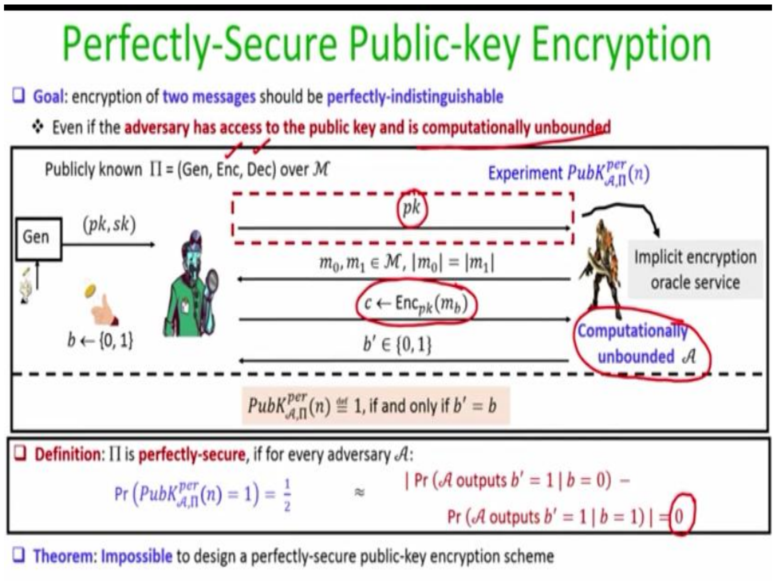
Π = (Gen, Enc, Dec) over $\mathcal{M}$
$PubK^{coa}_{\mathcal{A},\Pi}(n)$
pk
Gen() $\rightarrow (pk, sk)$
$b \leftarrow \{0,1\}$
$m_0, m_1, |m_0| = |m_1|$
$c \leftarrow \mathsf{Enc}_{pk}(m_b)$
$b' \in \{0,1\}$
$\mathcal{A}$
Single-message COA-security
$\approx$
Single-message CPA-security
Π = (Gen, Enc, Dec) over $\mathcal{M}$
$PubK^{cpa-mult}_{\mathcal{A},\Pi}(n)$
pk
Gen() $\rightarrow (pk, sk)$
$b \leftarrow \{0,1\}$
$\overleftarrow{M_0} = (m_{0,1}, \ldots, m_{0,t})$
$\overleftarrow{M_1} = (m_{1,1}, \ldots, m_{1,t})$
$\overleftarrow{C} = (c_1, \ldots, c_t)$
$c_i \leftarrow \mathsf{Enc}_{pk}(m_{b,i})$
$b' \in \{0,1\}$
$\mathcal{A}$
Multi-message CPA-security

So now in the public game encryption process, we can show that the COA security, single message CPA security and multi message CPA security are all equivalent. So here, we have the single message COA security experiment and since the public key is explicitly given to the adversary, that automatically implies that if we have an encryption process, which satisfies this notion of security, every single message COA security. Then, it automatically implies it also satisfies the definition of single message CPA security. Whereas on your right hand side, you have the indistinguishability experiment for multi message CPA security, where again the adversary is implicitly given the encryption oracle service. So it is easy to see that if you have an encryption process, which is multi message CPA secure, then of course it is single message CPA secure and automatically a single message COA secure.

This is because the multi message CPA security game is a special case of the single message security game. If I restrict the size of the vectors to consist of only single message. Interestingly, it can be proved that if we have an encryption process, which satisfies the definition of single message CPA security, then it also has to satisfy the definition of multi message CPA security. I am not going in to proof. The proof is given in the book by Katz and Lindall.

So that means, in the public key world, all these three notions of security, namely COA security, single message CPA security, and multi message CPA security are same and that is why it suffice to focus on the design of COA secure public key encryption process. That automatically implies, it satisfies the other two notions of security as well and this is unlike the symmetric key world, where we know that COA security and single message CPA security are not equivalent to each other.

(Refer Slide Time: 26:32)



Perfectly-Secure Public-key Encryption

- Goal: encryption of two messages should be perfectly-indistinguishable
  - Even if the adversary has access to the public key and is computationally unbounded

Publicly known Π = (Gen, Enc, Dec) over $\mathcal{M}$
Experiment $PubK^{per}_{\mathcal{A},\Pi}(n)$
Gen
(pk, sk)
pk
$b \leftarrow \{0, 1\}$
$m_0, m_1 \in \mathcal{M}$, $|m_0| = |m_1|$
$c \leftarrow \mathsf{Enc}_{pk}(m_b)$
$b' \in \{0, 1\}$
Implicit encryption oracle service
Computationally unbounded $\mathcal{A}$
$PubK^{per}_{\mathcal{A},\Pi}(n) \overset{\text{def}}{=} 1$, if and only if $b' = b$

- Definition: Π is perfectly-secure, if for every adversary $\mathcal{A}$:

$$\Pr\left(PubK^{per}_{\mathcal{A},\Pi}(n) = 1\right) = \frac{1}{2} \quad \approx \quad |\Pr(\mathcal{A} \text{ outputs } b' = 1 \mid b = 0) - \Pr(\mathcal{A} \text{ outputs } b' = 1 \mid b = 1)| = 0$$

- Theorem: Impossible to design a perfectly-secure public-key encryption scheme

So now, you might be wondering whether in the context of public key encryption scheme, we can design public key encryption scheme, which is perfectly secure, right. So remember, in the context of symmetric key encryption, we know how to construct perfectly secure encryption process, namely we have a candidate perfectly secure encryption process, one time pad and we have seen what exactly are the price you have to pay to achieve perfect secrecy.

So let us see whether we can achieve something similar in the context of public key encryption. So if I want to design a perfectly secure public key encryption, then my goal should be that my encryption process should ensure that encryption of two messages should be perfectly indistinguishable, even if the adversary has access to the public key and is computationally unbounded, right.

Because in the perfectly secure world, there is absolutely no restriction on the computing speed of the adversary. So if you try to model this requirement by an experiment, then the experiment will be more or less the same as the COA game in the public key world, except that my adversary is now computationally unbounded and it will have access to the encryption oracle service and we will say that the adversary has won the experiment if and only if b is equal to b'.

And my definition of perfectly secure public key encryption scheme will be, I will say that my encryption process Π is perfectly secure if the probability that any computationally unbounded adversary wins the experiment, is exactly half or equivalently there is absolutely no advantage, distinguishing advantage for the adversary A. That means, it should not be able to distinguish apart with any probability whatsoever whether the cipher text c is an encryption of m_0 or whether the cipher text c is an encryption of m_1. Interestingly, we can prove that if we are in the public key setting, then it simply impossible to design any kind of encryption scheme, which is going to satisfy the above definition. That means, we cannot design a perfectly secure encryption scheme in the public key setting and I leave the proof of this theorem as an exercise for you.

On a very high level, the reason we cannot design a perfectly secure encryption process in the public key setting, is as follows: Imagine we have a computationally unbounded adversary who knows the steps of the key generation algorithm and the decryption algorithm. Now on seeing the challenge cipher text c, it has to identify whether it is seeing an encryption of m_0 or m_1, where m_0 and m_1 are selected by the adversary itself and it knows the public key pk as well.

So now what the adversary has to do is, it knows the details of the encryption algorithm, so it knows what possible randomness my encryption algorithm can take. So it can do a kind of brute force attack, where it can search every candidate value of the randomness and use that candidate randomness and a message m_0, run the steps of the encryption process and check whether the resultant cipher text matches c or not.

My claim is that if at all there exist any candidate randomness, such that that candidate randomness along with the message m_0 under the public key pk gives you the cipher text c, then adversary can safely say that cipher text c is an encryption of m_0, otherwise it can safely say that, on the other hand, if there is no candidate randomness for which the message m_0 along with that randomness under the public key pk would have given you cipher text c, then you can safely say that the cipher text c is an encryption of m_1 and I claim that the probability that this brute force attack by the adversary correctly identifies whether c is an encryption of m_0 or m_1 is 1. That means, 100% chance the adversary is going to correctly identify, whether it is seeing an encryption of m_0 or m_1. That is the intuition behind the proof of this theorem. I am leaving the exact formal details to you as an exercise.

So that brings me to the end of this lecture. Just to summarize, in this lecture we have introduced the concept of public key cryptosystem. We have seen how exactly it differs from symmetric key cryptosystem and we have seen that in the public key world, the cipher text only attack security, single message CPA security and multi message CPA security are all equivalent. Thank you.