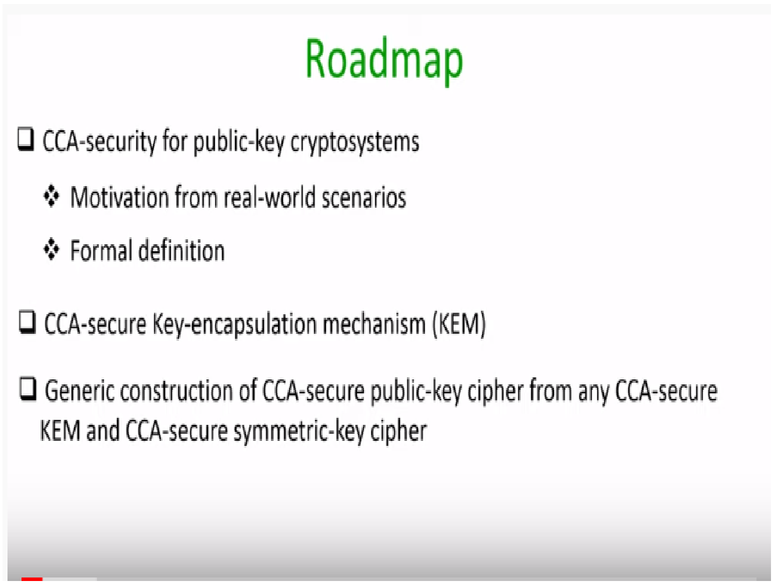
Foundations of Cryptography
Dr. Ashish Choudhury
Department of Computer Science
Indian Institute of Science – Bangalore

Lecture – 48
CCA Secure Public Key Ciphers

(Refer Slide Time: 00:34)

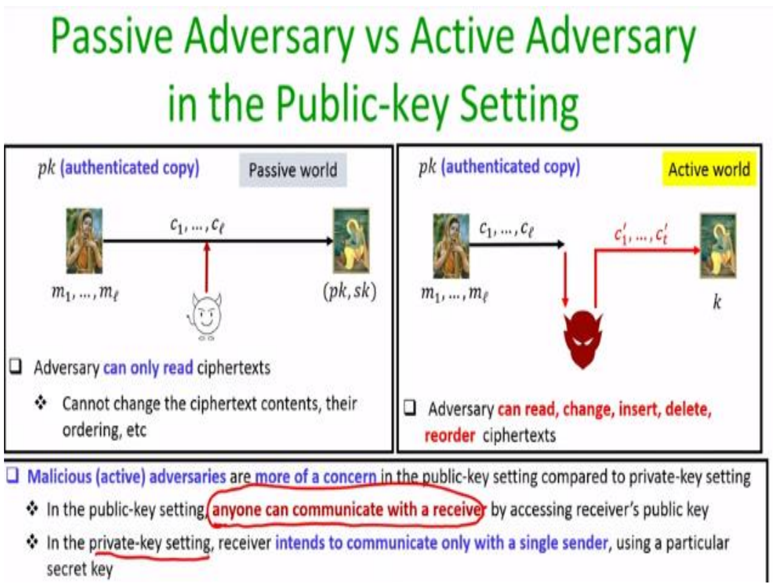


Roadmap

- ❑ CCA-security for public-key cryptosystems
  - ❖ Motivation from real-world scenarios
  - ❖ Formal definition
- ❑ CCA-secure Key-encapsulation mechanism (KEM)
- ❑ Generic construction of CCA-secure public-key cipher from any CCA-secure KEM and CCA-secure symmetric-key cipher

Hello everyone, welcome to this lecture. Just to recap, in the last lecture we have discussed rigorously the notion of CPA security in the context of public encryption schemes. So the plan for this lecture is as follows. We will introduce the notion of CCA security for public key cryptosystems. We will discuss the motivation for studying CCA security, considering real world scenarios. We will see the formal definition of CCA security Then we will see the CCA security definitions for key-encapsulation mechanism and then we will see a generic construction of CCA secure public key cipher, from any CCA secure key-encapsulation mechanism and any CCA secure symmetric-key cipher.

(Refer Slide Time: 01:13)

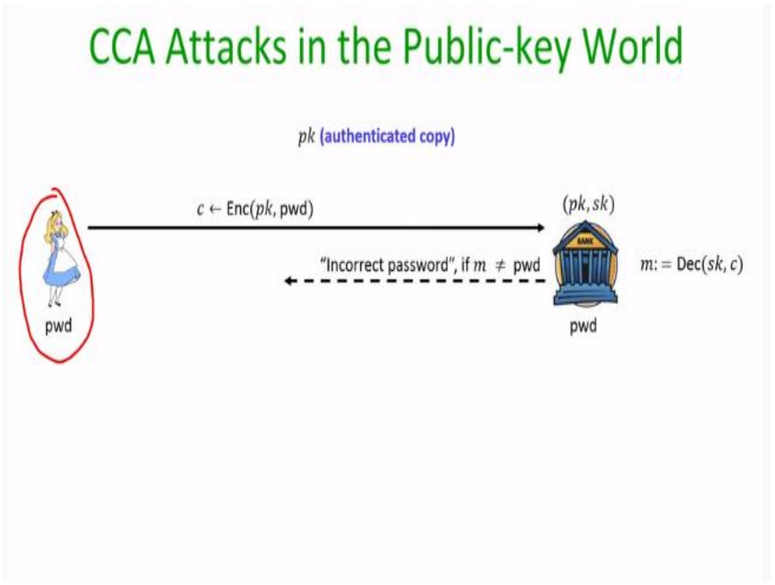


Passive Adversary vs Active Adversary in the Public-key Setting

- Adversary can only read ciphertexts
  - Cannot change the ciphertext contents, their ordering, etc
- Adversary can read, change, insert, delete, reorder ciphertexts
- Malicious (active) adversaries are more of a concern in the public-key setting compared to private-key setting
  - In the public-key setting, anyone can communicate with a receiver by accessing receiver's public key
  - In the private-key setting, receiver intends to communicate only with a single sender, using a particular secret key

So let us begin our discussion with the difference between the passive adversary model and an active adversary model, in the context of public-key setting. So if we consider the passive adversarial, then the scenario is the following. So imagine we have a receiver here who has done the key generation and he has set up its public key, made it available in the public domain. And we assume that an authenticated copy of receiver's public key is available in the public domain. And using it, say a sender encrypts a sequence of messages say m_1 to m_ℓ, and the resultant cipher text are communicated over the channel. Then in the passive model we assume that the adversary has the capability to only read the ciphertexts, it cannot change the ciphertext contents, it cannot change their ordering, it cannot introduce new ciphertext and so on. Whereas if we go in the active adversarial model then the adversary is more powerful in the sense it can not only read the ciphertexts, but it can change the cipher text contents, it can insert new ciphertext on its behalf or pretending that as if they are coming from the sender, it can delete the ciphertexts communicated by the sender, it can reorder the ciphertexts and it can do any other any kind of attack which you can think of. So it is a more powerful adversarial model compared to the passive adversarial model. And just to recall, we have seen the difference between the passive adversarial model and the active adversary model in the context of symmetric key setting. So right now we are now doing the same discussion in the context of public key setting. And it turns out that compared to the private-key setting, malicious or active adversaries are more of a concern in the public key setting. And this is because of the fact that in the public-key setting, anyone can compute or communicate with a receiver just by accessing receivers public key, because in the public key setting the encryption happens using the public key of the receiver and since it is going to be available in the public domain, if I am an adversary and I want to compute ciphertext and send it to the receiver, I can do that. This is in contrast to the private key setting, where if I am an adversary and I do not have the symmetric key which is available between the sender and the receiver, then it is very unlikely for me to come up with a valid cipher text and send it to the receiver on the behalf of the sender, if I am using an authenticated encryption scheme. So that means malicious adversaries are really more of a concern in the public key setting compared to the private key setting.

(Refer Slide Time: 03:56)



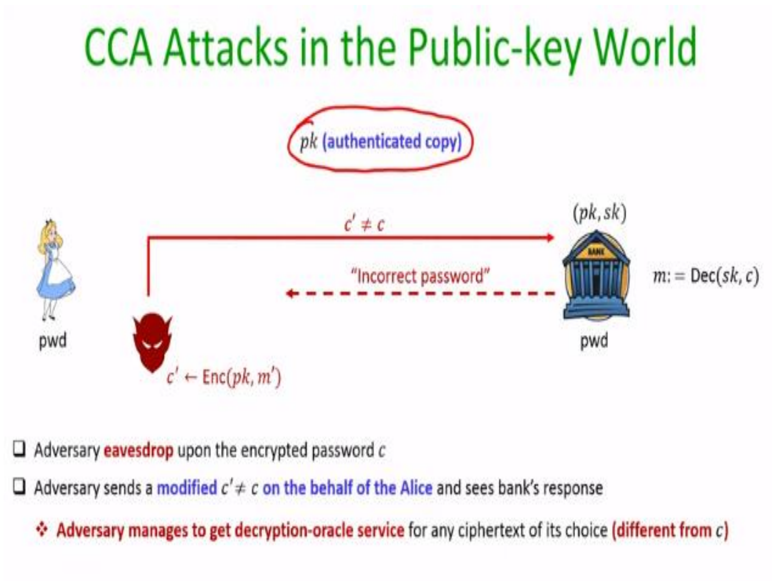
CCA Attacks in the Public-key World

pk (authenticated copy)
$c \leftarrow \text{Enc}(pk, \text{pwd})$
(pk, sk)
"Incorrect password", if $m \neq \text{pwd}$
$m := \text{Dec}(sk, c)$
pwd
pwd

So that motivates us to study CCA attacks in the context of public key world. So what we are going to do next is, we are going to see some real world scenarios where indeed CCA attacks can be launched. So consider this example where say a password is shared between a user and the bank's public key is pk and its secret key is sk. And its public key pk is available in the public domain. And the usual protocol between a legitimate user and a bank is as follows.

So if a user wants to initiate a session with the bank then the first thing the user does is it encrypts its password using the public key of the bank using some public key encryption process. And the resultant cipher text is communicated over the channel. And on receiving the encrypted password, the bank decrypts the encrypted password and compares it with the password that it has stored with itself. And it gives an error message, namely incorrect password, if it finds that on decryption that recovered password, namely m, is not the same as the password pwd, which is stored at the bank's site.

(Refer Slide Time: 05:15)


CCA Attacks in the Public-key World

- ❑ Adversary **eavesdrop** upon the encrypted password c
- ❑ Adversary sends a **modified** $c' \neq c$ **on the behalf of the Alice** and sees bank's response
  - ❖ **Adversary manages to get decryption-oracle service** for any ciphertext of its choice **(different from** c**)**

So this is a standard protocol. So now let us see what happens if you take this simple protocol in the malicious adversarial model. So imagine we are given an adversary, who is an active adversary and who has eavesdrop upon the encrypted password c. It knows the public key of the bank because its available in the public domain, but the adversary is not aware of the password which is encrypted in the cipher text c. And its goal is to find out what exactly is the password.

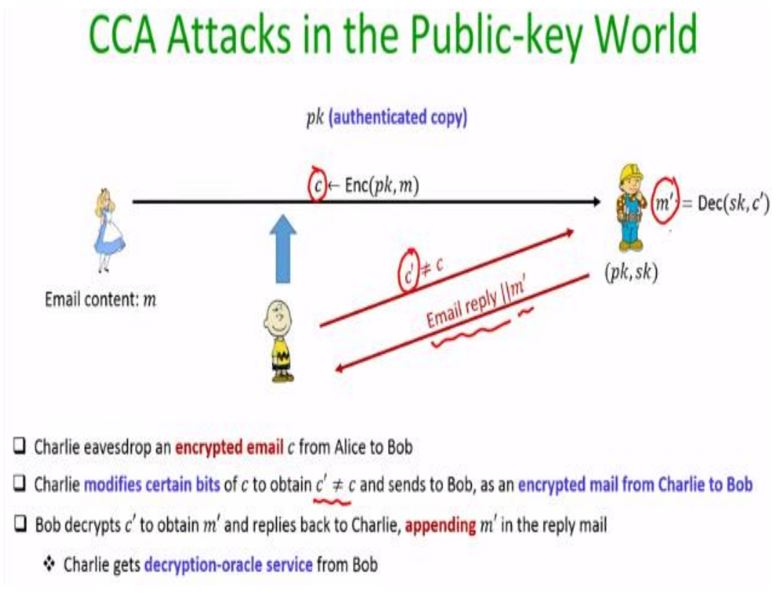
Now what the adversary can do is, it can take the ciphertext c which has been communicated by a legitimate user to the bank and it interrupts the communication. And what it does it, it modifies certain bits of the cipher text c and come up with a new ciphertext c', by encrypting some message m' itself using the public key of the bank.

So what it is doing is, it just stops the communication between the legitimate user and the bank. And instead it comes up with a new ciphertext, say c' and c' is an encryption of some known plain text m', which is known already to the adversary. And maybe we can imagine that in this particular example, the adversary can even do nasty things. In fact, it can so happen that adversary does not know m' and it just modifies certain bits of the cipher text c, and comes up with c'. And it might be the case that c' is an encryption of m', that is also a possibility.

So whatever may be the case the adversary forwards the ciphertext c' to the bank and wait for the response of the bank. Now if it sees that in the response to c', on decrypting c' the bank throws out the error message incorrect password, then basically adversary here is actually coming to know that m' is not the right password, which was encrypted in c. Because if indeed m' would have been the right password, which is encrypted in the ciphertext c, then on decrypting ciphertext c', bank will not have thrown the error message incorrect password.

But since the bank is throwing the message incorrect password, somehow the adversary here is getting to learn that the password which is shared between the legitimate user and the bank is not m', it is something different from m'. And now the adversary can try to repeat the same attack again. That means what it can do is it can just come up with another ciphertext, say c'', which could be an encryption of some plain text, say m'', and hope that indeed m'' is the right password and forwards the cipher text c'' to the bank and wait to see the bank's response. And again if the error message comes, then the adversary learns here that the password is not m'' and so on. So what is happening here, basically in this example the adversary is somehow getting a decryption oracle service from the bank without actually letting the bank know that it is adversary who is actually persuading the bank to decrypt ciphertexts of adversary's choice.

(Refer Slide Time: 08:42)



CCA Attacks in the Public-key World

- ❑ Charlie eavesdrop an **encrypted email** c from Alice to Bob
- ❑ Charlie **modifies certain bits** of c to obtain $c' \neq c$ and sends to Bob, as an **encrypted mail from Charlie to Bob**
- ❑ Bob decrypts c' to obtain m' and replies back to Charlie, **appending** m' in the reply mail
  - ❖ Charlie gets **decryption-oracle service** from Bob

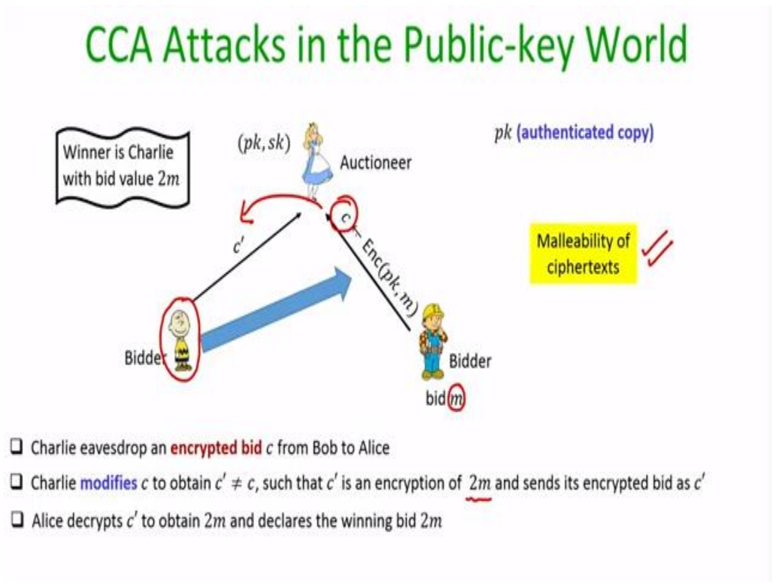
Now let us consider another application here. And here in this application, say we have a receiver Bob, who has set up its public key and secret key and the public key is available in public domain. And say Alice has an email, say m, which it wants to communicate secretly to Bob. So what it does is, it runs the public key encryption algorithm using the public key of Bob and the resultant encrypted mail c is communicated to Bob.

Now suppose Charlie is interested to find out what is happening, what exactly are the contents of the email? So what Charlie can do it is, it can eavesdrop the encrypted email and what it can do it, it can modify certain bits of c to produce a new encrypted mail say c' and send it to Bob and pretend as if that is an encrypted mail which Charlie would like to send to Bob.

Now Bob what is going to do is, when it receives the ciphertext c', it will think as if Charlie wants to send an encrypted email to Bob. And on decryption, what Bob can do is, suppose on decrypting c' it recovers the email content m'. And it might be possible that Bob would like to reply back to Charlie. And while replying back, it might want to quote the message or the email which Bob has obtained after decrypting the encrypted email c'.

That means the email reply which now Bob is sending might be concatenated with m', depending upon the underlying application. Now when this response from Bob along with the decrypted email m' comes back to Charlie, what basically Charlies getting here is, it is getting a decryption oracle service. Namely it learns that the modified ciphertext c' actually encrypts the email content m'. And in this case, actually if indeed my encryption process would have been secure CCA secure, then this this should not be possible. But since my encryption process is not CCA secure here, Bob here on receiving a modified ciphertext c' its completely clueless that the modified email has been forwarded by an adversary here. And it is simply decrypting that modified ciphertext and responding back to the adversary, thinking that the email originated from that person.

(Refer Slide Time: 11:15)


Winner is Charlie with bid value $2m$
(pk, sk)
Auctioneer
pk (authenticated copy)
c'
c
$c \leftarrow \text{Enc}(pk, m)$
Malleability of ciphertexts
Bidder
Bidder
bid m

CCA Attacks in the Public-key World

- ❑ Charlie eavesdrop an **encrypted bid** c from Bob to Alice
- ❑ Charlie **modifies** c to obtain $c' \neq c$, such that c' is an encryption of $\underline{2m}$ and sends its encrypted bid as c'
- ❑ Alice decrypts c' to obtain $2m$ and declares the winning bid $2m$

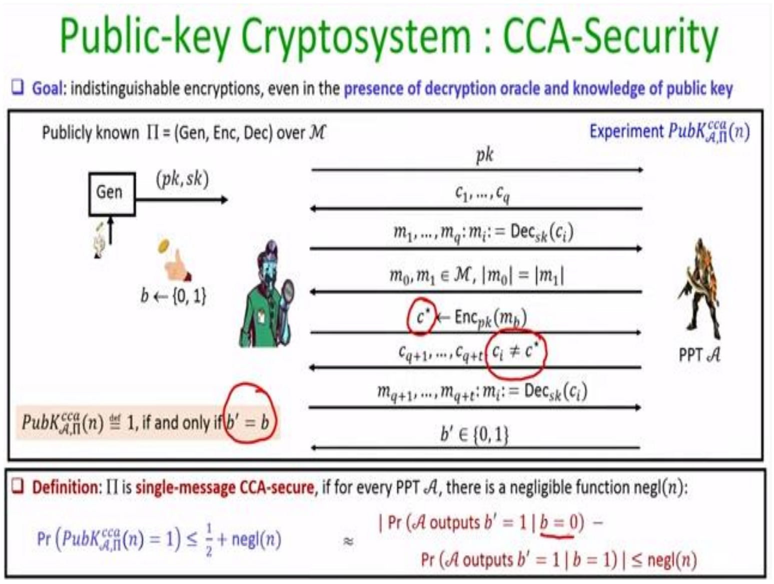
Now let us see the final example here. And this you can imagine a bidding protocol here and the scenario is the following. We have an auctioneer who has its public key set up available in the public domain. And say we have 2 bidders who are bidding for a valuable object and say the bidder Bob goes first. It has a private bid m, which it encrypts using the public key of the auctioneer, with the auctioneer in this case is Alice.

And now assume that Charlie is a malicious bidder who wants to win the bid, but it does not know the value m because that is encrypted using the public key of the Alice. So what Charlie can do here is, it can eavesdrop upon the encrypted bid of Bob. And after doing that, it can modify the encrypted bid c to another encrypted bid c'. And suppose my encryption process is such that that the modified encrypted bid c' is an encryption of the bid 2 times the bid of bob, and forwards the modified encrypted bid c' to Alice. And Charlie pretends as if c' is the bid which Charlie would like to make here.

So this property here, where it is possible for an adversary, namely malicious Charlie, to eavesdrop a ciphertext of an unknown message and from that ciphertext produce another related ciphertext c' which is an encryption of some related message, namely 2 times the message m, which was encrypted in the ciphertext c, is called as the malleability property of ciphertext. So recall when we were discussing symmetric encryption process, there also, we discussed the notion of malleability and malleability could be possible even in the context of public encryption process.

So now in this example if indeed it is possible for Charlie to convert c to c', such that c' is a public-key encryption of the message 2 times m, then what Alice might do is, when she decrypts c and c', she will find that c' is the winning bid. Because it corresponds to the value 2 times m. And she can now publicly announce that Charlie has won the bid by bidding for the bid value 2 times m. And in this case after learning the result, Charlie ends up getting a decryption oracle service from Alice and ends up winning the auction which should have been avoided if indeed my encryption process would have been CCA secure.

(Refer Slide Time: 13:55)



Public-key Cryptosystem : CCA-Security

- **Goal**: indistinguishable encryptions, even in the presence of decryption oracle and knowledge of public key

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

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

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

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

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

Gen
(pk, sk)
$b \leftarrow \{0, 1\}$
pk
$c_1, \ldots, c_q$
$m_1, \ldots, m_q : m_i := \text{Dec}_{sk}(c_i)$
$m_0, m_1 \in \mathcal{M}, |m_0| = |m_1|$
$c^* \leftarrow \text{Enc}_{pk}(m_b)$
$c_{q+1}, \ldots, c_{q+t}, c_i \neq c^*$
$m_{q+1}, \ldots, m_{q+t} : m_i := \text{Dec}_{sk}(c_i)$
$b' \in \{0, 1\}$
PPT $\mathcal{A}$

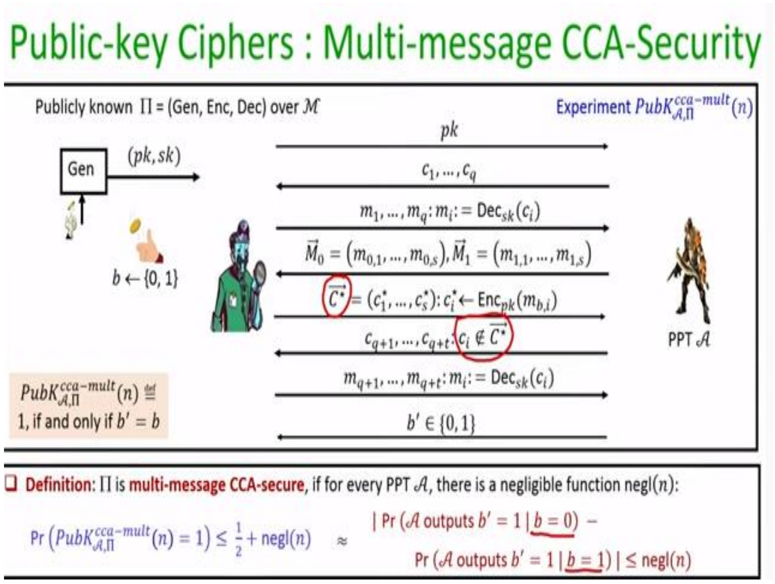
So now we have seen several real world scenarios where CCA attacks can be launched where the adversary can get the decryption oracle service. So hence now we have to study formally the notion of CCA security in the context of public key crypto system. So let us formally define that. So on a very high level, the goal of CCA security is to achieve indistinguishable encryptions, even in the presence of decryption oracle and the knowledge of public key being available with the adversary. And this is modeled by a challenge response game.

The rules of the game are as follows. The challenger runs the key-generation algorithm, gives the public key to that adversary who is computationally bounded. So since the public key is given explicitly to the adversary, it can get encryption oracle service on its own by encrypting any plain text of its choice. Now what it can do in this experiment here is, it can ask for the decryption oracle service by submitting several cipher texts from the cipher text space. And in response, the challenger has to decrypt back all those ciphertexts, using the secret key, which is not known to the adversary.

Now the challenge phase starts where the adversary submits a pair of plain texts, with the only restriction being that their lenghts should be same. And to prepare the challenge ciphertext, our challenger randomly picks one of those messages and encrypts it using the public key pk. And now we give the adversary access to the post-challenge decryption oracle service, where again it can ask for decryption oracle service or decryptions for a many ciphertexts of its choice, with the only restriction being that, that post challenge decryption oracle service is restricted, as adversary is restricted to ask for the decryption of the ciphertext $c^\star$. Because if we do not put this restriction, then we cannot achieve any meaningful notion of secrecy. And if you look the 3 motivating examples that I given earlier, in all those 3 examples the goal of the Charlie or the bad guy was to get a decryption oracle service of a modified ciphertext, but not for the ciphertext which it is interested to crack. Now once the adversary gets a decryption oracle service for the post challenge decryption oracle queries, the adversary's goal is to identify whether $c^\star$ is an encryption of m_0 or m_1. So it submits it's response or output. And the rule of the experiment is, we say that the adversary has won the experiment, which equivalent to saying that the output of the experiment is 1, if and only if $b' = b$. That means adversary has correctly identified what is encrypted in challenge ciphertext. And our security definition is, we say that our encryption processes is single message CCA secure, if for every poly time adversary, the success probability of adversary winning the game is upper bounded by half plus negligible function in the security parameter. Or equivalently, the distinguishing advantage of the adversary is upper bounded by some negligible function. Namely it does not matter, whether $c^\star$ is an encryption of m_0 or m_1, with almost same probability the response of the adversary should be the same.

The reason we are calling this experiment a single message CCA secure, because adversary has just submitted a pair of challenge plain texts and is seeing an encryption of one of them.

(Refer Slide Time: 17:13)


Public-key Ciphers : Multi-message CCA-Security

Publicly known Π = (Gen, Enc, Dec) over $\mathcal{M}$
Experiment $PubK_{\mathcal{A},\Pi}^{cca-mult}(n)$
Gen
(pk, sk)
$b \leftarrow \{0,1\}$
pk
$c_1, \ldots, c_q$
$m_1, \ldots, m_q : m_i := \mathsf{Dec}_{sk}(c_i)$
$\vec{M}_0 = (m_{0,1}, \ldots, m_{0,s}), \vec{M}_1 = (m_{1,1}, \ldots, m_{1,s})$
$\vec{C^\star} = (c_1^\star, \ldots, c_s^\star) : c_i^\star \leftarrow \mathsf{Enc}_{pk}(m_{b,i})$
$c_{q+1}, \ldots, c_{q+t} : c_i \notin \vec{C^\star}$
$m_{q+1}, \ldots, m_{q+t} : m_i := \mathsf{Dec}_{sk}(c_i)$
$b' \in \{0,1\}$
PPT $\mathcal{A}$
$PubK_{\mathcal{A},\Pi}^{cca-mult}(n) \overset{\text{def}}{=}$ 1, if and only if $b' = b$

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

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

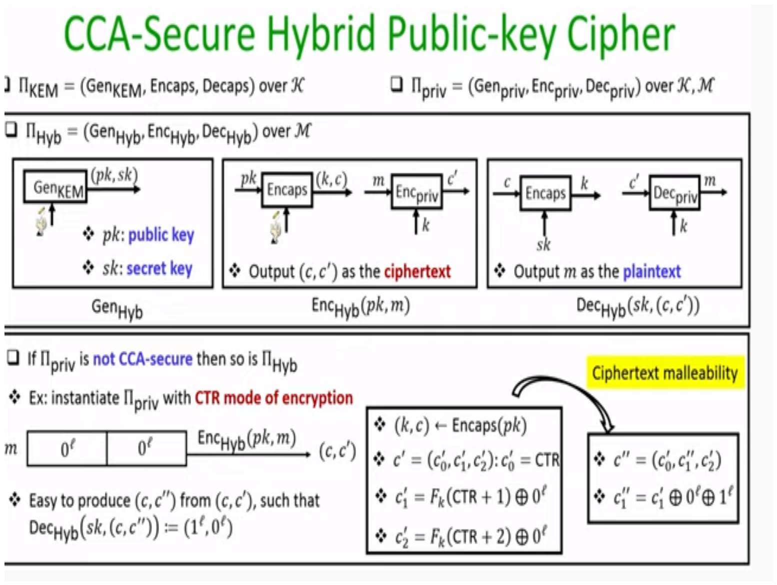
We can extend this definition in a straightforward version or the natural way, to incorporate multi message CCA security. The rules of the game will be almost same as for the single message CCA security, where challenger throws the public key to the adversary, adversary gets decryption oracle service and now in the challenge phase, it is allowed to submit a pair of vector of messages. But the only restriction being that component-wise the messages in the 0^{th} vector and the message in the 1^{st} vector, should be of the same length. To prepare the challenge cipher text, the challenger picks one of these 2 vectors with equal probability for encryption, and then encrypts all the plain texts in the selected vector and the challenge ciphertext vector is given to the adversary. The adversary is again now allowed to have access to the post challenge decryption oracle service, with the only restriction that it cannot ask for the decryption of any ciphertext which is present in the challenge ciphertext vector.

Now once our adversary is sufficiently trained, it has to identify whether the challenge ciphertext vector it has seen corresponds to an encryption of the 0^{th} vector or the first vector. And we say that adversary has won the experiment or the output of the experiment is 1, if and only if it has correctly identified whether it is vector M_0 or whether it is vector M_1, which is encrypted in the challenge ciphertext vector $C^\star$.

And our security definition is, we say that our encryption processes is multi message CCA secure if for any poly time adversary participating in this experiment, the probability that it can win the experiment is upper bounded by half plus some negligible function in the security parameter. Or equivalently, the distinguishing advantage of the adversary is upper bounded by some negligible function in the security parameter.

And as expected we can prove that single message CCA security and multimedia CCA security are equivalent, even in the context of public key encryption schemes. So I am not giving the full proof here you, can refer to the book by Katz-Lindell for the full proof.

(Refer Slide Time: 19:25)



CCA-Secure Hybrid Public-key Cipher

- $\Pi_{\text{KEM}} = (\text{Gen}_{\text{KEM}}, \text{Encaps}, \text{Decaps})$ over $\mathcal{K}$
- $\Pi_{\text{priv}} = (\text{Gen}_{\text{priv}}, \text{Enc}_{\text{priv}}, \text{Dec}_{\text{priv}})$ over $\mathcal{K}, \mathcal{M}$
- $\Pi_{\text{Hyb}} = (\text{Gen}_{\text{Hyb}}, \text{Enc}_{\text{Hyb}}, \text{Dec}_{\text{Hyb}})$ over $\mathcal{M}$

Gen_{Hyb}: $\text{Gen}_{\text{KEM}} \rightarrow (pk, sk)$
- pk: public key
- sk: secret key

$\text{Enc}_{\text{Hyb}}(pk, m)$: $pk \rightarrow \text{Encaps} \rightarrow (k, c)$; $m \rightarrow \text{Enc}_{\text{priv}} \xrightarrow{} c'$ (with k)
- Output (c, c') as the ciphertext

$\text{Dec}_{\text{Hyb}}(sk, (c, c'))$: $c \rightarrow \text{Encaps} \rightarrow k$ (with sk); $c' \rightarrow \text{Dec}_{\text{priv}} \rightarrow m$ (with k)
- Output m as the plaintext

- If Π_{priv} is not CCA-secure then so is Π_{Hyb}
  - Ex: instantiate Π_{priv} with CTR mode of encryption

$m = 0^{\ell} \,|\, 0^{\ell} \xrightarrow{\text{Enc}_{\text{Hyb}}(pk, m)} (c, c')$

- $(k, c) \leftarrow \text{Encaps}(pk)$
- $c' = (c'_0, c'_1, c'_2)$: $c'_0 = \text{CTR}$
- $c'_1 = F_k(\text{CTR} + 1) \oplus 0^{\ell}$
- $c'_2 = F_k(\text{CTR} + 2) \oplus 0^{\ell}$

Ciphertext malleability

- $c'' = (c'_0, c''_1, c'_2)$
- $c''_1 = c'_1 \oplus 0^{\ell} \oplus 1^{\ell}$

- Easy to produce (c, c'') from (c, c'), such that $\text{Dec}_{\text{Hyb}}(sk, (c, c'')) := (1^{\ell}, 0^{\ell})$

So now let us define the notion of CCA security in the context of hybrid public key ciphers. So remember in the last lecture, we have discussed that how using a key encapsulation mechanism and a symmetric encryption scheme, we can come up with a combination of both of them to come up with a more efficient hybrid encryption process. Where the key generation algorithm of the hybrid encryption process will run the key generation algorithm of the KEM and output the public key and secret key, where secret key will be available with the receiver and public key will be available in the public domain. The encryption process of the hybrid scheme will be as follows, it runs first a key encapsulation algorithm and obtains a symmetric key k and an encapsulation of the key, denoted by c. And then the key k is used to encrypt the plain text, according to the symmetric key encryption algorithm to produce the cipher text c'.

And the overall ciphertext is the encapsulation of the symmetric key and the encryption of the plain text. Analogously, the decryption happens at the receiving end; the receiver first de capsulate the encapsulation c and obtains the symmetric key k. And once it obtains the symmetric key k, it decapsulates the ciphertext component c', to recover back the plain text m, using the decryption algorithm of the symmetric encryption process.

And also to recall, in the last lecture, we have proved that if my KEM is CPA secured and my symmetric encryption process is COA secure, then the overall scheme is CPA secure. But since now we are considering CCA security, we have to identify what should be the security properties of my underlying building blocks. It turns out that if I want to achieve CCA security, then definitively my underlying symmetric encryption process in the hybrid encryption scheme should be CCA secure. It does not suffice to just have COA security or CPA security for the underlying symmetric encryption process.

To demonstrate my point, let us instantiate the underlying symmetric-key block here in this hybrid encryption process by counter mode of operation, which we know is CPA secure but not CCA secure. So imagine that a sender has encrypted a message consisting of 2 blocks of all 0s, using the hybrid encryption scheme as per the above hybrid encryption process. And the resultant ciphertext is (c, c'). And as per the details of the encryption process of this hybrid encryption scheme, the way (c, c') would have been produced is as follows.

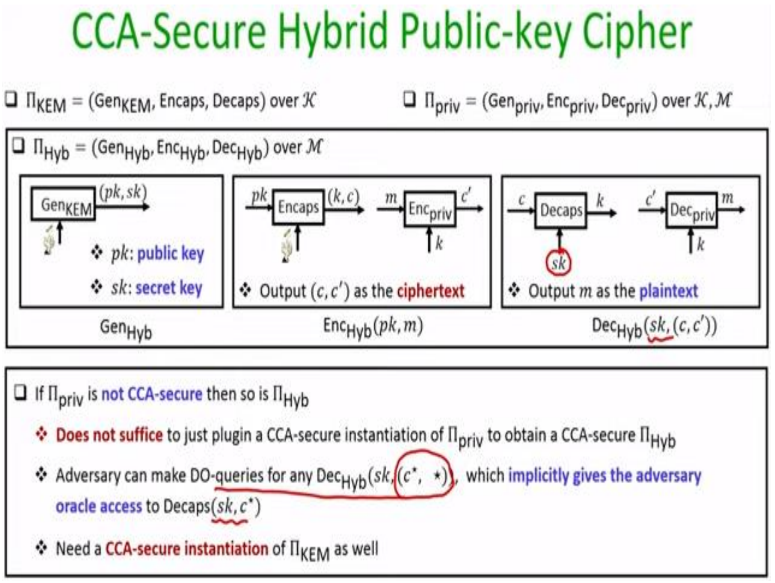
First an encapsulation algorithm would have been executed to obtain a symmetric key and the encapsulation of that key. And then using the key k, by invoking the counter mode of operation, the message block all 0s, followed by all 0s, would have been encrypted. So the encryption of the message m using the key k as per the counter mode of operation will be as follows. A random counter will be selected, which will be available as part of the cipher text component c'. And actual encryption of the blocks of the message will be c_1' and c_2' as per the counter mode of operation.

Now what we are going to see here in the example is that imagine there is an adversary who has eavesdropped the ciphertext (c, c') and imagine that the adversary is an active adversary. Then by observing (c, c'), it is very easy for the adversary to produce a modified cipher text (c, c''), such that when this modified cipher text is forwarded to the receiver and decrypted as per this hybrid encryption process, it leads to the plain text all 1s, followed by all 0s. And the way adversary can do that is by exploiting the malleability of the counter mode of operation.

Basically he has to produce c'', where the counter value namely c'_0 is retained as it is and the c'_2 component of the ciphertext c' is also retained as it is. The modification is only in the ciphertext component c'_1. c'_1 is now changed to c''_1 as $c''_1 = c'_1 \oplus 0^{\ell} \oplus 1^{\ell}$. And if c'_1 is changed to c''_1, then the effect of all 0s and all 0s cancels out.

And basically c''_1 now corresponds to a counter mode of encryption for the message block all 1's. And now this since this overall process is malleable, we can easily show that this is not going to be CCA secure. So that means if at all we want the overall hybrid encryption process to be CCA secure, definitely my underlying symmetric encryption scheme which I am using should be CCA secure.

(Refer Slide Time: 24:20)



CCA-Secure Hybrid Public-key Cipher

- $\Pi_{\text{KEM}} = (\text{Gen}_{\text{KEM}}, \text{Encaps}, \text{Decaps})$ over $\mathcal{K}$
- $\Pi_{\text{priv}} = (\text{Gen}_{\text{priv}}, \text{Enc}_{\text{priv}}, \text{Dec}_{\text{priv}})$ over $\mathcal{K}, \mathcal{M}$
- $\Pi_{\text{Hyb}} = (\text{Gen}_{\text{Hyb}}, \text{Enc}_{\text{Hyb}}, \text{Dec}_{\text{Hyb}})$ over $\mathcal{M}$

Gen_{KEM}
(pk, sk)
pk
Encaps
(k, c)
m
Enc_{priv}
c'
k
c
Decaps
k
c'
Dec_{priv}
m
sk
k

- pk: public key
- sk: secret key

Gen_{Hyb}

- Output (c, c') as the ciphertext

$\text{Enc}_{\text{Hyb}}(pk, m)$

- Output m as the plaintext

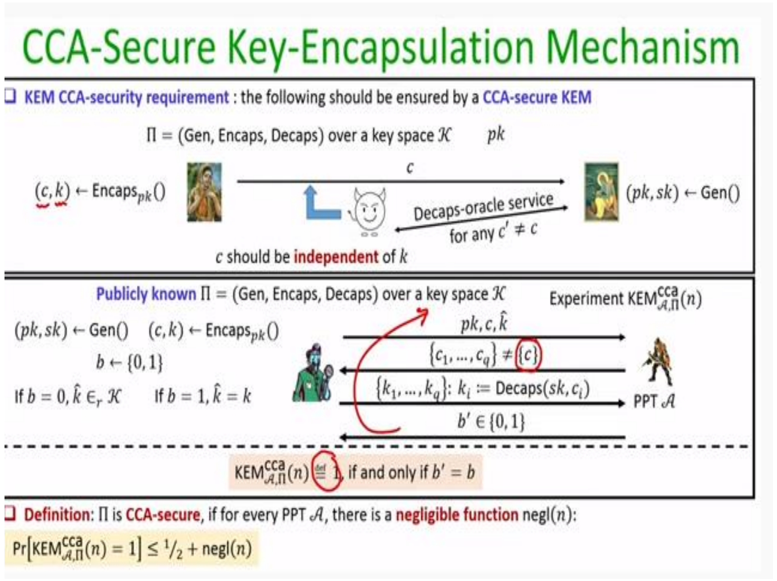
$\text{Dec}_{\text{Hyb}}(sk, (c, c'))$

- If Π_{priv} is not CCA-secure then so is Π_{Hyb}
  - Does not suffice to just plugin a CCA-secure instantiation of Π_{priv} to obtain a CCA-secure Π_{Hyb}
  - Adversary can make DO-queries for any $\text{Dec}_{\text{Hyb}}(sk, (c^*, \star))$, which implicitly gives the adversary oracle access to $\text{Decaps}(sk, c^*)$
  - Need a CCA-secure instantiation of Π_{KEM} as well

But it turns out that just by instantiating the underlying symmetric encryption by a CCA secure symmetric encryption, does not suffice to give us an overall CCA secure hybrid encryption process. And the reason for this is, if you play the CCA GAME against this hybrid encryption process then remember that in the CCA game, adversary is given access to the decryption oracles service. Namely the adversary can now make decryption oracle service for any kind of modified cipher text, where the first part of the ciphertext can be any $c^{\star}$, namely any encapsulation, followed by anything. And to respond to those modified decryption oracle queries, the challenger has to basically decrypt such modified cipher texts using the secret key. Now if you see the decryption algorithm of this hybrid encryption scheme, any decryption oracle query of the form $c^{\star}$ followed by anything, when it gets decrypted by the challenger in the CCA game, basically it provides implicitly the adversary oracle access to the decapsulation oracle service, under the unknown secret key. Because on decrypting the modified ciphertext, adversary will come to learn what exactly is the decapsulation of $c^{\star}$, under the unknown secret key sk.

That means we now need a CCA secure instantiation of the key encapsulation mechanism as well, to hope that the overall hybrid encryption process results in a CCA secure public encryption process.

(Refer Slide Time: 26:01)



CCA-Secure Key-Encapsulation Mechanism

- KEM CCA-security requirement : the following should be ensured by a CCA-secure KEM

Π = (Gen, Encaps, Decaps) over a key space $\mathcal{K}$ pk

$(c, k) \leftarrow \text{Encaps}_{pk}()$ c $(pk, sk) \leftarrow \text{Gen}()$

Decaps-oracle service for any $c' \neq c$

c should be independent of k

Publicly known Π = (Gen, Encaps, Decaps) over a key space $\mathcal{K}$ Experiment $\text{KEM}^{\text{cca}}_{\mathcal{A},\Pi}(n)$

$(pk, sk) \leftarrow \text{Gen}()$ $(c, k) \leftarrow \text{Encaps}_{pk}()$

$b \leftarrow \{0,1\}$

If $b = 0, \hat{k} \in_r \mathcal{K}$ If $b = 1, \hat{k} = k$

$pk, c, \hat{k}$

$\{c_1, \ldots, c_q\} \neq \{c\}$

$\{k_1, \ldots, k_q\}$: $k_i := \text{Decaps}(sk, c_i)$

$b' \in \{0,1\}$

PPT $\mathcal{A}$

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

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

$\Pr[\text{KEM}^{\text{cca}}_{\mathcal{A},\Pi}(n) = 1] \leq {}^{1}/_{2} + \text{negl}(n)$

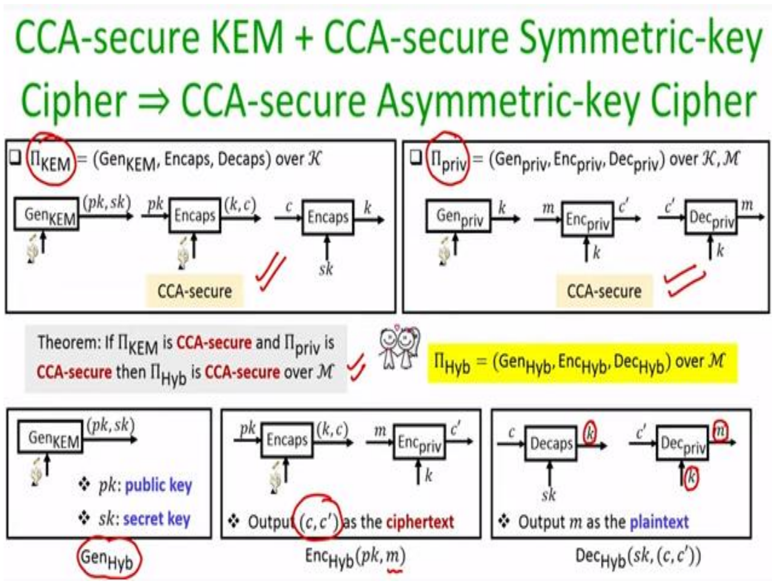
So let us first define the notion of CCA security for key encapsulation mechanism. And on a very high level, the goal of a CCA secure key encapsulation mechanism should be to ensure the following. So imagine we have a receiver who runs the key generation algorithm of a CCA secure encapsulation mechanism and set up the public key. And say our sender is there which runs the key encapsulation algorithm of that scheme, obtains a secret key k, and an encapsulation of the key c. And the encapsulation is sent to the receiver. And say there is a malicious adversary, who has eavesdropped the encapsulation c.

And now imagine that my adversary gets the decapsulation oracle service for any encapsulation c', different from c. Now by getting polynomial number of decapsulation oracle service, we require that from the viewpoint my adversary, the encapsulation c, which it has seen earlier, should be still independent of the key k, which is encapsulated in c. So the advantage here that my adversary is now getting is an explicit decapsulation oracle service, which we have to now model in our experiment.

So to model above requirement, the experiment is as follows. Our challenger runs the key generation algorithm and using the public key it runs the encapsulation algorithm to obtain a pair (c, k). And now it prepares the challenge for the adversary as follows. It tosses a fair coin, if the coin tosse is 0, then it picks a random element $\hat{k}$ from the key space. Whereas if the coin toss is 1, then the element $\hat{k}$ is the key k, which is actually encapsulated in the encapsulation c. And the challenge for the adversary is as follows. The public key is given, the encapsulation c is given and $\hat{k}$ is given. And the goal of the adversary is to identify whether $\hat{k}$ is a random element from the key space, namely whether $b = 0$. Or whether $\hat{k}$ is the same key which is encapsulated in c, namely $b = 1$. But now we model that decapsulation oracle service by allowing the adversary to ask for decapsulation of any encapsulation of its choice, with the only restriction being that this decapsulation oracle service should not be for the encapsulation c. They should be different from c. And our adversary is allowed to adaptively submit its query and in response to the decapsulation oracle queries, the challenger respond by decapsulating all those queries under the unknown secret key sk, which is not known to the adversary. And after making polynomial number of queries, the adversary has now has to identify and solve its challenge.

Namely it has to identify whether he has seen a challenge as per method $b = 0$ 0 or as per the method $b = 1$. And the definition of the experiment is, we say that adversary has won the experiment, which we denote by saying that output of the experiment is 1, if and only if adversary has ensured $b' = b$. And we say that our key encapsulation mechanism is CCA secure, if for every poly time adversary, there exists some negligible function, such that the probability of that adversary winning the experiment is upper bounded by half plus negligible function. Or equivalently the distinguishing advantage of that adversary is upper bounded by some negligible function in the security parameter.

(Refer Slide Time: 29:42)



CCA-secure KEM + CCA-secure Symmetric-key Cipher ⇒ CCA-secure Asymmetric-key Cipher

Theorem: If Π_{KEM} is CCA-secure and Π_{priv} is CCA-secure then Π_{Hyb} is CCA-secure over $\mathcal{M}$

So now we are going to see that, if we are given a CCA-secure KEM and a CCA secure symmetric key cipher, then if we combine them, we get a CCA secure asymmetric key cipher. So imagine we are given a CCA secure KEM and the CCA secure symmetric key encryption process. Then we can combine it in the same way as we have done to obtain a CPA secure asymmetric key cipher in the last lecture.

So my key generation algorithm of the hybrid encryption process will be simply the key generation algorithm of the key encapsulation mechanism. To encrypt a plain text using the public key pk, what sender is going to do is, it will run the encapsulation algorithm and will obtain a key k and its encapsulation c. And using the key k, it will invoke the encryption algorithm of the underlying symmetric encryption process to encrypt the plain text m and obtain its encryption c'. And the overall cipher text will be (c, c').

On the other hand, the receiver who possesses the secret key sk, on the receiving the ciphertext (c, c'), will first decapsulate the c part of the ciphertext to retrieve the encapsulated key k. And then that key k is used to decrypt the c' component of the cipher text, as per the decryption algorithm of the symmetric encryption process, to get back the actual plain text m.

And we can prove that if my key encapsulation mechanism is CCA secure as per the definition that we have just given. And if my underlying symmetric encryption processes is CCA secure, then this generic way of combining these 2 primitives is going to give us a public key encryption process which is CCA secure. And the proof again will be something similar to the hybrid argument style proof, which we had given in the last lecture, to prove the CPA security of the generic construction of the hybrid scheme that we had discussed in that lecture. So I am leaving the full formal details of the proof to you as an exercise.

So that brings me to the end of this lecture. Just to summarize, in this lecture we introduced the notion of CCA security in the context of public encryption scheme. We have seen the malleability, what exactly malleability of public encryptions schemes means and we have seen the definition of CCA security for key encapsulation mechanism and discussed that if we are given a CCA secure key encapsulation mechanism and a CCA secure symmetric decipher, then we can combine them generically to obtain a hybrid encryption process which is a public key hybrid encryption process and CCA secure. Thank you.